

Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification

Palakh Mignonne Jude

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Application

Edward A. Fox, Chair

Anuj Karpatne

Christopher North

June 9, 2020

Blacksburg, Virginia

Keywords: Electronic Theses and Dissertations, Classification, Machine Learning, Deep
Learning, Natural Language Processing

Copyright 2020, Palakh Mignonne Jude

Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification

Palakh Mignonne Jude

(ABSTRACT)

Great progress has been made to leverage the improvements made in natural language processing and machine learning to better mine data from journals, conference proceedings, and other digital library documents. However, these advances do not extend well to book-length documents such as electronic theses and dissertations (ETDs). ETDs contain extensive research data; stakeholders – including researchers, librarians, students, and educators – can benefit from increased access to this corpus. Challenges arise while working with this corpus owing to the varied nature of disciplines covered as well as the use of domain-specific language. Prior systems are not tuned to this corpus. This research aims to increase the accessibility of ETDs by the automatic classification of chapters of an ETD using machine learning and deep learning techniques. This work utilizes an ETD-centric target classification system. It demonstrates the use of custom trained word and document embeddings to generate better vector representations of this corpus. It also describes a methodology to leverage extractive summaries of chapters of an ETD to aid in the classification process. Our findings indicate that custom embeddings and the use of summarization techniques can increase the performance of the classifiers. The chapter-level labels generated by this research help to identify the level of interdisciplinarity in the corpus. The automatic classifiers can also be further used in a search engine interface that would help users to find the most appropriate chapters.

Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification

Palakh Mignonne Jude

(GENERAL AUDIENCE ABSTRACT)

Electronic Theses and Dissertations (ETDs) are submitted by students at the end of their academic study. These works contain research information pertinent to a given field. Increasing the accessibility of such documents will be beneficial to many stakeholders including students, researchers, librarians, and educators. In recent years, a great deal of research has been conducted to better extract information from textual documents with the use of machine learning and natural language processing. However, these advances have not been applied to increase the accessibility of ETDs. This research aims to perform the automatic classification of chapters extracted from ETDs. That will reduce the human effort required to label the key parts of these book-length documents. Additionally, when considered by search engines, such categorization can aid users to more easily find the chapters that are most relevant to their research.

Dedication

To my beloved parents Chris and Lolita Jude and my darling sister Naina Cara Jude.

Acknowledgments

I would like to thank my advisor and mentor Dr. Edward Fox for his guidance, support, and efforts, without which this research would not have been possible. Your wise words and timely suggestions helped me throughout my Master's degree. I would also like to thank Dr. North and Dr. Karpatne for their time and valuable suggestions that helped in the completion of this study. I would like to acknowledge William Ingram for his invaluable suggestions and assistance throughout this research. I would also like to acknowledge the Computer Science Department for the opportunity to pursue and fund my Master's degree at Virginia Tech, and Sharon for helping out with all administrative issues and queries.

I thank Bipasha for her patience and guidance with this research and beyond, and for all the laughs and tears we've shared together. I would like to thank Siddharth for ensuring that I did my best during my Master's, and being a constant source of hope. I am grateful to Asmita and Sandhya for their constant support in every aspect of my life over the past two years. I thank Nilesh for keeping track of my daily tasks and ensuring that I was sticking to my timeline. I thank Maanav for always helping me keep things in perspective. I am extremely grateful to Prapti for her ability to calm me down in moments of panic.

I thank Prashant and Satvik for their help and for patiently listening to my multiple queries. I thank Venkat Srinivasan for the various discussions related to his work that helped shape this research. I also thank all the members of the Digital Library Research Laboratory for their support and suggestions.

I thank Aayush Vats for being a constant support in my life for many many years. I thank Saloni Bali for her support and ability to understand me better than I understand myself. I also thank Kim, my oldest friend, for sticking around all these years.

This project was made possible in part by the Institute of Museum and Library Services grant LG-37-19-0078-19.

Contents

List of Figures	xii
List of Tables	xix
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Motivation	3
1.4 Hypotheses	3
1.5 Research Questions	4
1.6 Research Contributions	5
1.7 Outline of the Thesis	5
2 Background	7
2.1 Embeddings	7
2.1.1 Word Embeddings	7
2.1.2 Document Embeddings using Doc2Vec	11
2.2 Multi-label Classification	12
2.2.1 One vs. Rest	13

2.2.2	Binary Relevance	13
2.2.3	Classifier Chains	13
2.2.4	Label Powerset	14
2.2.5	Adaptive Algorithm	14
2.3	Machine Learning Classifiers	14
2.3.1	Logistic Regression	14
2.3.2	Support Vector Machine	15
2.3.3	Random Forest	15
2.4	Deep Learning Architectures	15
3	Review of Literature	18
3.1	Chapter Segmentation	18
3.1.1	Text Extraction from PDFs	18
3.1.2	Segmenting PDFs into Sections	21
3.2	Summarization	22
3.2.1	Extractive Summarization	23
3.3	Classifiers	24
3.3.1	Machine Learning Classifiers	25
3.3.2	Deep Learning Classifiers	29
3.3.3	Evaluation	36

4	Data	39
4.1	Data Description	39
4.1.1	Data for Baseline Models	40
4.1.2	Data for Embeddings	42
4.1.3	Data for Deep Learning Models	47
4.1.4	Number of Chapters in Each Subject Category	70
4.1.5	Number of Words in Each Chapter for Each Subject Category	71
5	Chapter Segmentation	72
5.1	Comparison of Text Extraction Tools from PDFs	72
5.1.1	PDFMiner	75
5.1.2	PyPDF2	76
5.1.3	PyMuPDF	76
5.1.4	Pdftotext	78
5.1.5	ABBYY Cloud OCR SDK	78
5.1.6	Textextract	82
5.1.7	Tika-Python	84
5.2	Comparison of Segmentation Tools	86
5.2.1	Grobid	86
5.2.2	ITCore (Intelligent Textbooks Core)	87

6	Summarization	94
6.1	TextRank	97
6.1.1	Summaries	97
6.1.2	Keywords	99
6.2	LexRank	99
6.3	Luhn’s Algorithm	100
6.4	Latent Semantic Analysis (LSA)	101
7	Classification	103
7.1	System Design Overview	103
7.1.1	PDF Extraction and Embedding Process	104
7.1.2	Classification Models Training and Testing Process	105
7.2	Embeddings	107
7.2.1	Data Pre-processing	108
7.2.2	Embedding Parameters	109
7.3	Multi-Label Classification	110
7.4	Virginia Tech Dataset	110
7.4.1	Data Pre-Processing	116
7.5	PQDT Dataset	122
7.5.1	Data Pre-processing	123

7.6	Evaluation Metrics	126
8	Results and Discussion	128
8.1	Virginia Tech Dataset	128
8.1.1	BinaryRelevance	129
8.1.2	LabelPowerset	131
8.2	PQDT Dataset	139
8.2.1	Machine Learning Models	141
8.2.2	Deep Learning Models	157
9	Future Work	174
10	Conclusions	175
	Bibliography	176
	Appendices	186
	Appendix A Data Description	187
	Appendix B Classifier Performance	190
B.1	Machine Learning Classifiers	190
B.2	Deep Learning Classifiers	220

List of Figures

2.1	Neural Network architecture for Word2Vec [35]	8
2.2	Conceptual model for GloVe [58]	9
2.3	fastText skip gram model topography [64]	10
2.4	PV-DM and PV-DBOW variants of Doc2Vec [26]	11
2.5	Illustration of a traditional RNN [46]	16
2.6	Illustration of an LSTM [46]	16
2.7	Illustration of a GRU [41]	17
3.1	Eleven CCS top level branches used to build the ACM Digital Library Classifiers [11]	25
3.2	An example of how CCS terms are displayed in a published work [2]	26
3.3	Hierarchical Transform BERT model [49]	29
3.4	Results for the Hierarchical Transform BERT model [49]	30
3.5	Model architecture of proposed model LSTM _{reg} : (a) input word embeddings, (b) BiLSTM, (c, d) concatenated forward $h^f_{1:n}$ and backward $h^b_{1:n}$ hidden features, (e) max-pooling overtime, (f) document feature vector, (g) softmax or sigmoid output [4]	31
3.6	Comparison of LSTM _{reg} results with other models [4]	31
3.7	Comparison of KD-LSTM _{reg} results with other models [3]	32

3.8	Overall architecture of BiLSTM model [70]	33
3.9	Document embedding process through BiLSTM framework [70]	34
3.10	Performance of models using simple linear classifier [70]. W_c indicates average words per chunk.	35
3.11	Performance of models using simple SVM classifier [70]. W_c indicates average words per chunk.	35
4.1	Distribution of ETDs present in each of the 21 secondary headings [6]	40
4.2	Distribution of ETDs with multiple subject categories [6]	41
4.3	JSON representation of partial metadata from a sample ETD [6]	42
4.4	Part 1: Co-occurrence of the 28 subject categories	56
4.5	Part 2: Co-occurrence of the 28 subject categories	56
4.6	Co-occurrence heat map of the 28 subject categories (without zero values)	57
4.7	Number of STEM, non-STEM, and STEM + non-STEM ETDs	65
4.8	Average number of words present in the title of ETDs for each subject category	67
4.9	Average number of words present in the abstract of ETDs for each subject category	69
4.10	Average number of words present in the full-text of ETDs for each subject category	69
4.11	Average number of chapters in each subject category	70
4.12	Average number of words present in chapters of ETDs for each subject category	71

5.1	Snippet from the ‘Introduction’ chapter of a sample ETD [68]	73
5.2	Snippet of the HTML generated page of the ‘Introduction’ chapter of a sample ETD using PDFMiner	74
5.3	HTML markup of the ‘Introduction’ chapter of a sample ETD using PDFMiner	75
5.4	Snippet of the ‘Introduction’ chapter of a sample ETD using PyPDF2	77
5.5	Snippet of the ‘Introduction’ chapter of a sample ETD using PyMuPDF	78
5.6	Snippet of the ‘Introduction’ chapter of a sample ETD using pdftotext	79
5.7	Snippet of the DOCX generated for the ‘Introduction’ chapter of a sample ETD using ABBYY Cloud OCR SDK	81
5.8	Snippet of the .txt generated for the ‘Introduction’ chapter of a sample ETD using ABBYY Cloud OCR SDK	82
5.9	Snippet of the ‘Introduction’ chapter of a sample ETD using <code>textextract</code>	83
5.10	Snippet of the ‘Introduction’ chapter of a sample ETD using <code>Tika-Python</code>	85
5.11	Number of chapters identified for the Computer Science edpartment using Grobid	87
5.12	Table of Contents of the sample ETD titled ‘Learning And Inference Algorithms for Dynamical System Models Of Dextrous Motion’ [69]	88
5.13	Number of chapters identified for the Computer Science subject category using IT Core	90
5.14	First and last pages of the chapter segmented PDFs of the sample ETD titled ‘Learning And Inference Algorithms for Dynamical System Models Of Dextrous Motion’ [69] using the ITCore tool	92

6.1	Chapter 1: Page 1 of the sample ETD [27]	95
6.2	Chapter 1: Page 2 of the sample ETD [27]	96
6.3	Summary of the sample ETD generated using TextRank with ratio=0.2	98
6.4	100 word count summary of the sample ETD generated using TextRank	98
6.5	Keywords of the sample ETD generated using TextRank	99
6.6	Summary of the sample ETD generated using LexRank	100
6.7	Summary of the sample ETD generated using Luhn's Algorithm	101
6.8	Summary of the sample ETD generated using LSA	102
6.9	Summary of the sample ETD generated using LSA with stopwords	102
7.1	System overview: Extraction and embedding process	104
7.2	System overview: Classification workflow	105
7.3	Folder contents of the dissertation with handle number 83391	111
7.4	Entry in MongoDB with search fields for a sample ETD	112
7.5	Contents file of an ETD folder with a single PDF	114
7.6	PDF files of the dissertation with handle number 73236 with support PDF	114
7.7	Contents file of an ETD folder with a single PDF along with supporting documents	114
7.8	Special Case: Contents file of an ETD folder with a single PDF along with supporting documents	115
7.9	PDF files of the dissertation with handle number 26928 with multiple PDFs	115

7.10	Contents file of an ETD folder with multiple PDFs	116
7.11	Entry in the <code>metadata_consolidated</code> MongoDB collection for a sample ETD	118
7.12	Sample 5 ETD records with transformed class labels	120
7.13	Sample 5 ETD records with transformed class labels (with probability scores)	121
8.1	Various experimental setups	129
8.2	Analysis of 5 differently classified ETDs using the BinaryRelevance approach	131
8.3	Analysis of 5 differently classified ETDs using the BinaryRelevance approach (with abstract)	132
8.4	Analysis of 5 incorrectly classified ETDs using the LabelPowerset approach .	134
8.5	Analysis of 5 incorrectly classified ETDs using the LabelPowerset approach (with abstract information)	135
8.6	Probability values predicted by the classifier (BOW + RF)	137
8.7	Probability values predicted by the classifier (TF-IDF with bi-grams + SVM)	137
8.8	Predicted labels after picking the top 3 subject categories (BOW + RF) . .	137
8.9	Expected and Predicted labels after picking the top 3 subject categories (BOW + RF)	138
8.10	Analysis of 5 incorrectly classified ETDs (with probability scores)	139
8.11	Various experimental setups	140
8.12	Chapter-level labels for the ETD titled ‘Quasi-one-dimensional models for glassy dynamics’	154

8.13	Performance of SVM trained on Chapter subset data (summary data using Luhn’s Algorithm) per subject category at the chapter-level	155
8.14	Part 1: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the SVM model trained using the Chapter subset data (summary data using Luhn’s Algorithm) . . .	156
8.15	Part 2: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the SVM model trained using the Chapter subset data (summary data using Luhn’s Algorithm) . . .	156
8.16	STEM and non-STEM categories predicted for chapters of ETDs by the SVM model trained using the Chapter subset data (summary data using Luhn’s Algorithm)	157
8.17	Visual representation of accuracy and loss values across epochs	159
8.18	Visual representation of F1-score and loss values across epochs for the best performing LSTM on the Full-text (All data)	169
8.19	Performance of LSTM trained on Full-text (All data) at the chapter-level . .	171
8.20	Part 1: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the LSTM model trained on Full-text (All data)	172
8.21	Part 2: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the LSTM model trained on Full-text (All data)	172
8.22	STEM and non-STEM categories predicted for chapters of ETDs by the LSTM model trained using Full-text (All data)	173

A.1	Part 1: Co-occurrence of the 28 subject categories	188
A.2	Part 2: Co-occurrence of the 28 subject categories	189

List of Tables

3.1	Top classifiers for the ACM dataset [11]	27
3.2	Multilayer Perceptron performance compared to other ML classifiers [74]. . .	28
4.1	Number of ETDs in each department under the Virginia Tech thesis collection	43
4.2	Number of ETDs in each department under the Virginia Tech dissertation collection	44
4.3	10 most commonly occurring keywords in the Virginia Tech thesis collection	45
4.4	10 most commonly occurring keywords in the Virginia Tech dissertation col- lection	45
4.5	Number of ETDs in each program under the Pennsylvania State University ETD Collection	46
4.6	Number of ETDs in each department under the University of Illinois at Urbana-Champaign ETD collection	48
4.7	10 most commonly occurring keywords in the University of Illinois at Urbana- Champaign ETD collection	49
4.8	Number of ETDs in each subject category under the PQDT ETD collection (alphabetic order)	50
4.9	Number of ETDs in each subject category under the PQDT ETD collection (ordered by count)	50

4.10	Number of ETDs in the largest 10 universities (based on number of ETDs present)	51
4.11	Number of ETDs from Historically Black Colleges and Universities (HBCUs)	51
4.12	Number of ETDs associated with different years of publication	52
4.13	Part 1: Number of ETDs contributed by the largest 3 universities under each subject category	53
4.14	Part 2: Number of ETDs contributed by the largest 3 universities under each subject category	54
4.15	Part 3: Number of ETDs contributed by the largest 3 universities under each subject category	55
4.16	Top 3 most frequent departments present in our largest 10 universities. Count of ETDs associated with each department is in brackets.	58
4.17	Part 1: Most frequent keywords in each subject category. Frequency of the keywords is represented in brackets.	59
4.18	Part 2: Most frequent keywords in each subject category. Frequency of the keywords is represented in brackets.	60
4.19	Part 1: Most common departments associated with each subject category. Count of ETDs associated with each department is in brackets.	61
4.20	Part 2: Most common departments associated with each subject category. Count of ETDs associated with each department is in brackets.	62
4.21	Number of keywords associated with each subject category (alphabetic order)	63
4.22	Number of keywords associated with each subject category (ordered by count)	63

4.23	Categorization of subject categories into STEM and non-STEM	64
4.24	Number of STEM and non-STEM ETDs associated belonging to the largest 10 universities	66
4.25	Part 1: Number of ETDs that contain subject categories from STEM and non-STEM	67
4.26	Part 2: Number of ETDs that contain subject categories from STEM and non-STEM	68
7.1	Counts of ETDs in train-test splits for the different experimental setups . . .	106
7.2	Set of parameter variations for fastText	109
7.3	Set of parameter variations for Doc2Vec	110
7.4	Set of features used to train the model	119
7.5	Example ETD with multiple subject categories	120
7.6	Default parameter values utilized for Logistic Regression	121
7.7	Parameter values selected for Support Vector Machine	122
7.8	Parameter values selected for Random Forest	122
7.9	Set of features used to train the model	124
7.10	Set of combined chapter summary features used to train the model	124
7.11	Parameter values selected for Support Vector Machine	125
7.12	Parameter values selected for Random Forest	125

8.1	Performance of various experimental setups using BinaryRelevance (without abstract)	130
8.2	Performance of various experimental setups using BinaryRelevance (with abstract information)	130
8.3	Performance of various experimental setups using LabelPowerset (without abstract)	133
8.4	Performance of various experimental setups using LabelPowerset (with abstract)	133
8.5	Performance of various experimental setups using LabelPowerset (without abstract) with probability scores class labels	136
8.6	Performance of various experimental setups using LabelPowerset (with abstract) with probability scores class labels	136
8.7	Performance of SVM and RF built using pre-trained embeddings	141
8.8	Performance of best RF and SVM models on the Full-text (Half data)	143
8.9	Performance of best RF and SVM models on the Full-text (All data)	144
8.10	Performance of best RF and SVM models on the chapter subset data (full-text data)	145
8.11	Performance of best RF and SVM models on the chapter subset data (summary data) using gensim's TextRank Generated Summary with ratio of 0.2	147
8.12	Performance of best RF and SVM models on the chapter subset data (summary data) using gensim's TextRank Generated Summary with 100 words .	148
8.13	Performance of best RF and SVM models on the chapter subset data (summary data) using sumy's LexRank Generated Summary	149

8.14	Performance of best RF and SVM models on the chapter subset data (summary data) using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	150
8.15	Performance of best RF and SVM models on the chapter subset data (summary data) using <code>sumy</code> 's Generated Summary using LSA	151
8.16	Performance of best RF and SVM models on the chapter subset data (summary data) using <code>sumy</code> 's Generated Summary using LSA with stopwords . .	152
8.17	Performance of various Machine Learning experimental setups	153
8.18	Performance of best Machine Learning algorithms at the chapter-level	154
8.19	Performance of LSTM built using pre-trained embeddings on the Full-text (half data)	158
8.20	Performance of LSTM built using pre-trained embeddings	159
8.21	Performance of best LSTM models on the Full-text (Half data)	160
8.22	Performance of best LSTM models on the Full-text (All data)	161
8.23	Performance of best LSTM models on the chapter subset data (full-text data)	162
8.24	Performance of best LSTM models on the chapter subset data (summary data) using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	163
8.25	Performance of best LSTM models on the chapter subset data (summary data) using <code>gensim</code> 's TextRank Generated Summary with 100 words	164
8.26	Performance of best LSTM models on the chapter subset data (summary data) using <code>sumy</code> 's LexRank Generated Summary	164
8.27	Performance of best LSTM models on the chapter subset data (summary data) using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	165

8.28	Performance of best LSTM models on the chapter subset data (summary data) using <code>sumy</code> 's Generated Summary using LSA	165
8.29	Performance of best LSTM models on the chapter subset data (summary data) using <code>sumy</code> 's Generated Summary using LSA with stopwords	166
8.30	Performance of various deep learning experimental setups	166
8.31	Performance of various deep learning experimental setups across different number of epochs	168
8.32	Performance of various deep learning architectures	170
8.33	Performance of best deep learning algorithms at the chapter-level	170
B.1	Performance of top 10 RF models with embeddings trained using the PSU + Illinois dataset on the Full-text (Half data)	191
B.2	Performance of top 10 SVM models with embeddings trained using the PSU + Illinois dataset on the Full-text (Half data)	191
B.3	Performance of top 10 RF models with embeddings trained using the VT dataset on the Full-text (Half data)	192
B.4	Performance of top 10 SVM models with embeddings trained using the VT dataset on the Full-text (Half data)	192
B.5	Performance of top 10 RF models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (Half data)	193
B.6	Performance of top 10 SVM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (Half data)	193

B.7	Performance of top 10 RF models with embeddings trained using the PSU + Illinois dataset on the Full-text (All data)	194
B.8	Performance of top 10 SVM models with embeddings trained using the PSU + Illinois dataset on the Full-text (All data)	195
B.9	Performance of top 10 RF models with embeddings trained using the VT dataset on the Full-text (All data)	195
B.10	Performance of top 10 SVM models with embeddings trained using the VT dataset on the Full-text (All data)	196
B.11	Performance of top 10 RF models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (All data)	196
B.12	Performance of top 10 SVM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (All data)	197
B.13	Performance of top 10 RF models with embeddings trained using the PSU + Illinois dataset on the Chapter subset data (full-text data)	198
B.14	Performance of top 10 SVM models with embeddings trained using the PSU + Illinois dataset on the Chapter subset data (full-text data)	199
B.15	Performance of top 10 RF models with embeddings trained using the VT dataset on the Chapter subset data (full-text data)	199
B.16	Performance of top 10 SVM models with embeddings trained using the VT dataset on the Chapter subset data (full-text data)	200
B.17	Performance of top 10 RF models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)	200

B.18 Performance of top 10 SVM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)	201
B.19 Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	202
B.20 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	202
B.21 Performance of top 5 RF models with embeddings trained using the VT dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2 . . .	203
B.22 Performance of top 5 SVM models with embeddings trained using the VT dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2 . . .	203
B.23 Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	204
B.24 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	204
B.25 Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	205
B.26 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	205

B.27 Performance of top 5 RF models with embeddings trained using the VT dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	206
B.28 Performance of top 5 SVM models with embeddings trained using the VT dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	206
B.29 Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	207
B.30 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	207
B.31 Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's LexRank Generated Summary	208
B.32 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's LexRank Generated Summary	208
B.33 Performance of top 5 RF models with embeddings trained using the VT dataset using <code>sumy</code> 's LexRank Generated Summary	209
B.34 Performance of top 5 SVM models with embeddings trained using the VT dataset using <code>sumy</code> 's LexRank Generated Summary	209
B.35 Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's LexRank Generated Summary	210
B.36 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's LexRank Generated Summary	210

B.37 Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	211
B.38 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	211
B.39 Performance of top 5 RF models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	212
B.40 Performance of top 5 SVM models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	212
B.41 Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	213
B.42 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	213
B.43 Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using LSA	214
B.44 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using LSA	214
B.45 Performance of top 5 RF models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using LSA	215
B.46 Performance of top 5 SVM models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using LSA	215
B.47 Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using LSA	216

B.48 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using LSA	216
B.49 Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using LSA with stopwords	217
B.50 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using LSA with stopwords	217
B.51 Performance of top 5 RF models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using LSA with stopwords	218
B.52 Performance of top 5 SVM models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using LSA with stopwords	218
B.53 Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using LSA with stopwords	219
B.54 Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using LSA with stopwords	219
B.55 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois dataset on the Full-text (Half data)	220
B.56 Performance of top 10 LSTM models with embeddings trained using the VT dataset on the Full-text (Half data)	221
B.57 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (Half data)	221

B.58 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois dataset on the Full-text (All data)	222
B.59 Performance of top 10 LSTM models with embeddings trained using the VT dataset on the Full-text (All data)	223
B.60 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (All data)	223
B.61 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)	224
B.62 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)	225
B.63 Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)	225
B.64 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	226
B.65 Performance of top 5 LSTM models with embeddings trained using the VT dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2 . . .	226
B.66 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2	227
B.67 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	228

B.68 Performance of top 5 LSTM models with embeddings trained using the VT dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	228
B.69 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using <code>gensim</code> 's TextRank Generated Summary with 100 words	229
B.70 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's LexRank Generated Summary	230
B.71 Performance of top 5 LSTM models with embeddings trained using the VT dataset using <code>sumy</code> 's LexRank Generated Summary	230
B.72 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's LexRank Generated Summary	231
B.73 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	232
B.74 Performance of top 5 LSTM models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	232
B.75 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using <code>sumy</code> 's Generated Summary using Luhn's Algorithm	233
B.76 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using <code>sumy</code> 's Generated Summary using LSA	234
B.77 Performance of top 5 LSTM models with embeddings trained using the VT dataset using <code>sumy</code> 's Generated Summary using LSA	234

B.78 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA	235
B.79 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using sumy's Generated Summary using LSA with stopwords	236
B.80 Performance of top 5 LSTM models with embeddings trained using the VT dataset using sumy's Generated Summary using LSA with stopwords	236
B.81 Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA with stopwords	237

List of Abbreviations

ETD Electronic Theses and Dissertations

GRU Gated Recurrent Units

LSTM Long Short-Term Memory

ML Machine Learning

PQDT ProQuest Dissertations and Theses

RF Random Forest

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Background

A great deal of progress has been made to leverage the advances made in natural language processing and machine learning to better extract and mine data from journals, conference proceedings, and other digital library documents. However, these techniques do not extend well to book-length documents. This research aims to identify methods that are catered to such documents, demonstrated with Electronic Theses and Dissertations (ETDs).

Theses and Dissertations contain a vast body of research information. Students graduating from universities have been increasingly required to submit Electronic Theses and Dissertations (ETDs). These works contain a detailed description of the research conducted by students during the course of their academic study. This collection covers a range of STEM as well as non-STEM disciplines. Additionally, the corpus contains highly domain specific vocabulary. All of these factors indicate the wealth of knowledge present in this corpus and hint at the benefits of increasing its accessibility.

By solely using author-supplied keywords, it is difficult to appropriately categorize an ETD. As explained by McCutcheon [36], ETD authors do not always assign adequate keywords to identify the categories to which an ETD may belong. Existing systems such as the ProQuest Subject Categories [53] or the Library of Congress Subject Headings [45]) contain terms

that are related to ETDs which could aid in their categorization. However, the manual cataloging of ETDs is expensive. Thus, we aim to build an automatic classification system that leverages the advancements made in the field of machine learning.

When ETDs are interdisciplinary in nature, the chapters contained within an ETD could have varying degrees of interdisciplinarity that differ from that of the full-text. Knowledge of these varied categories with which an ETD can be related is advantageous to researchers. We intend to perform multi-label classification at the chapter-level with a view to identifying all possible subject categories to which an ETD may belong. This research could be further extended to build a search interface on top of the ETD corpus, which would enable researchers to find similar works, at the chapter-level, that are most relevant to their study. For the purpose of this study, we utilize the ProQuest subject category system as our target classification system.

1.2 Problem Statement

At a very high-level, we have the following problem statement: Given a chapter of an ETD, along with metadata information about the ETD and the ProQuest subject category system, we are to build a classifier that can find a set of appropriate subject categories to be associated with the chapter.

For this research, our given corpus is a set of Electronic Theses and Dissertations. We leverage the metadata information provided by the authors and corpus managers along with the text of the ETD to identify the most appropriate set of subject categories (from the list of categories as described as part of the ProQuest subject category system) that can be associated with a given chapter of an ETD.

1.3 Motivation

As part of the larger goal of increasing the accessibility of book-length documents such as ETDs, multiple services such as classification, citation parsing, and summarization are useful to different stakeholders like researchers, librarians, and educators. The classification would aid in grouping similar ETDs together and better understanding the interdisciplinary nature of the collection.

Chapters often contain similar ideas clubbed together, which can aid a researcher in identifying similar research topics. The degree of interdisciplinarity of a chapter can vary from that of the full-text of the ETD. Having knowledge about this is useful in helping researchers to find prior research that is most similar to their current research efforts. As the size of this corpus of ETDs increases in size, it becomes increasingly more challenging to manually catalog ETDs. Thus, an automatic classification system capable of assigning labels to ETDs, both at the full-text as well as the chapter-level, will ease the process of cataloging this collection.

1.4 Hypotheses

Training a classifier to automatically categorize chapters of ETDs would require us to first segment the full-text of the ETD into chapters, extract the text from the PDF, transform the text into an appropriate vector representation, and then train a machine learning or deep learning model. The quality of the classifier can be measured using its F-1 score.

Considering the above context, the hypotheses of this work are:

- Using custom trained `fastText` word embeddings and `Doc2Vec` document embeddings

will result in better F-1 scores than methods using pre-trained embeddings trained using Common Crawl or Wikipedia datasets.

- Using extractive summaries of chapters when training the classifiers will increase the F-1 score of the classifiers as compared to the classifiers trained just using the full-text chapter data.
- Using deep learning methods such as LSTM can help increase the F-1 score as compared to machine learning models such as Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest (RF).
- Some of the subject categories assigned to chapters of an ETD will differ from the ones assigned to the full-text of the entire ETD.

1.5 Research Questions

The research questions that this thesis will attempt to answer are:

RQ1: Can the use of custom word and document embeddings improve the F-1 score of the classifiers as compared to pre-trained embeddings trained using Common Crawl or Wikipedia datasets?

RQ2: Can the use of extractive summaries generated from the chapters improve the F-1 score of the classifiers?

RQ3: Can deep learning techniques like Long Short Term Memory (LSTM) be used to better classify chapters of ETDs?

RQ4: Do some of the subject categories predicted for chapters of the ETD differ from the ETD full-text subject categories?

1.6 Research Contributions

This work makes the following contributions:

- This work provides custom trained word embeddings and document embeddings that have been trained on a large corpus of ETDs.
- It describes tools and techniques that can be used to segment ETDs into chapters and to extract their textual content.
- It describes an approach for the automatic classification of chapters of ETDs using the ProQuest subject category system.
- It shows that for some chapters, the set of categories assigned differs from the set of categories assigned to the full ETD.
- It describes a methodology to study the impact of summarization on the performance of chapter-level classifiers.
- For each of the above contributions, it demonstrates an improvement relative to baseline schemes.

1.7 Outline of the Thesis

- Chapter 1, above, outlines the problem statement, motivation, hypotheses, research questions, and research contributions.
- Chapter 2 discusses relevant background information required to better understand this work. Chapter 3 extends that with a review of relevant literature in the areas of segmentation, text extraction, summarization, and text classification.

- Chapter 4 presents a detailed description of the different sets of ETDs utilized for various tasks performed in this research.
- Chapter 5 presents the findings from the segmentation of full-text ETDs into chapters as well as the extraction of the text from the PDFs.
- Chapter 6 presents the findings from the summarization of chapters of ETDs.
- Chapter 7 elaborates on the design of the system. It also describes methods used for full-text classification employing machine learning.
- Chapter 8 presents the results and discussions of the different experiments conducted as part of this, on each of two datasets.
- Chapters 9 and 10 present the limitations of the system, possible future work, and conclusions of the research.

Chapter 2

Background

In this chapter, we discuss relevant background information needed to better understand the steps performed in our research.

2.1 Embeddings

Words need to be represented in the form of continuous vectors such that the machine learning models are capable of interpreting the words. Traditional approaches for these word representations include the bag of words approach [8] and one-hot encoding [9]. Both these approaches face issues with high dimensionality and the loss of context of words, which causes a loss in the semantic meaning of the text. Embeddings, on the other hand, are capable of capturing the context of the text. They are used to represent discrete categorical features as continuous vectors.

2.1.1 Word Embeddings

Word embeddings are dense, low-dimensional distributed representations of words. In this section, we discuss three commonly used word embeddings.

Word2Vec

Word2Vec was the first word embedding model. It was proposed in the seminal paper [38] published in 2013. The model architecture comprises of small neural networks that calculate the word embeddings based on two approaches. The first is the continuous bag of words or CBOW wherein the model attempts to predict the most likely word in the given context, while the second is skip-gram wherein the model uses the target word to predict its context.

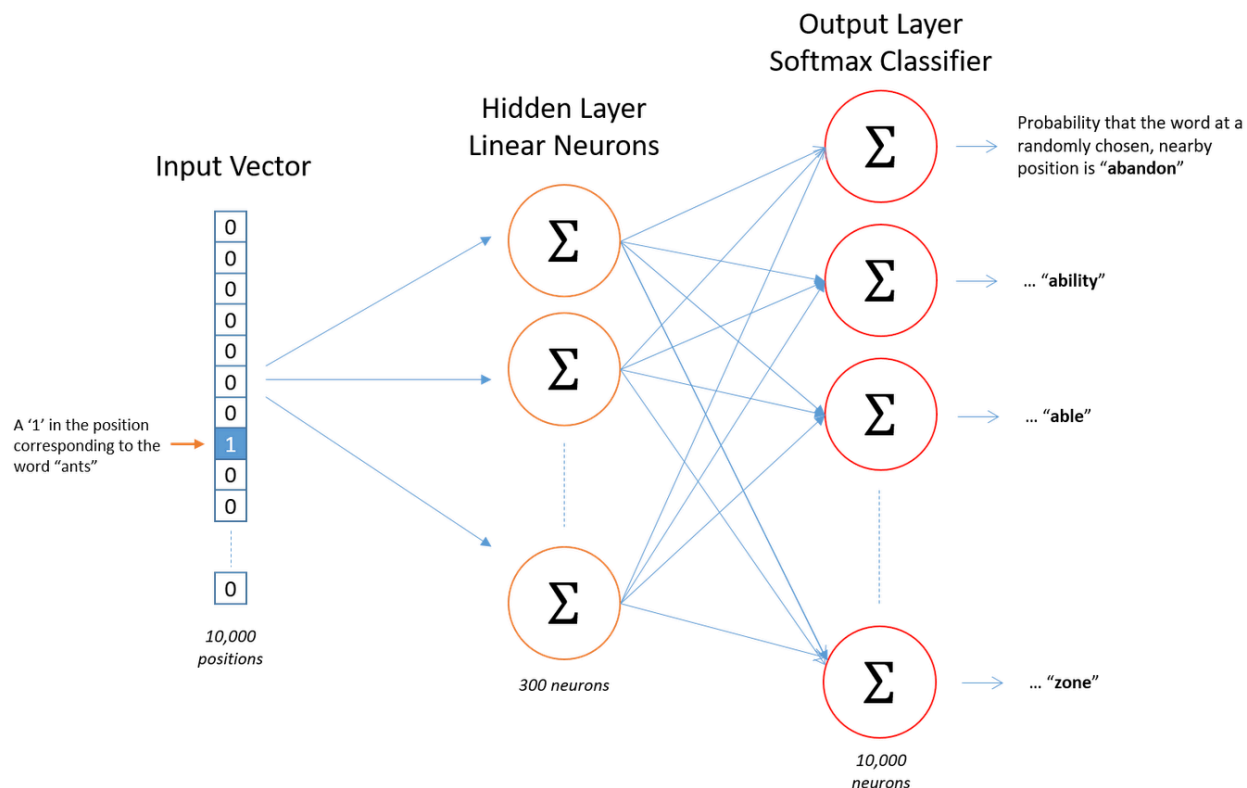


Figure 2.1: Neural Network architecture for Word2Vec [35]

Word2Vec, however, does not leverage the additional global statistical information given by the frequent co-occurrence of words; it merely uses it as more training examples. It also suffers from issues related to out-of-vocabulary (OOV) words.

GloVe

GloVe [50] word embeddings were created in 2014 and attempted to improve over the existing Word2Vec embeddings. It was built on two main methods, namely, global matrix factorization and local context window. Global matrix factorization uses linear algebra to reduce the large term frequency matrices that represent the presence or absence of words in a document [55]. The local context window comprises of continuous bag of words (CBOW) and skip-gram.

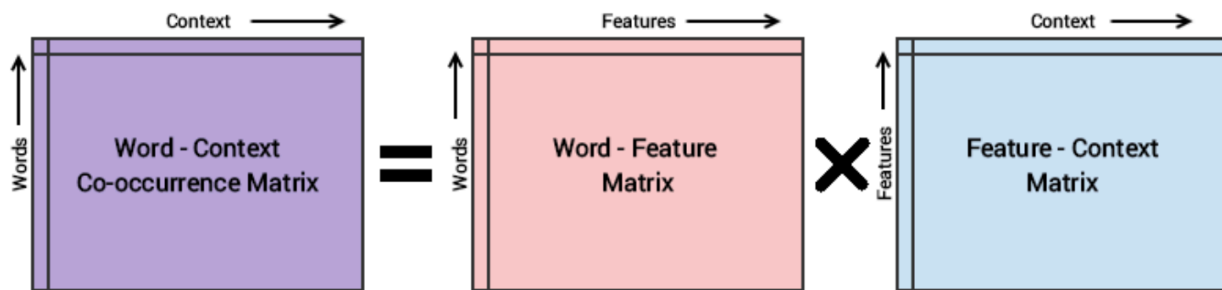


Figure 2.2: Conceptual model for GloVe [58]

When building the GloVe model, instead of simply predicting neighboring words (in the CBOW case) or the focus word (in the skip-gram case), the word embeddings are optimized such that the dot product of the vector generated for two words is equal to the logarithmic value of the number of times the words co-occur. Thus, these embeddings can be viewed as a summary of the entire corpus that can reflect the co-occurrences between words [10].

Similar to Word2Vec, GloVe embeddings also suffer from an inability to generalize well to out-of-vocabulary (OOV) words.

fastText

The scheme for fastText word embeddings [24] [7] attempts to overcome the OOV issue that plagues both GloVe and Word2Vec. To achieve this, it makes use of sub-word information. Here, the model considers words as well as characters. The embeddings generated are similar to Word2Vec; however, they are not directly calculated. They consist of a combination of lower-level embeddings.

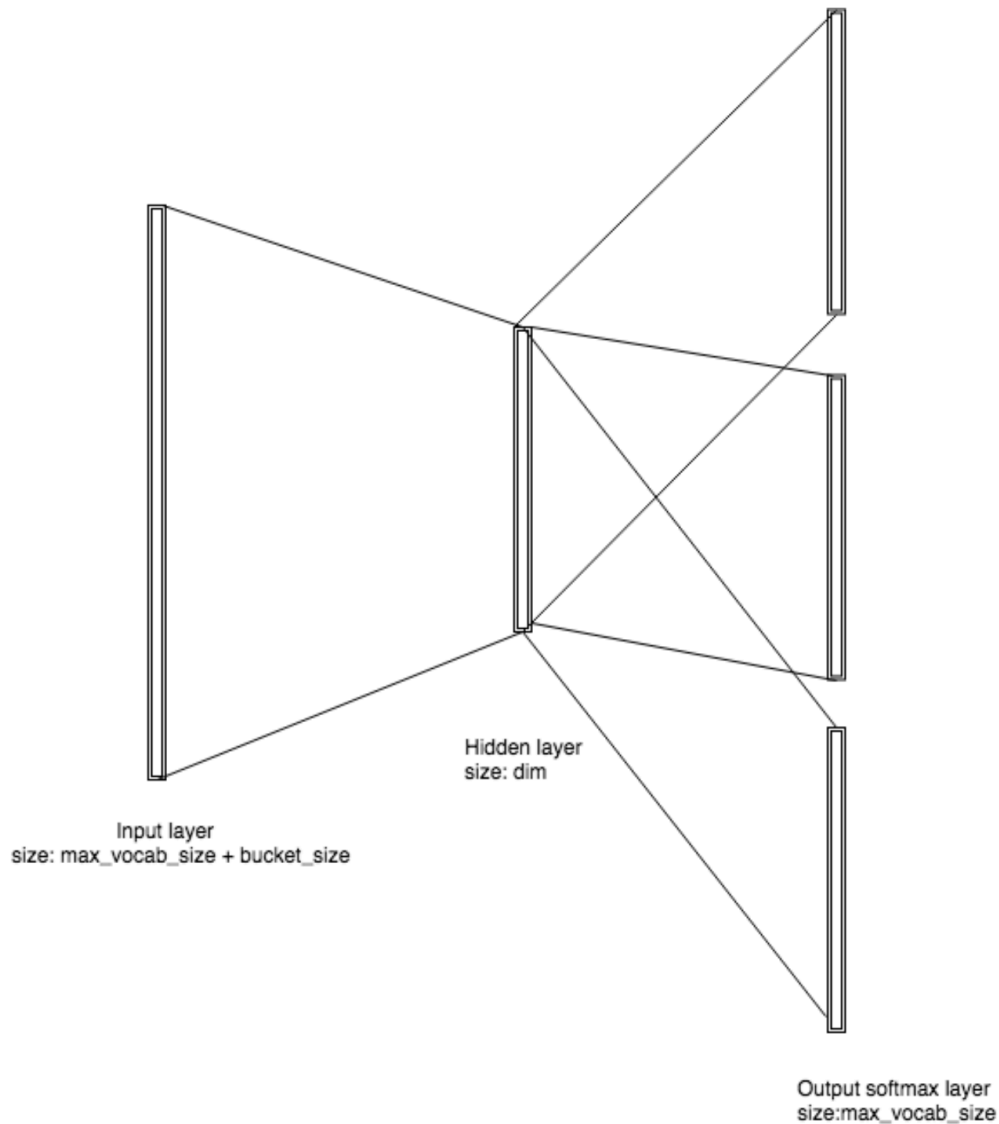


Figure 2.3: fastText skip gram model topography [64]

Due to this, the amount of training data required by this model is less than that of the original Word2Vec model, since more information can be extracted from each part of the text data. In addition to this, it is able to overcome the OOV word issue provided that the new words introduced contain the same set of characters as the original corpus on which the model was trained.

2.1.2 Document Embeddings using Doc2Vec

Documents do not have the same logical structure as words, so it is not possible to directly utilize word embeddings on long document data. Traditional approaches like the bag-of-words do not maintain the order of words in a sentence and lose important semantic information. Thus, the authors of [26] propose an algorithm that is capable of generating fixed-length representations from variable length documents. Commonly known as Doc2Vec, this model draws inspiration from the Word2Vec model.

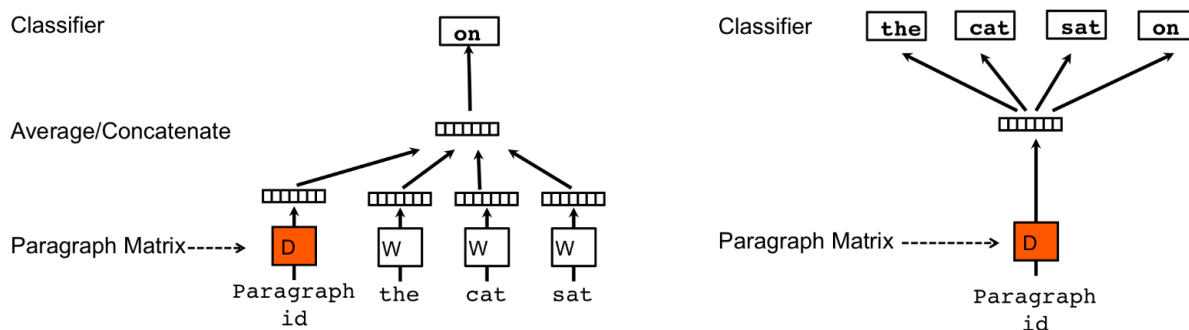


Figure 2.4: PV-DM and PV-DBOW variants of Doc2Vec [26]

As indicated by Figure 2.4, there are two variants of Doc2Vec – Distributed Memory version of Paragraph Vector (PV-DM), and Distributed Bag of Words version of Paragraph Vector (PV-DBOW). In both cases, the model includes an additional feature vector ‘Paragraph ID’ that is unique to each document. New document vectors can be ‘inferred’ by using this trained model.

In the case of the PV-DM model, the word vectors W represent the concept of a word whereas document vectors D intend to represent the concept of the document [60]. The PV-DBOW model is similar to the skip-gram model of Word2Vec [26]. This model is trained faster as it is not required to store the word vectors. The authors experimentally found that the PV-DM model performed consistently better than the PV-DBOW model. For example, they found that the PV-DM model was able to achieve an error rate of only 7.63% on the IMDB dataset.

2.2 Multi-label Classification

The chapter-level classification that we perform in this work is multi-label in nature since each chapter could be associated with multiple labels. In the case of classifying textual documents, a single document could belong to many categories at once. Multi-label classification of a sample can be formulated as the prediction of properties that are not mutually exclusive. The classification problem that we attempt to solve as part of this research extends beyond multi-class classification wherein a single document can only belong to a single class-label since an ETD can belong to multiple ProQuest subject categories simultaneously. Most of the traditional machine learning and deep learning algorithms have been built for single-label (both single-class as well as multi-class) classification problems. Thus, some of the techniques used to solve multi-label classification problems involve converting this multi-label problem into multiple single-label problems and then utilizing the existing single-label algorithms. Some of these techniques have been mentioned in this section.

2.2.1 One vs. Rest

An intuitive technique that can be applied to solve a multi-label classification problem is to divide it into multiple independent binary classification problems (one classification problem for each of the individual categories). In this strategy, multiple independent binary classifiers are built, and when an unseen test instance is to be classified, several labels that have the highest confidence can be picked as the class labels. This approach does not consider underlying correlations between the class labels. Additionally, the implementation provided by scikit-learn [13] does not specify the method used to break ties in case of multiple labels that have the highest confidence value.

2.2.2 Binary Relevance

The Binary Relevance approach is one of the techniques that is used to convert a multi-label classification problem into multiple single-label classification problems. In this approach, an ensemble of single-label binary classifiers is trained where each classifier represents a single class. Each classifier predicts the membership or non-membership of a class, and the union of all these classes is taken as the multi-label output. This technique, however, does not consider possible correlations that may exist among the labels [42].

2.2.3 Classifier Chains

The Classifier Chains approach builds a chain of binary classifiers (one for each of the class labels) C_0, C_1, \dots, C_n . A classifier C_i will utilize the predictions of all prior classifiers C_j (where $j < i$). This method is capable of taking into account possible correlations among the class labels. The total number of classifiers that need to be built for this technique is equal

to the total number of classes that are present [42].

2.2.4 Label Powerset

The Label Powerset approach aims to take into consideration possible correlations between class labels. It considers each member of the power set of labels in the training set as a single label. However, this approach has high computational complexity and can cause a deterioration in performance when the number of classes increases [42].

2.2.5 Adaptive Algorithm

This technique involves adapting single-label classification algorithms to multiple labels by making changes to the cost/decision functions. For example, in case of the K-nearest neighbor (KNN) algorithm, a multi-label lazy learning approach (ML-KNN) that is derived from the traditional single-label algorithm can be utilized [42].

2.3 Machine Learning Classifiers

This section describes commonly used Machine Learning Algorithms. Here, we consider three common machine learning algorithms that are used in text classification, namely Logistic Regression, Support Vector Machines, and Random Forest.

2.3.1 Logistic Regression

Logistic Regression is a statistical classification algorithm that is typically used when the class label is categorical. This algorithm uses a sigmoid function to convert the model's

prediction into a value between 0 and 1 [14]. It is most often used in binary classification problems but can be extended to other forms of classification as well.

2.3.2 Support Vector Machine

Support Vector Machine is a large-margin classifier that aims to create a hyperplane such that it distinctly classifies each of the data points. Support Vector Machine classifiers are very effective in high dimensional spaces. Since it utilizes a subset of the training data points in the decision function, it is known to be memory efficient.

2.3.3 Random Forest

Random Forest is an ensemble machine learning algorithm that fits a number of weak-learner decision tree classifiers. It can be described as a meta-estimator that fits multiple decision tree classifiers on different smaller sub-sets of the dataset. It controls over-fitting by averaging. This also helps to improve the predictive accuracy of the classifier.

2.4 Deep Learning Architectures

Here we discuss two commonly used deep learning architectures for text classification, namely *Long Short-Term Memory (LSTM)* and *Gated Recurrent Units (GRU)*.

Traditional Recurrent Neural Networks (RNNs) [57] face issues with longer sequence length documents. For longer sequences, there is an issue faced by the RNN to carry information from an earlier time step to a later one. Additionally, during backpropagation, RNNs suffer from the problem of vanishing gradients. Figure 2.5 illustrates the architecture of a

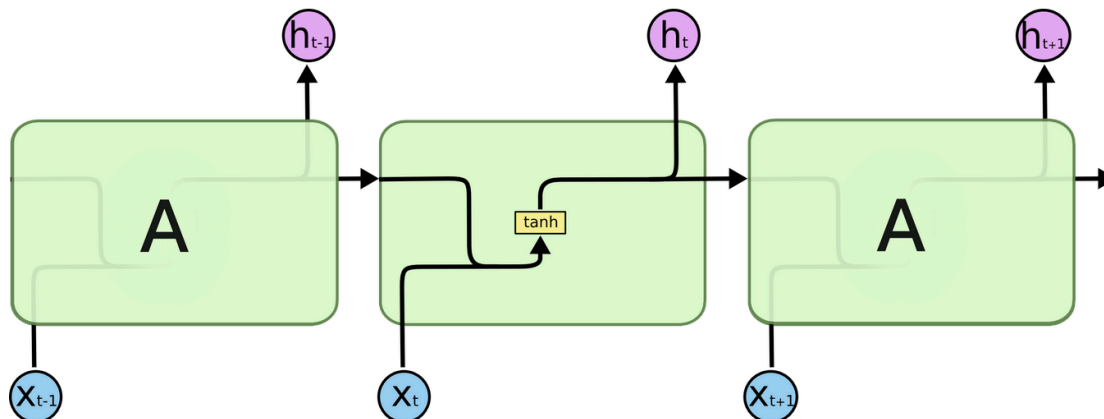


Figure 2.5: Illustration of a traditional RNN [46]

traditional RNN cell.

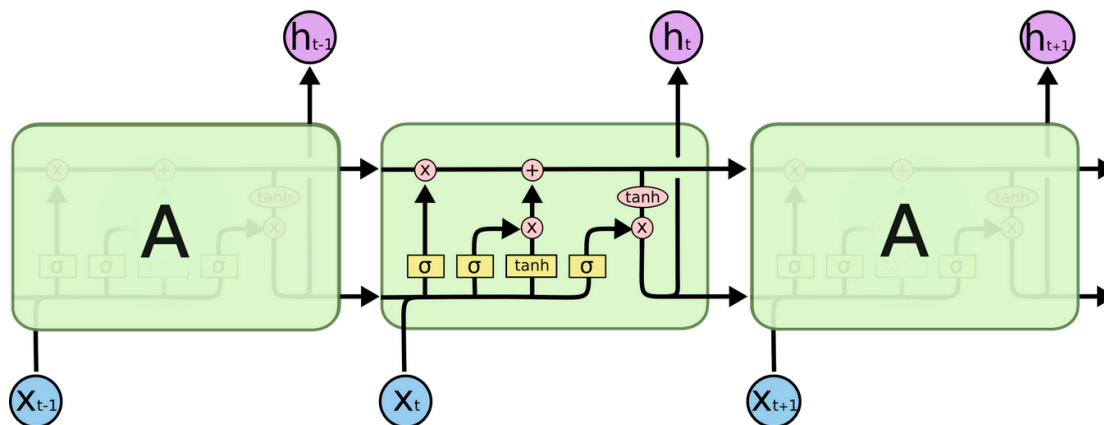


Figure 2.6: Illustration of an LSTM [46]

LSTMs [21] and GRUs [12] were introduced in an attempt to overcome these limitations of traditional RNNs. Figures 2.6 and 2.7 illustrate the architecture of LSTM and GRU cells, respectively. Each of these utilized different ‘gates’ that enable them to better determine important parts of the input sequence.

The LSTM consists of a cell state and three gates, namely the input gate, forget gate, and output gate. The cell state acts as the ‘memory’ of the network. The gates enable the network to learn relevant information that is to be kept or forgotten during training. [41]

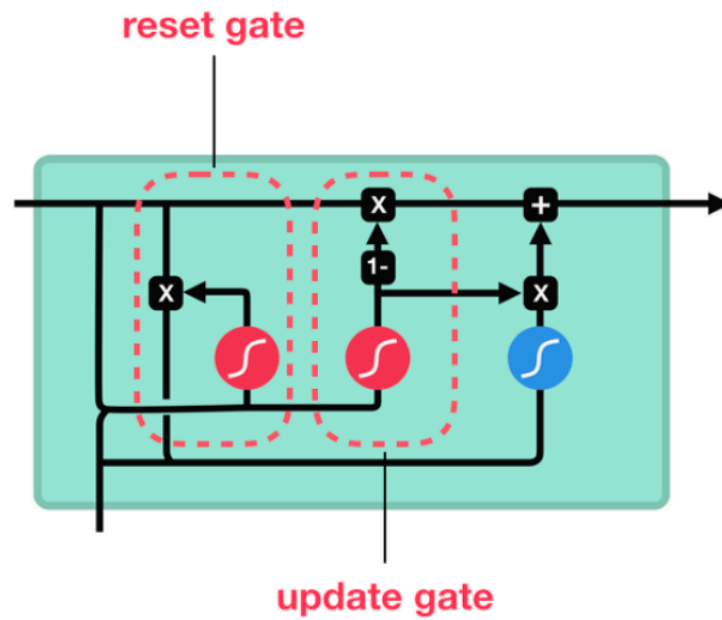


Figure 2.7: Illustration of a GRU [41]

The GRU, on the other hand, does not have a cell state and it uses the hidden state in order to transform information. There are only two gates: the reset gate and the update gate. Since GRUs have fewer operations, they generally train faster than LSTMs.

Chapter 3

Review of Literature

In this chapter, we discuss the tools, relevant algorithms used by the tools, and related research in the fields of classification of long documents and digital libraries.

3.1 Chapter Segmentation

To perform chapter-level classification, we needed to segment a long PDF document into chapters. In order to perform this task, we evaluated a number of tools. Some of these tools helped with the segmentation task whereas some tools were developed to extract text from PDFs. Here, we first talk about the tools that can be used to extract data from the PDF documents (either in the form of text files, DOCX files, or hierarchical formats such as HTML/XML formats). We then talk about the tools that can be utilized in order to segment the PDF document into sections.

3.1.1 Text Extraction from PDFs

In this section, we list various tools such as PDFMiner [59], PyPDF2 [51], PyMuPDF [29], pdftotext [48], ABBYY Cloud OCR SDK [1], textract [32], and Tika-Python [33] that can be used to extract text from PDFs. Some of these tools are Python bindings over software written in other languages whereas some are written in Python. While the tools enlisted

here can be used to get the text content from PDFs, they do not maintain the structure of the document and hence are not suitable to be used for segmentation into chapters.

PDFMiner

PDFMiner is a Python-based tool that can be used to extract text from PDFs. It allows the user to extract the location of text on a page, font, or line information. It also enables the user to convert a PDF file into other file formats such as text (which is the default option), HTML, XML, or Tagged PDF. It includes two command-line tools ‘pdf2txt.py’ and ‘dumppdf.py’. ‘pdf2txt.py’ can be used to extract the text contents from a PDF file and save it into a pre-defined output format. ‘dumppdf.py’ enables a user to dump the contents of a PDF file into a pseudo-XML format and is generally used for debugging purposes.

PyPDF2

PyPDF2 is a Pure-Python library that was built as a PDF toolkit. It has the ability to extract information from a document (such as the title, author, etc.), split the PDF page by page, merge documents page by page, crop pages, merge multiple pages into a single page, as well as encrypt and decrypt PDF files. Since this is a Pure-Python library, it can be run on any Python platform and does not require any external dependencies. It allows for PDF manipulation to be done in memory as it can work on StringIO objects instead of file streams.

PyMuPDF

PyMuPDF is a Python binding for MuPDF. It has the ability to access content from files with multiple extensions such as ‘.pdf’, ‘.xps’, ‘.oxps’, ‘.cbz’, ‘.fb2’, or ‘.epub’. It is capable of de-

crypting a PDF document, accessing metadata information including the meta-information, links, and bookmarks. It can also render the pages of a PDF into raster formats such as PNG or vector format such as SVG. PyMuPDF can also convert the PDF into other formats such as XHTML, XML, JSON, and text.

Pdftotext

Pdftotext is an open-source utility that can be used to convert PDF files to plain text files by extracting text data from the PDF encapsulated file. It also has the ability to extract a single page by utilizing its page number. It can read text data from password protected PDFs.

ABBYY Cloud OCR SDK

ABBYY Cloud OCR SDK is a web-based OCR service that can be used to extract text from PDFs and perform PDF conversion tasks. It can perform text recognition of printed text for over 200 languages as well as convert documents to searchable PDF, PDF/A, Microsoft Word, Excel, and PowerPoint. By leveraging ABBYY's Advanced Document Recognition Technology (ADRT), we can recreate the structure and layout of the original document and convert the PDF version of the document back into its original form (in case of documents created using Microsoft Office).

Textextract

Textextract provides a single interface that can be used to extract content from various different file types without any irrelevant markup. This tool can be used in two ways – a command-line interface or the `textextract` Python package. Some of the file types supported for text

extraction include CSV, DOC, DOCX, HTML, JSON, and PDF.

Tika-Python

Tika-Python is the Python binding to Apache Tika REST services. Apache Tika is a content analysis toolkit written in Java developed by the Apache Software Foundation. It can be used to detect and extract both metadata and text from multiple file types including PPT, XLS, and PDF. Tika is very useful for content analysis and translation; it also can be leveraged for search engine indexing. It has the ability to get text from images by using Tesseract (which is an OCR software package). The Tika-Python tool enables developers to easily utilize all of the functionalities offered by Apache Tika and integrate it with Python applications.

3.1.2 Segmenting PDFs into Sections

The tools enlisted in this section – Grobid and ITCore (Intelligent Textbooks Core) – enable us to segment a PDF into smaller sections.

Grobid

Grobid means GeneRation Of BIBliographic Data. It is a machine learning library that can be used to extract, parse, and re-structure PDFs into XML/TEI documents [30]. It works well with scientific publications. It can be used to extract metadata information such as title, abstract, authors, affiliations, keywords, references, etc. It also has the ability to perform full-text extraction from PDF articles and segment the document (by using different tags in the XML generated to demarcate the various segments).

Grobid can be used via a RESTful API, Java API, or Docker container. It also includes

batch processing which can be leveraged when a large number of documents need to be processed.

ITCore (Intelligent Textbooks Core)

ITCore [5] is a Java based tool that can be used to process book-length PDFs and produce a TEI model of a text book with structural information of the PDF including sections and sub-sections. It also provides an option to generate PDF files of the segments identified. It utilizes a rule-based algorithm that leverages the Table of Contents of books to identify the sections and sub-sections of the full-text PDF.

3.2 Summarization

Text Summarization is a very useful Natural Language Processing technique that enables us to convert a large document text into a few sentences without losing the important content of the text. Summarization can be of two types, extractive and abstractive. With extractive summarization, the algorithm attempts to identify the most informative sentences from the entire text to form the summary. With abstractive summarization, the algorithm attempts to generate concise phrases or sentences that are semantically similar to the original text. This type of summarization is well suited for tasks that require these summaries to be viewed by humans, as it works in a manner that is similar to the method used by humans to generate summaries. Thus, it attempts to paraphrase the original text [66].

For the purpose of this research, we intend to use summarization to shorten the full-text of chapters and feed the generated summaries into the machine learning model. Thus, for our purposes, extractive summarization techniques are well suited for the task at hand.

3.2.1 Extractive Summarization

Extractive summaries are generated by identifying words, phrases, sentences, and/or passages from the original full-text to generate the summary. Four common extractive summarization techniques are discussed in this section.

TextRank

TextRank [37] is an unsupervised weighted-graph based algorithm that was built on top of the PageRank algorithm [47]. The TextRank algorithm first pre-processes the input text by removing all stop words, and then stemming the remaining words. It next generates a graph with sentences as the vertices. It then connects each sentence to every other sentence by an edge that represents the similarity of each pair of sentences. It runs the PageRank algorithm on this graph. The sentences with the highest PageRank score are then added as part of the final output summary.

LexRank

Similar to TextRank, is another graph based approach called LexRank [16]. In this approach, the algorithm uses an IDF-modified cosine score as the similarity measure determining the weight of the edge between the sentences. It also includes a post-processing step which ensures that the top sentences selected for the final summary are not too similar to one another.

Luhn’s Algorithm Based Summarizer

This algorithm [31] attempts to rank sentences by taking into account the significance of words that frequently occur in a document. It also takes into account the distance between these significant words due to the other non-significant ones.

Latent Semantic Analysis (LSA) Based Summarizer

Latent Semantic Analysis (LSA) [72] is a Natural Language Processing technique that attempts to project data into a lower dimensional space. It analyses the relationship of documents and the terms within the documents. The spatial decomposition can be interpreted as singular vectors that are capable of capturing and representing word combination patterns within the text corpus. The importance of a specific pattern in such a document is indicated by the magnitude of the singular vector. This summarization technique leverages the LSA technique while generating the summaries [65].

3.3 Classifiers

The chapter-level classification that we perform in this work is multi-label in nature since each chapter could be associated with multiple labels. In addition to this, as part of this research, we aim to compare the performance of machine learning classifiers and deep learning classifiers.

Prior work has been done to classify ETDs into the Library of Congress Classification (LCC) system [63] [62] [17] [56]. Ideas explored in the aforementioned works were used to formulate some aspects of this thesis.

3.3.1 Machine Learning Classifiers

In this section, we discuss prior work that has been performed with machine learning classifiers related to scientific documents.

ACM Digital Library Classifiers

As reported in his 2017 doctoral dissertation, Yinlin Chen [11] built multiple classifiers using the 2012 ACM Computing Classification System as the target classification system. He trained his classifiers on the ACM Digital Library metadata dataset. These are described in detail in the following sections.

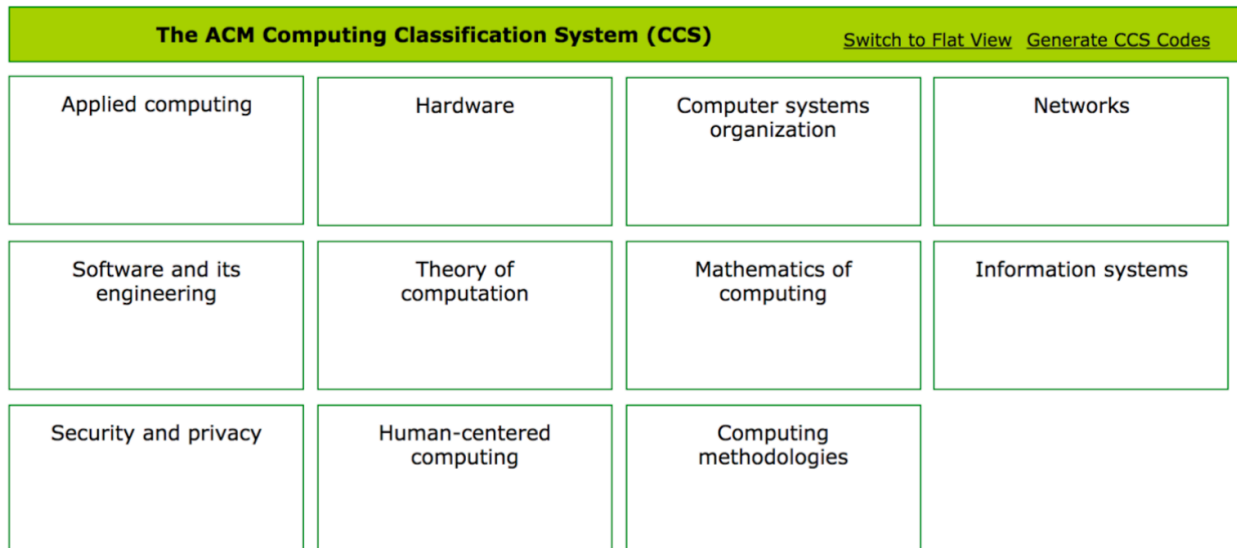


Figure 3.1: Eleven CCS top level branches used to build the ACM Digital Library Classifiers [11]

2012 ACM Computing Classification System

The 2012 ACM Computing Classification System (CCS) is a standard classification system in the computing field. It is represented as a six-level poly-hierarchical structure. The

topmost level consists of 13 branches; each of these branches has at most six children. Out of the 13 branches present in the original system, Chen selected the 11 shown in Figure 3.1. The other two branches, *General and reference* and *Social and professional topics*, contained cross-cutting computing concepts and were therefore excluded from Chen’s use of the system.

CCS CONCEPTS

• Information systems~Digital libraries archives • Human-centered computing~Empirical studies in HCI • Human-centered computing~Empirical studies in interaction design

Figure 3.2: An example of how CCS terms are displayed in a published work [2]

Authors submitting to ACM are asked to use any relevant first- and second-level nodes of the classification scheme, then identify the lowest branches of the tree that apply to their particular paper. Authors must then give a weighted score for each of the CCS terms they have chosen. A score of 500 indicates high relevance, 300 medium relevance, and 100 low relevance. These weights affect the order in which the terms are listed in the published work. Figure 3.2 shows an example of how the CCS section looks in an ACM published paper.

ACM Digital Library metadata dataset

Chen used an ACM Digital Library metadata dataset as training data. The dataset contains 1,761,956 journal and conference proceeding metadata records in XML format. Each of these metadata records contains a title, abstract, CCS, general term, author, and reference of the article. They do not include the full text of the article.

Methodology

A subset dataset for each ACM node was prepared for model training. Data cleaning was performed on the entire dataset. Under-sampling was used to deal with the imbalanced dataset. The bag-of-words model was used to represent the ACM dataset, and term frequency-inverse

document frequency (TF-IDF) was used as the weighting approach. Chi-square statistics were used for feature selection.

Multiple classifiers including Ridge regression [22], Perceptron [71], k-nearest neighbors (KNN) [18], Random Forest [28], SVM [19, 23], multinomial Naive Bayes [25], and BernoulliNB [34] were trained since these classifiers can handle sparse matrices.

Table 3.1: Top classifiers for the ACM dataset [11]

Algorithms	ACM CCS branch
Linear Support Vector Classification using (L1 or L2) penalty	Mathematics of computing Information systems Human-centered computing Applied computing Computer systems organization
Linear SVM with SGD training using (L1, L2 or Elastic-Net) penalty	Information systems Security and privacy Networks Theory of computation Applied computing Hardware Computing methodologies Software and its engineering

Based on the experiments conducted, it was found that Linear Support Vector Classification and Linear SVM with SGD training outperformed the other classification algorithms listed above. Table 3.1 gives an overview of which of these two classifiers worked better for the top-level ACM CCS branches.

CiteSeerX

CiteSeerX [75] is an online public digital library and search engine used for scientific and academic papers. It consists primarily of papers in the fields of computer and information science.

Classification using WoS Subject Categories

In [74], Wu et al. utilize a Web of Science (WoS) dataset which contains abstracts and titles of nearly 25 million papers. Web of Science has a Subject Category (SC) scheme consisting of 252 subject categories which cover a range of disciplines including science, social sciences, arts, and humanities. In this effort, the authors focus on classifying into Physics (PHYS), Chemistry (CHEM), Biology (BIO), Materials Science (MATSC), Computer Science (CMPSC), and Others. Four machine learning classifiers were trained, namely Support Vector Machine (SVM), Logistic Regression (LR), Multinomial Naive Bayes (MNB), and Random Forest (RF). The authors compared the performance of these ML models against a Multilayer Perceptron (MLP) with 3 hidden layers. The first layer had 1,024 neurons; the second and third layers had 512 neurons. Each layer used ReLU as the activation function. The input layer consisted of 5,000 neurons and the output layer consisted of 6 neurons (with softmax activation); each of these output neurons corresponded to the subject categories selected. Their results are summarized in Table 3.2.

Table 3.2: Multilayer Perceptron performance compared to other ML classifiers [74].

Metric	LR	RF	MNB	SVM	MLP
micro-F1	0.83	0.83	0.78	0.82	0.83
T_{test} (sec)	4.78	8.67	5.40	6.74	6.16

The authors considered using GloVe word embeddings [50] to represent the data. However, they found that the best F1-score obtained using GloVe was less than 0.80. The explanation offered by the authors was that only 37% of the vocabulary overlapped with the WoS abstracts. Therefore, the authors employed the Word2Vec [39] Skip-Gram model with softmax, along with TF-IDF [61] features. Without the TF-IDF features, the F1-score obtained was lower, potentially due to the equal importance given to each word by the Word2Vec embeddings.

3.3.2 Deep Learning Classifiers

This section describes prior work that has been done with long document classification and deep learning.

Hierarchical Transformers for Long Document Classification

The authors [49] extend the fine-tuning process of Bidirectional Encoder Representations from Transformers (BERT) [15] to address its applicability to longer input texts. The authors segment the original input text into smaller chunks of data and feed these inputs to a base model. Each of the outputs obtained are then passed on through a single recurrent/-transformer layer. This is then followed by a softmax activation layer. The final classification output is obtained after all the input segments have been consumed by the model.

The authors propose two techniques: Recurrence over BERT (RoBERT) and Transformer over BERT (ToBERT). The authors refer to these models as ‘Hierarchical Transformers’ as they introduce a hierarchy of representation.

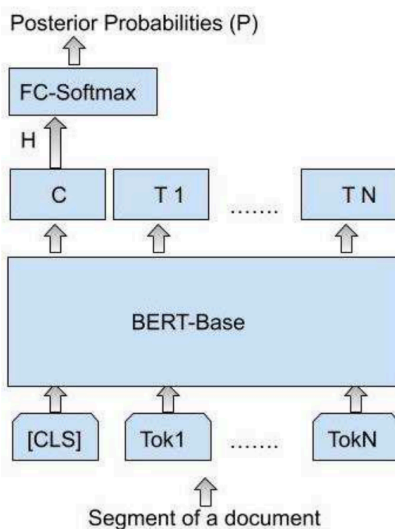


Figure 3.3: Hierarchical Transform BERT model [49]

Figure 3.3 illustrates the model proposed. Here, ‘H’ represents the segment representation obtained from the last block of the transformer. ‘P’ denotes the posterior probabilities of the segment.

dataset	Model	Accuracy
CSAT	MS-CNN	79.53
	ToBERT	83.48
	RoBERT	83.65
20 newsgroups	SCDV	84.6
	MS-CNN	86.12
	ToBERT	85.52
	RoBERT	84.71
Fisher	SVM MCE	91.9
	MS-CNN	92.93
	ToBERT	95.48
	RoBERT	91.18

Figure 3.4: Results for the Hierarchical Transform BERT model [49]

The authors evaluated the performance on three datasets, namely CSAT, 20 newsgroups, and Fisher. Each of these datasets contained an average of 787, 266, and 1788 words per document, respectively. Table 3.4 represents the results obtained for this model as compared to other baseline models.

Rethinking Complex Neural Network Architectures for Document Classification

The authors [4] examine the need for larger more complex neural networks in the case of long document classification. They found that a simple BiLSTM architecture that was trained with the correct hyperparameters could yield comparable F_1 and accuracy scores. Additionally, this model did not make use of attention mechanisms.

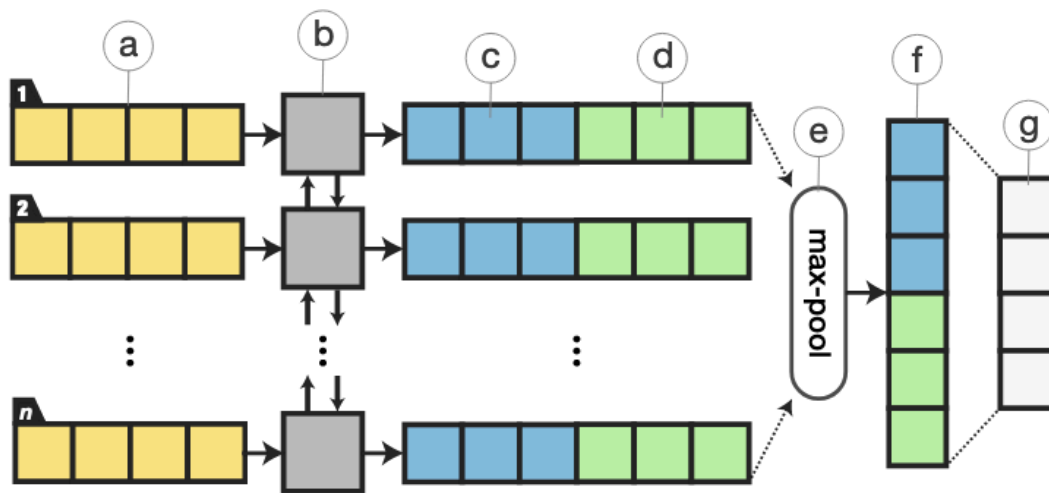


Figure 3.5: Model architecture of proposed model $LSTM_{reg}$: (a) input word embeddings, (b) BiLSTM, (c, d) concatenated forward $h^f_{1:n}$ and backward $h^b_{1:n}$ hidden features, (e) max-pooling overtime, (f) document feature vector, (g) softmax or sigmoid output [4]

The design of the model architecture, $LSTM_{reg}$, is represented in Figure 3.5, which was meant to be minimalistic. The authors evaluated their model using the Reuters, AAPD, IMDB, and Yelp 2014 datasets. These datasets had an average of 144.3, 167.3, 393.8, and 148.8 words per document, respectively. Figure 3.6 illustrates the results obtained for the $LSTM_{reg}$ model compared to other models.

# Model	Reuters		AAPD		IMDB		Yelp '14	
	Val. F ₁	Test F ₁	Val. F ₁	Test F ₁	Val. Acc.	Test Acc.	Val. Acc.	Test Acc.
1 LR	77.0	74.8	67.1	64.9	43.1	43.4	61.1	60.9
2 SVM	89.1	86.1	71.1	69.1	42.5	42.4	59.7	59.6
3 KimCNN Repl.	83.5 ±0.4	80.8 ±0.3	54.5 ±1.4	51.4 ±1.3	42.9 ±0.3	42.7 ±0.4	66.5 ±0.1	66.1 ±0.6
4 KimCNN Orig.	–	–	–	–	–	37.6 ^{††}	–	61.0 ^{††}
5 XML-CNN Repl.	88.8 ±0.5	86.2 ±0.3	70.2 ±0.7	68.7 ±0.4	–	–	–	–
6 HAN Repl.	87.6 ±0.5	85.2 ±0.6	70.2 ±0.2	68.0 ±0.6	51.8 ±0.3	51.2 ±0.3	68.2 ±0.1	67.9 ±0.1
7 HAN Orig.	–	–	–	–	–	49.4 [‡]	–	70.5[‡]
8 SGM Orig.	82.5 ±0.4	78.8 ±0.9	–	71.0[†]	–	–	–	–
9 $LSTM_{base}$	87.6 ±0.2	84.9 ±0.3	72.1 ±0.4	69.6 ±0.4	52.5 ±0.2	52.1 ±0.3	68.6 ±0.1	68.4 ±0.1
10 $LSTM_{reg}$	89.1 ±0.8	87.0 ±0.5	73.1 ±0.4	70.5 ±0.5	53.4 ±0.2	52.8 ±0.3	69.0 ±0.1	68.7 ±0.1

Figure 3.6: Comparison of $LSTM_{reg}$ results with other models [4]

DocBERT: BERT for Document Classification

The BERT language model is a large model that contains hundreds of millions of parameters, making it have a heavy computational expense. The authors [3] attempt to address this computational expense that is incurred while inferring from the BERT model. Here, they perform knowledge distillation [20] from the BERT_{large} model to smaller bidirectional LSTMs. In this work, the authors mention how the syntactic structure of the input text is not as important for document classification as compared to tasks such as paraphrasing or language inference, which are known to be more typical BERT tasks. The authors were able to reduce the number of parameters and train a BiLSTM with 30x fewer parameters.

#	Model	Reuters		AAPD		IMDB		Yelp '14	
		Val. F ₁	Test F ₁	Val. F ₁	Test F ₁	Val. Acc.	Test Acc.	Val. Acc.	Test Acc.
1	LR	77.0	74.8	67.1	64.9	43.1	43.4	61.1	60.9
2	SVM	89.1	86.1	71.1	69.1	42.5	42.4	59.7	59.6
3	KimCNN Repl.	83.5 ±0.4	80.8 ±0.3	54.5 ±1.4	51.4 ±1.3	42.9 ±0.3	42.7 ±0.4	66.5 ±0.1	66.1 ±0.6
4	KimCNN Orig.	–	–	–	–	–	37.6 ⁸	–	61.0 ⁸
5	XML-CNN Repl.	88.8 ±0.5	86.2 ±0.3	70.2 ±0.7	68.7 ±0.4	–	–	–	–
6	HAN Repl.	87.6 ±0.5	85.2 ±0.6	70.2 ±0.2	68.0 ±0.6	51.8 ±0.3	51.2 ±0.3	68.2 ±0.1	67.9 ±0.1
7	HAN Orig.	–	–	–	–	–	49.4 ³	–	70.5 ³
8	SGM Orig.	82.5 ±0.4	78.8 ±0.9	–	71.0 ²	–	–	–	–
9	LSTM _{reg}	89.1 ±0.8	87.0 ±0.5	73.1 ±0.4	70.5 ±0.5	53.4 ±0.2	52.8 ±0.3	69.0 ±0.1	68.7 ±0.1
10	BERT _{base}	90.5	89.0	75.3	73.4	54.4	54.2	72.1	72.0
11	BERT _{large}	92.3	90.7	76.6	75.2	56.0	55.6	72.6	72.5
12	KD-LSTM _{reg}	91.0 ±0.2	88.9 ±0.2	75.4 ±0.2	72.9 ±0.3	54.5 ±0.1	53.7 ±0.3	69.7 ±0.1	69.4 ±0.1

Figure 3.7: Comparison of KD-LSTM_{reg} results with other models [3]

Following along the work done in [15], the authors introduce a fully-connected layer over the state corresponding to the ‘CLS’ token. In the fine-tuning process, they add additional softmax classifier parameters. The knowledge is distilled into the smaller LSTM_{reg} presented in [4]. For the objective function of the distillation, the authors minimize the Kullback-Leibler (KL) divergence. The final objective is given by the equation:

$$\mathcal{L} = \mathcal{L}_{classification} + \lambda \cdot \mathcal{L}_{distill} \quad (3.1)$$

The authors evaluated their model using the Reuters, AAPD, IMDB, and Yelp 2014 datasets. These datasets had an average of 144.3, 167.3, 393.8, and 148.8 words per document, respectively. The KD-LSTM_{reg} model was able to achieve high scores that were close to that of BERT_{base}. Figure 3.7 illustrates the results obtained for the KD-LSTM_{reg} model compared to other models.

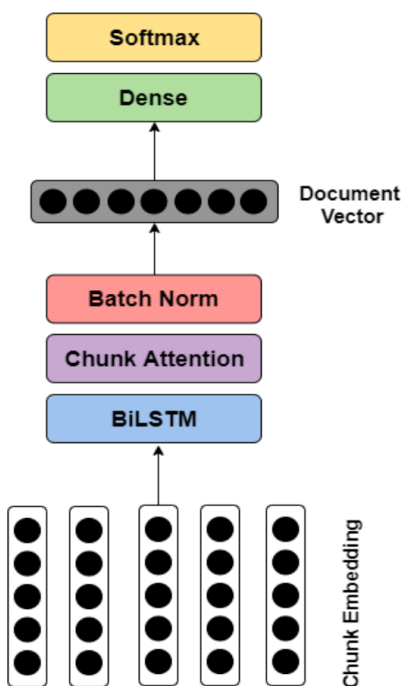


Figure 3.8: Overall architecture of BiLSTM model [70]

Long-length Legal Document Classification

The authors [70] focus on the classification of lengthy legal documents. They divide these documents into segments and then combine the embeddings for each of the segments together with a BiLSTM that is used to form a single embedding for the entire document. The authors

investigated automatic audio segmentation [67] to observe the techniques used to segment audio signals and then apply the same in the context of legal document classification.

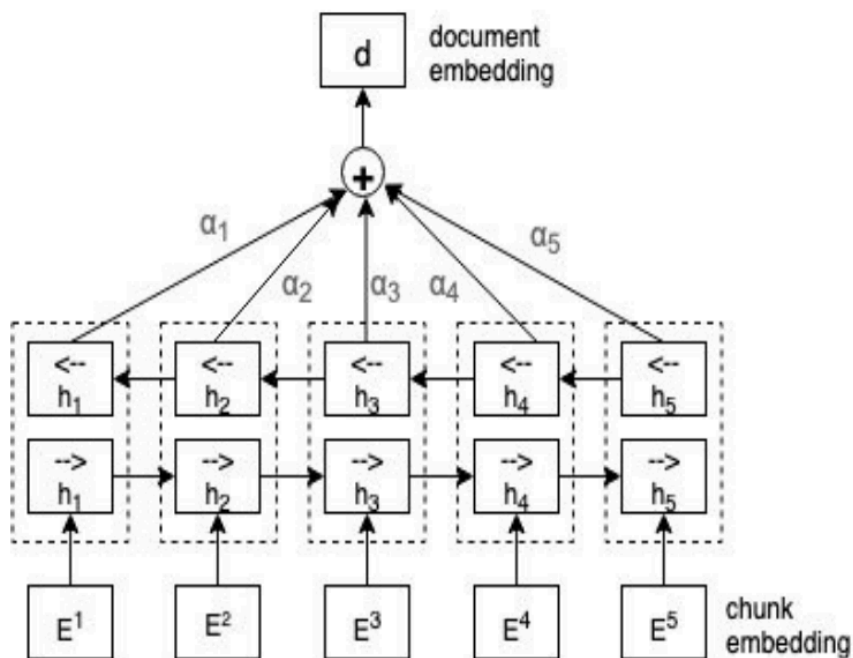


Figure 3.9: Document embedding process through BiLSTM framework [70]

The length of the documents considered in other prior work is relatively lower as compared to the legal documents the authors use as part of this study. They define a ‘document’ as one that contains a minimum of 5,000 words.

Figure 3.8 illustrates the proposed BiLSTM architecture. The authors split the original legal document into multiple chunks and then use Doc2Vec [26] to embed each of these chunks. The chunks are then aggregated into a single vector using a BiLSTM. Figure 3.9 represents this document embedding process. The authors consider different classifiers in an attempt to assess the impact caused by the segmentation of the document. The first type of models use a simple linear classifier. The final classification of the document is done using the different features that each chunk contains. A softmax layer is used to obtain the multi-class probabilities. The second type make use of a Support Vector Machine (SVM) classifier.

The authors evaluate their model on a set of documents obtained from EDGAR, which is an online public U.S. Securities and Exchange Commission (SEC) database. The documents can be grouped based on their filing types. The authors found that splitting the documents into chunks yielded higher test accuracy as compared to using the whole document.

Tables 3.10 and 3.11 indicate that splitting the document into chunks resulted in improved performance compared to models where the whole document was provided as the input.

Model	W_c	Test.F1	Val.F1
1-chunk	24,744	96.96	97.50
3-chunk	8,248	97.85	98.11
5-chunk	4,949	97.97	97.85
7-chunk	3,535	97.45	97.55
10-chunk	2,474	97.87	97.87
25-chunk	990	97.41	97.64
50-chunk	495	97.34	97.22

Figure 3.10: Performance of models using simple linear classifier [70]. W_c indicates average words per chunk.

Model	W_c	Test.F1	Val.F1
1-chunk	24,744	97.71	97.50
3-chunk	8,248	97.97	98.20
5-chunk	4,949	98.04	98.04
7-chunk	3,535	98.11	97.83
10-chunk	2,474	97.92	97.83
25-chunk	990	97.64	97.92
50-chunk	495	97.24	97.38

Figure 3.11: Performance of models using simple SVM classifier [70]. W_c indicates average words per chunk.

3.3.3 Evaluation

The evaluation metrics commonly used to gauge the performance of multi-class and binary class classification problems are: accuracy, precision, recall, and F1-score. In Equations 3.2, 3.3, 3.4, and 3.5 TP = true positive, TN = true negative, FP = false positive, and FN = false negative.

Accuracy

Accuracy identifies the fraction of predictions that the model predicted correctly. It calculates the value of the number of correct predictions against the total number of predictions. Unfortunately, in case of class-imbalanced datasets, accuracy is not the best metric for evaluation. Other more suitable metrics for such datasets are described in the following sections.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.2)$$

Precision

The precision score attempts to identify the proportion of the positive identifications that are actually correct. As can be inferred from 3.3, a classifier that has no false positives will yield a precision score of 1.0.

$$Precision = \frac{(TP)}{(TP + FP)} \quad (3.3)$$

Recall

In contrast to the precision score, recall attempts to identify the proportion of the actually positive that were identified correctly. As can be inferred from 3.4, a classifier that has no false negatives will yield a recall score of 1.0.

$$Recall = \frac{(TP)}{(TP + FN)} \quad (3.4)$$

F1-score

F1-score is the harmonic mean or weighted average of precision and recall. A high value of this score indicates a high value of both precision and recall, as the contribution of both to the F1 score is equal.

$$F1 - score = \frac{(2 \times precision \times recall)}{(precision + recall)} \quad (3.5)$$

Exact Match

Exact match or subset accuracy is a very strict metric. It measures the percentage of the set of predicted labels that exactly match the set of labels in the ground truth. [42]

$$ExactMatchRatio = \frac{1}{n} \sum_{i=1}^n I(Y_i = Z_i) \quad (3.6)$$

Hamming Loss

Hamming loss computes the fraction of incorrectly predicted labels to the total number of labels. As this is a loss function, the desired value is zero. [73]

$$\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \text{xor}(y_{i,j}, z_{i,j}) \quad (3.7)$$

Jaccard Index

The Jaccard index is also referred to as the Intersection over Union. It computes the fraction of the number of labels that were predicted correctly over the union of predicted and true labels. [73]

$$\frac{|T \cap P|}{|T \cup P|}, \quad (3.8)$$

where T and P are the true labels and predicted labels respectively.

Chapter 4

Data

In this chapter, we discuss the different datasets that have been used for the creation of the embeddings, for training and testing the baseline machine learning models created as part of the *CS6604: Digital Libraries* [6] course, and for training and testing the deep learning models.

The full-text of ETDs from three universities – Pennsylvania State University, University of Illinois at Urbana-Champaign, and Virginia Tech – have been used to train the word and document embeddings.

A small subset of the PQDT ETDs coming from Virginia Tech were used to form the baseline models generated as part of the *CS6604: Digital Libraries* [6] course. A larger dataset of the PQDT ETDs coming from multiple universities was used to train and test the deep learning models. Both full-text as well as chapter segments were extracted from this dataset.

4.1 Data Description

In this section, we describe the different datasets. There are three parts, namely: ‘Data for baseline models’, ‘Data for embeddings’, and ‘Data for deep learning models’.

4.1.1 Data for Baseline Models

The dataset discussed in this section was utilized for the *CS6604: Digital Libraries* [6] course project.

PQDT Data: Virginia Tech

Our experiments require a large corpus of ETDs and the accompanying metadata, which include ProQuest subject categories for each ETD. To obtain the data, we first identified the number of Virginia Tech ETDs belonging to each of the 432 ProQuest Subject Categories as of Fall 2019. After obtaining these counts, we selected the top (by count) 30 categories that contained the largest number of ETDs. As indicated in Figure 4.1, when these 432 subject categories are grouped by the secondary headings, “Engineering” has the highest number of ETDs (2,937) while “Language And Literature” has the lowest number of ETDs (20).

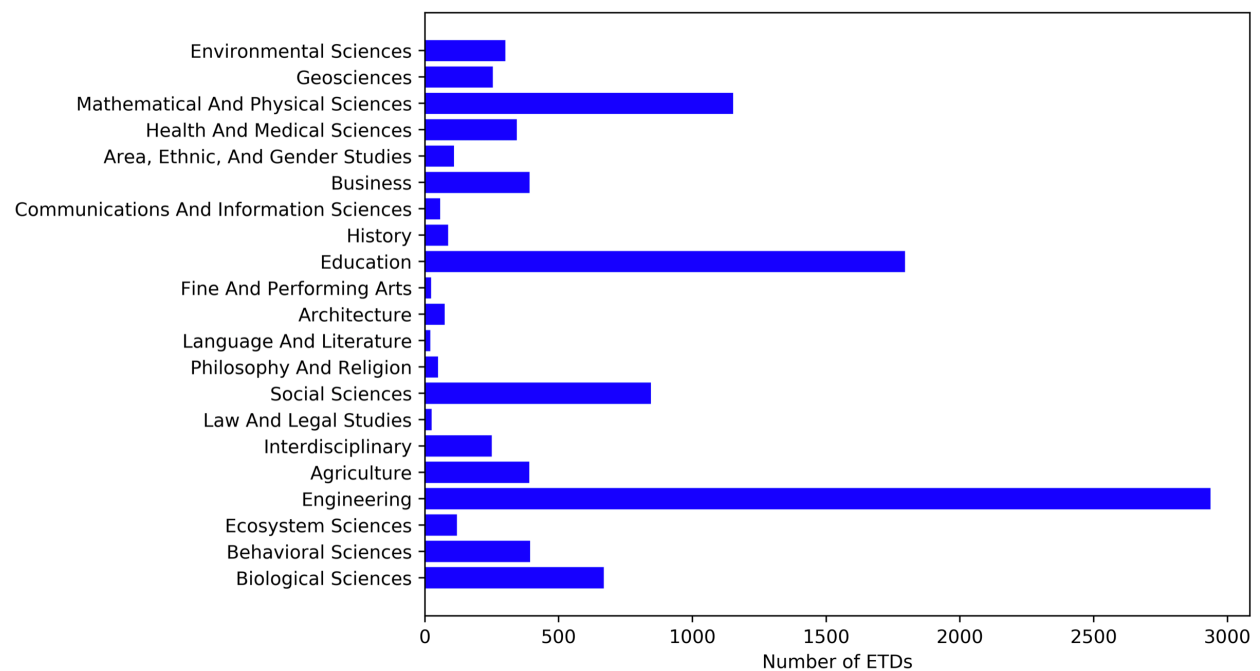


Figure 4.1: Distribution of ETDs present in each of the 21 secondary headings [6]

The ProQuest system allows authors to add multiple secondary subject categories, but authors are required to add at least one primary subject category. Figure 4.2 illustrates the distribution of ETDs with multiple subject categories across the top 10 subject categories, which were chosen as they were assigned to the maximum number of ETDs across the entire time range of documents present in the ProQuest digital library.

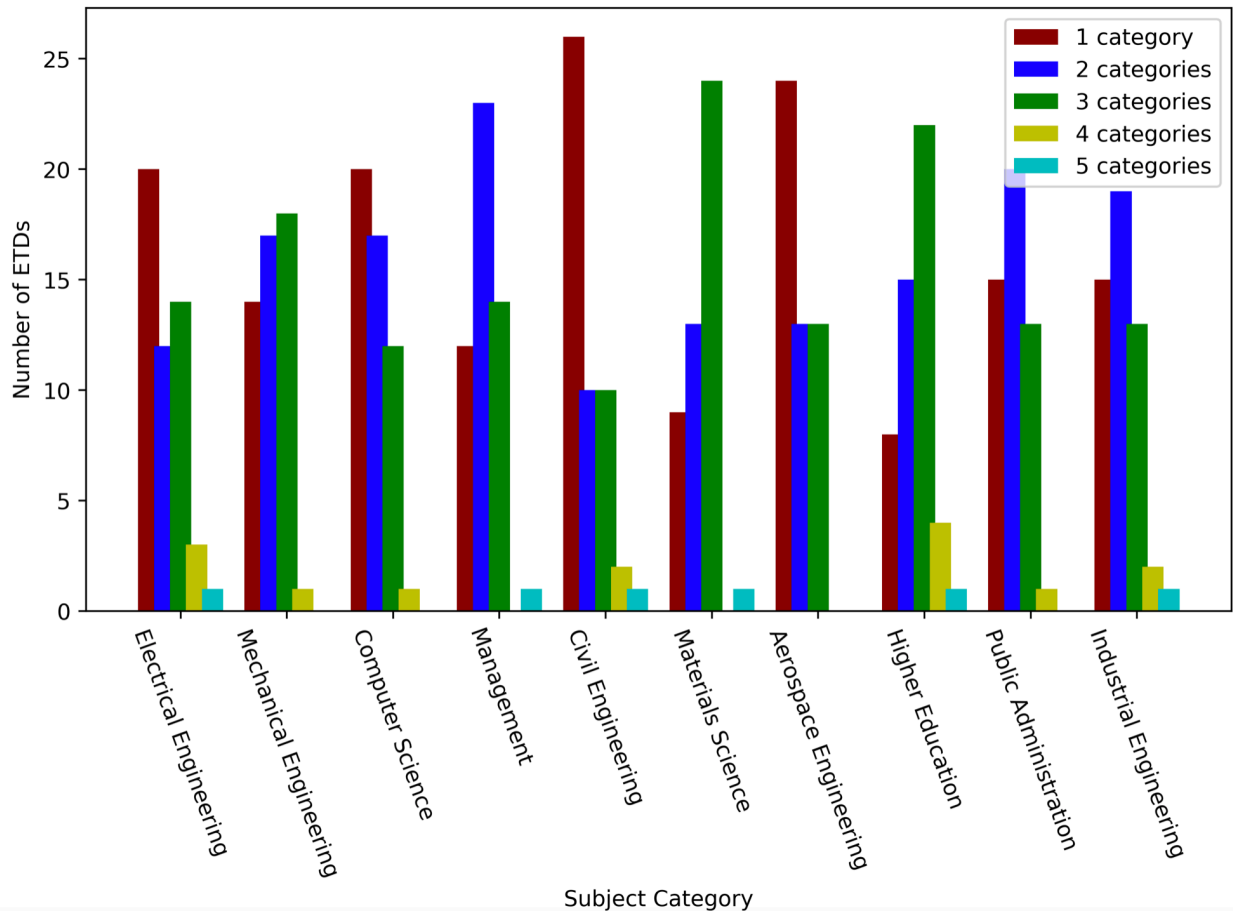


Figure 4.2: Distribution of ETDs with multiple subject categories [6]

As seen in Figure 4.2, a majority of ETDs contain 1–2 secondary categories in addition to the primary category. A small set of them contain 3–4 secondary categories. Certain subject categories such as “Environmental Science,” “Forestry,” and “Public Administration” contain ETDs with 5 secondary categories. The “Adult Education” subject category contains ETDs

with 6 or 7 secondary categories.

```

{
  "Advisor": ["Stubblefield, Harold W."],
  "Author": ["Redstrom-Plourd, Martha A."],
  "Classification": [
    "0516: Adult education",
    "0651: Continuing education",
    "0454: Management",
    "0337: American history"
  ],
  "Identifier / keyword": [
    "Social sciences",
    "Education",
    "Adult learning",
    "Career management",
    "Job search counseling",
    "Outplacement industry"
  ],
  "Title": "A history of the outplacement industry, 1960--1997: From job search
  ↪ counseling to career management. A new curriculum of adult learning"
}

```

Figure 4.3: JSON representation of partial metadata from a sample ETD [6]

4.1.2 Data for Embeddings

These three datasets were used to retrain the fastText [7][24] and Doc2Vec [26] embeddings. Thus, we chose datasets from three different universities to ensure a better spread of the disciplines associated with the collection in an attempt to maximize the number of unique words using which the embedding models could be trained. Here, we discuss the various departments present at each of these universities and the number of ETDs that belong to each of the departments.

Table 4.1: Number of ETDs in each department under the Virginia Tech thesis collection

Department	Number of ETDs
Architecture	1540
Mechanical Engineering	1512
Electrical and Computer Engineering	1148
Civil Engineering	858
Electrical Engineering	722
Computer Science	479
Industrial and Systems Engineering	429
Psychology	394
Aerospace and Ocean Engineering	325
Forestry	320
Environmental Engineering	318
Chemistry	286
Engineering Mechanics	271
Political Science	266
Chemical Engineering	253
Fisheries and Wildlife Sciences	250
Biology	237
Systems Engineering	236
Food Science and Technology	203
History	190
Civil and Environmental Engineering	190
Crop and Soil Environmental Sciences	186
Sociology	182
Veterinary Medical Sciences	180
Entomology	178

Virginia Tech ETD Collection

This collection consists of 17890 theses and 13071 dissertations as of Fall 2018. There are 326 departments present among the theses and 333 departments present among the dissertations. Table 4.1 represents the count of the number of ETDs present in the top 25 departments. Similarly, Table 4.2 represents the counts for the dissertations. As indicated by these tables, the most common departments in the thesis collection are ‘Architecture’, ‘Mechanical Engineering’, and ‘Electrical and Computer Engineering’. On the other hand,

Table 4.2: Number of ETDs in each department under the Virginia Tech dissertation collection

Department	Number of ETDs
Electrical and Computer Engineering	711
Educational Leadership and Policy Studies	611
Chemistry	577
Mechanical Engineering	499
Civil Engineering	403
Psychology	398
Educational Administration	369
Teaching and Learning	363
Industrial and Systems Engineering	326
Computer Science	299
Engineering Mechanics	271
Electrical Engineering	249
Mathematics	246
Curriculum and Instruction	238
Aerospace and Ocean Engineering	234
Statistics	233
Chemical Engineering	224
Engineering Science and Mechanics	215
Physics	206
Human Development	186
Vocational and Technical Education	175
Economics	173
Biology	172
Public Administration and Public Affairs	168
Entomology	148

the most common departments in the dissertation collection are ‘Electrical and Computer Engineering’, ‘Educational Leadership and Policy Studies’, and ‘Chemistry’.

Tables 4.3 and 4.4, represent the top 10 most frequent keywords in the thesis and dissertation collections, respectively. The most frequent keyword among the theses is ‘architecture’ followed by ‘simulation’ and ‘FPGA’. Each of these keywords could very well belong to the 3 most common departments. Similarly, for the dissertation collection, the most common keywords are ‘optimization’, ‘education’, and ‘leadership’ which again could very well belong

Table 4.3: 10 most commonly occurring keywords in the Virginia Tech thesis collection

Keyword	Number of ETDs
architecture	221
simulation	93
FPGA	80
Virginia	80
optimization	75
GIS	68
modeling	52
light	48
water quality	45
design	44

Table 4.4: 10 most commonly occurring keywords in the Virginia Tech dissertation collection

Keyword	Number of ETDs
optimization	79
education	71
leadership	67
modeling	60
simulation	30
technology	28
gender	25
self-efficacy	24
development	24
collaboration	23

to the 3 most common departments.

Pennsylvania State University ETD Collection

This collection consists of a total of 9634 ETDs that were obtained from Pennsylvania State University as of Spring 2020. In this section, we discuss the most common programs or departments associated with the ETDs present in this collection. As indicated by Table 4.5, there are a wide range of departments present in this collection ranging from ‘Mechanical

Table 4.5: Number of ETDs in each program under the Pennsylvania State University ETD Collection

Program	Number of ETDs
Mechanical Engineering	481
Electrical Engineering	449
Computer Science and Engineering	341
Industrial Engineering	321
Chemistry	317
Aerospace Engineering	297
Materials Science and Engineering	294
Psychology	244
Geosciences	194
Energy and Mineral Engineering	187
Civil Engineering	185
Curriculum and Instruction	177
Information Sciences and Technology	174
Physics	164
Engineering Science and Mechanics	156
Human Development and Family Studies	152
Chemical Engineering	148
Nuclear Engineering	136
Statistics	136
Mathematics	133
Economics	133
Biochemistry, Microbiology, and Molecular Biology	132
Architectural Engineering	129
Kinesiology	126
Adult Education	119
Meteorology	117
Business Administration	117
Sociology	108
Geography	103
Bioengineering	102

Engineering’, ‘Electrical Engineering’, and ‘Computer Science & Engineering’ to ‘Psychology’ and ‘Kinesiology’.

University of Illinois at Urbana-Champaign ETD Collection

This collection consists of a total of 7557 ETDs that were obtained from the University of Illinois at Urbana-Champaign as of Spring 2020. In this section, we discuss the most common departments as well as subjects (as entered by the authors) associated with the ETDs present in this collection.

As indicated by Table 4.6, there are a wide range of departments present in this collection ranging from ‘Electrical & Computer Eng’, ‘Computer science’, and ‘Mechanical Sci & Engineering’ to ‘Music’, and ‘Linguistics’. There are a total of 167 departments present in this collection.

Table 4.7 represents the top 10 most frequent keywords in this collection. The most frequent keyword is ‘Machine Learning’ followed by ‘Optimization’ and ‘Graphene’.

4.1.3 Data for Deep Learning Models

In this section, we discuss the larger PQDT dataset that contains ETDs from different universities collected in Spring 2020. We consider data from 28 different subject categories. This collection contains a total of 9,302 unique ETDs.

ETD Counts and Subject Categories

While our original collection has subject categories that contained more than 500 ETDs, we truncated the maximum number of ETDs in any subject category to 500. This was done to ensure that there isn’t a very large imbalance in the number of ETDs in each subject category. Tables 4.8 and 4.9 display the counts associated with each subject category.

Table 4.6: Number of ETDs in each department under the University of Illinois at Urbana-Champaign ETD collection

Department	Number of ETDs
Electrical & Computer Eng	848
Computer Science	604
Mechanical Sci & Engineering	460
Civil & Environmental Eng	385
Chemistry	278
Physics	240
Psychology	223
Aerospace Engineering	214
Educ Policy, Orgzn & Leadrshp	196
Animal Sciences	189
Crop Sciences	177
Mathematics	169
Materials Science & Engineerng	149
Kinesiology & Community Health	143
Natural Res & Env Sci	121
Nuclear, Plasma, & Rad Engr	115
Agr & Consumer Economics	102
Chemical & Biomolecular Engr	98
Educational Psychology	96
Music	94
Curriculum and Instruction	91
Bioengineering	90
Engineering Administration	89
Food Science & Human Nutrition	89
School of Molecular & Cell Bio	88
Industrial & Enterprise Sys Eng	87
Linguistics	74
Atmospheric Sciences	72
English	71
School of Integrative Biology	68

Largest Universities with Most ETDs

Our collection contains ETDs from 486 universities including 36 ETDs from 11 Historically Black Colleges and Universities (HBCUs). Table 4.10 gives the number of ETDs in our

Table 4.7: 10 most commonly occurring keywords in the University of Illinois at Urbana-Champaign ETD collection

Keyword	Number of ETDs
Machine Learning	92
Optimization	47
Graphene	42
Aging	32
soybean	31
optimization	30
Education	29
Race	28
Security	28
Gender	25

collection that belong to the largest 10 (by count) universities. Table 4.11 gives the universities and the number of ETDs that belong to those universities for all the HBCUs in our collection. 36 is a fairly small number given the size of our corpus, so we wish to include more HBCUs in our corpus in the future.

Range of Year of Publication of ETDs

As part of this research, we have mainly focused on born digital ETDs as we needed to extract the full-text and chapter segments from the ETDs. Given that working with older scanned ETDs is still a research problem, we focus on the more recent ETDs. Table 4.12 gives the number of ETDs that were published across different years in our collection.

Largest Universities Contributing ETDs for Subject Categories

Tables 4.13 and 4.14 indicate the number of ETDs contributed by the largest 3 Universities. As displayed in the table, ‘California State University, Long Beach’ contributes the largest number of ETDs to most subject categories.

Table 4.8: Number of ETDs in each subject category under the PQDT ETD collection (alphabetic order)

Subject Category	Number of ETDs
Adult education	500
Aerospace engineering	386
Biomedical engineering	485
Chemical engineering	325
Civil engineering	496
Computer Engineering	380
Computer science	500
Ecology	500
Educational leadership	500
Educational psychology	500
Electrical engineering	500
Elementary education	499
Environmental science	494
Forestry	121
Higher education	500
Industrial engineering	196
Marketing	271
Materials science	482
Mathematics	222
Mechanical engineering	500
Molecular biology	500
Occupational psychology	500
Organic chemistry	255
Public administration	428
Secondary education	500
Special education	500
Statistics	392
Teacher education	500

Table 4.9: Number of ETDs in each subject category under the PQDT ETD collection (ordered by count)

Subject Category	Number of ETDs
Computer science	500
Electrical engineering	500
Ecology	500
Adult education	500
Secondary education	500
Mechanical engineering	500
Molecular biology	500
Educational psychology	500
Higher education	500
Educational leadership	500
Special education	500
Teacher education	500
Occupational psychology	500
Elementary education	499
Civil engineering	496
Environmental science	494
Biomedical engineering	485
Materials science	482
Public administration	428
Statistics	392
Aerospace engineering	386
Computer Engineering	380
Chemical engineering	325
Marketing	271
Organic chemistry	255
Mathematics	222
Industrial engineering	196
Forestry	121

Table 4.10: Number of ETDs in the largest 10 universities (based on number of ETDs present)

University	Number of ETDs
California State University, Long Beach	864
The George Washington University	389
Southern Illinois University at Edwardsville	275
Capella University	216
University of Louisiana at Lafayette	193
Pepperdine University	180
University of Maryland, College Park	171
Lindenwood University	158
Walden University	154
University of Phoenix	148

Table 4.11: Number of ETDs from Historically Black Colleges and Universities (HBCUs)

University	Number of ETDs
Tennessee State University	9
North Carolina Agricultural and Technical State University	8
Howard University	4
Bowie State University	3
Delaware State University	3
Southern University and Agricultural and Mechanical College	3
Hampton University	2
Bethune-Cookman University	1
Florida Agricultural and Mechanical University	1
Morgan State University	1
Savannah State University	1

Co-occurrence of Subject Categories

In this section, we examine the co-occurrence of the different subject categories. Figures 4.4 and 4.5 represent the co-occurrence matrix between the 28 subject categories. Figure 4.6 represents a heat map of the occurrence matrix.

As indicated by these figures, it can be seen that the maximum co-occurrence between any two subject categories is between ‘Computer Engineering’ and ‘Computer science’ with a

Table 4.12: Number of ETDs associated with different years of publication

Year of publication	Number of ETDs
2018	1170
2015	1042
2017	1042
2016	956
2014	919
2019	824
2012	771
2013	769
2011	659
2010	462
2009	392
2008	189
2007	55
2020	22
2006	15
2005	6
2004	5
2001	3
2000	1

total of 175 such ETDs. This was to be expected given that there is often some amount of overlap between these two subject categories. The second most frequent co-occurrence is between ‘Educational psychology’ and ‘Higher education’ with 136 ETDs, followed by ‘Aerospace engineering’ and ‘Mechanical engineering’ with a 124 ETDs, followed by ‘Educational leadership’ and ‘Secondary education’ with 122 ETDs.

An interesting co-occurrence is seen between ‘Public administration’ and ‘Adult education’ with 4 such ETDs. Similarly, another interesting co-occurrence is visible between ‘Marketing’ and ‘Industrial Engineering’ with an ETD titled ‘What contributes to a technical purchasing decision maker’s reliance on brand name for design decisions involving I&T products’.

Table 4.13: Part 1: Number of ETDs contributed by the largest 3 universities under each subject category

Subject Category	Univ 1	Univ 2	Univ 3
Adult education	Capella University (51)	California State University, Long Beach (38)	Pepperdine University (33)
Aerospace engineering	California State University, Long Beach (64)	University of Colorado at Boulder (28)	University of Maryland, College Park (27)
Biomedical engineering	The University of Iowa (22)	California State University, Long Beach (21)	State University of New York at Buffalo (17)
Chemical engineering	California State University, Long Beach (26)	University of Louisiana at Lafayette (15)	Yale University (11)
Civil engineering	University of Louisiana at Lafayette (35)	California State University, Long Beach (34)	University of Colorado at Boulder (28)
Computer Engineering	California State University, Long Beach (46)	University of Louisiana at Lafayette (23)	Carnegie Mellon University (14)
Computer science	California State University, Long Beach (59)	The George Washington University (25)	University of Louisiana at Lafayette (15)
Ecology	California State University, Long Beach (45)	University of California, Davis (28)	University of Louisiana at Lafayette (22)
Educational leadership	The George Washington University (33)	Pepperdine University (30)	California State University, Long Beach (24)
Educational psychology	California State University, Long Beach (43)	Walden University (26)	Pepperdine University (23)

Common Departments in Largest Universities

Here we present the 3 most common departments for each of our largest 10 universities as indicated in Table 4.10. Table 4.16 presents the most common 3 departments. It is

Table 4.14: Part 2: Number of ETDs contributed by the largest 3 universities under each subject category

Subject Category	Univ 1	Univ 2	Univ 3
Electrical engineering	California State University, Long Beach (153)	Southern Illinois University at Edwardsville (29)	Carnegie Mellon University (13)
Elementary education	California State University, Long Beach (44)	Lindenwood University (43)	Pepperdine University (21)
Environmental science	Southern Illinois University at Edwardsville (31)	University of Maryland, College Park (24)	University of California, Davis (19)
Forestry	Mississippi State University (20)	State University of New York College of Environmental Science and Forestry (11)	University of Montana (6)
Higher education	California State University, Long Beach (55)	The George Washington University (27)	Lindenwood University (23)
Industrial engineering	Southern Illinois University at Edwardsville (20)	Mississippi State University (19)	The George Washington University (13)
Marketing	Capella University (17)	University of Phoenix (15)	California State University, Long Beach (15)
Materials science	California State University, Long Beach (32)	University of California, Santa Barbara (20)	Universitaet Bayreuth (Germany) (17)

interesting to note that out of the largest 10 universities, 5 universities have an ‘Education’ related department as their most common department. It is also interesting to note that all the ETDs in our corpus that belong to ‘Lindenwood University’ belong to the ‘Education’ department.

Table 4.15: Part 3: Number of ETDs contributed by the largest 3 universities under each subject category

Subject Category	Univ 1	Univ 2	Univ 3
Mathematics	California State University, Long Beach (15)	University of California, Berkeley (12)	University of Louisiana at Lafayette (12)
Mechanical engineering	California State University, Long Beach (49)	Southern Illinois University at Edwardsville (40)	University of Louisiana at Lafayette (21)
Molecular biology	California State University, Long Beach (38)	University of Arkansas for Medical Sciences (27)	New York University (17)
Occupational psychology	Capella University (62)	California State University, Long Beach (55)	Pepperdine University (35)
Organic chemistry	Southern Illinois University at Edwardsville (31)	California State University, Long Beach (24)	Yale University (11)
Public administration	The George Washington University (64)	Walden University (36)	University of Phoenix (35)
Secondary education	California State University, Long Beach (52)	Lindenwood University (42)	Pepperdine University (29)
Special education	California State University, Long Beach (46)	The George Washington University (43)	Lindenwood University (25)
Statistics	California State University, Long Beach (62)	The George Washington University (29)	University of California, Berkeley (15)
Teacher education	Pepperdine University (27)	The George Washington University (21)	California State University, Long Beach (21)

Most Frequent Keywords in Subject Categories

In this section, we present the top 3 most frequent keywords for each subject category. Tables 4.17 and 4.18 display these keywords. It is interesting to note that in case of subject

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry
Statistics	0	37	9	8	29	1	7	5	1	1	3	11	3	3	3	3	1	4	0	0	0	4	0	0	0	1	0	2
Computer science	37	0	66	6	15	0	6	175	0	3	15	6	13	5	2	4	3	1	3	0	0	1	5	12	0	2	3	0
Electrical engineering	9	66	0	3	4	0	1	98	0	0	24	3	27	0	0	4	9	1	36	0	0	0	0	71	1	0	0	0
Civil engineering	8	6	3	0	2	0	1	2	0	0	31	7	4	0	0	9	8	2	24	0	0	0	0	1	1	0	0	0
Mathematics	29	15	4	2	0	0	1	0	1	1	3	1	2	0	1	0	0	1	1	3	1	1	0	1	0	2	0	0
Public administration	1	0	0	0	0	0	0	0	4	1	0	0	0	0	0	4	0	9	0	10	0	0	1	0	0	0	13	1
Ecology	7	6	1	1	1	0	0	0	1	2	0	0	1	12	1	92	0	1	0	0	0	0	0	0	0	1	0	51
Computer Engineering	5	175	98	2	0	0	0	0	0	0	9	2	4	0	0	1	0	0	1	0	0	0	0	5	0	0	0	0
Adult education	1	0	0	0	1	4	1	0	0	8	0	0	0	0	22	0	0	97	0	61	8	35	3	0	0	6	5	0
Secondary education	1	3	0	0	1	1	2	0	8	0	0	0	0	0	48	0	0	22	0	122	35	35	1	0	0	12	6	0
Mechanical engineering	3	15	24	31	3	0	0	9	0	0	0	14	124	0	0	1	23	0	81	0	0	0	0	44	1	0	0	1
Industrial engineering	11	6	3	7	1	0	0	2	0	0	14	0	5	0	0	0	1	2	5	0	0	0	1	8	0	0	3	0
Aerospace engineering	3	13	27	4	2	0	1	4	0	0	124	5	0	0	0	0	6	0	21	0	0	0	1	1	1	0	3	0
Molecular biology	3	5	0	0	0	0	12	0	0	0	0	0	0	0	0	4	9	0	1	0	0	0	0	16	3	0	0	0

Figure 4.4: Part 1: Co-occurrence of the 28 subject categories

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry
Educational psychology	3	2	0	0	1	0	1	0	22	48	0	0	0	0	0	0	136	0	95	50	34	1	0	0	66	16	0	
Environmental science	3	4	4	9	0	4	92	1	0	0	1	0	0	4	0	0	2	2	0	1	1	0	1	3	0	0	10	
Chemical engineering	1	3	9	8	0	0	0	0	0	23	1	6	9	0	0	0	0	53	0	0	0	0	0	18	6	0	0	0
Higher education	4	1	1	2	1	9	1	0	97	22	0	2	0	0	136	2	0	0	100	26	33	9	0	0	3	14	0	
Materials science	0	3	36	24	1	0	0	1	0	0	81	5	21	1	0	2	53	0	0	0	0	0	0	22	8	0	0	1
Educational leadership	0	0	0	0	3	10	0	0	61	122	0	0	0	0	95	0	0	100	0	0	59	112	1	0	0	102	39	0
Special education	0	0	0	0	1	0	0	0	8	35	0	0	0	0	50	1	0	26	0	59	0	46	0	0	0	48	4	0
Teacher education	0	1	0	0	1	0	0	0	35	35	0	0	0	0	34	1	0	33	0	112	46	0	0	0	0	55	9	0
Marketing	4	5	0	0	0	1	0	0	3	1	0	1	1	0	1	0	0	9	0	1	0	0	0	0	0	2	1	
Biomedical engineering	0	12	71	1	1	0	0	5	0	0	44	8	1	16	0	1	18	0	22	0	0	0	0	0	1	0	0	0
Organic chemistry	0	0	1	1	0	0	0	0	0	0	1	0	1	3	0	3	6	0	8	0	0	0	0	1	0	0	0	0
Elementary education	1	2	0	0	2	0	1	0	6	12	0	0	0	0	66	0	0	3	0	102	48	55	0	0	0	1	0	
Occupational psychology	0	3	0	0	0	13	0	0	5	6	0	3	3	0	16	0	0	14	0	39	4	9	2	0	0	1	0	0
Forestry	2	0	0	0	0	1	51	0	0	0	1	0	0	0	0	10	0	0	1	0	0	1	0	0	0	0	0	0

Figure 4.5: Part 2: Co-occurrence of the 28 subject categories

categories such as ‘Adult education’, ‘Higher education’, ‘Marketing’, ‘Mathematics’, and ‘Special education’ the subject category itself is the most common keyword. The keyword ‘Machine learning’ is the most frequent keyword in case of multiple subject categories such as ‘Computer Engineering’, ‘Computer science’, ‘Electrical engineering’, and ‘Statistics’. Since each of these subject categories does in fact make use of or contribute to the understanding of ‘Machine learning’, this is in keeping with our understanding of the subject categories. While ‘Electrical engineering’ might appear to be a surprise in this case, this can be attributed to the fact that a large number of the ETDs belonging to the ‘Electrical engineering’ subject category have their department as ‘Electrical and Computer engineering’ as indicated in Table 4.19.

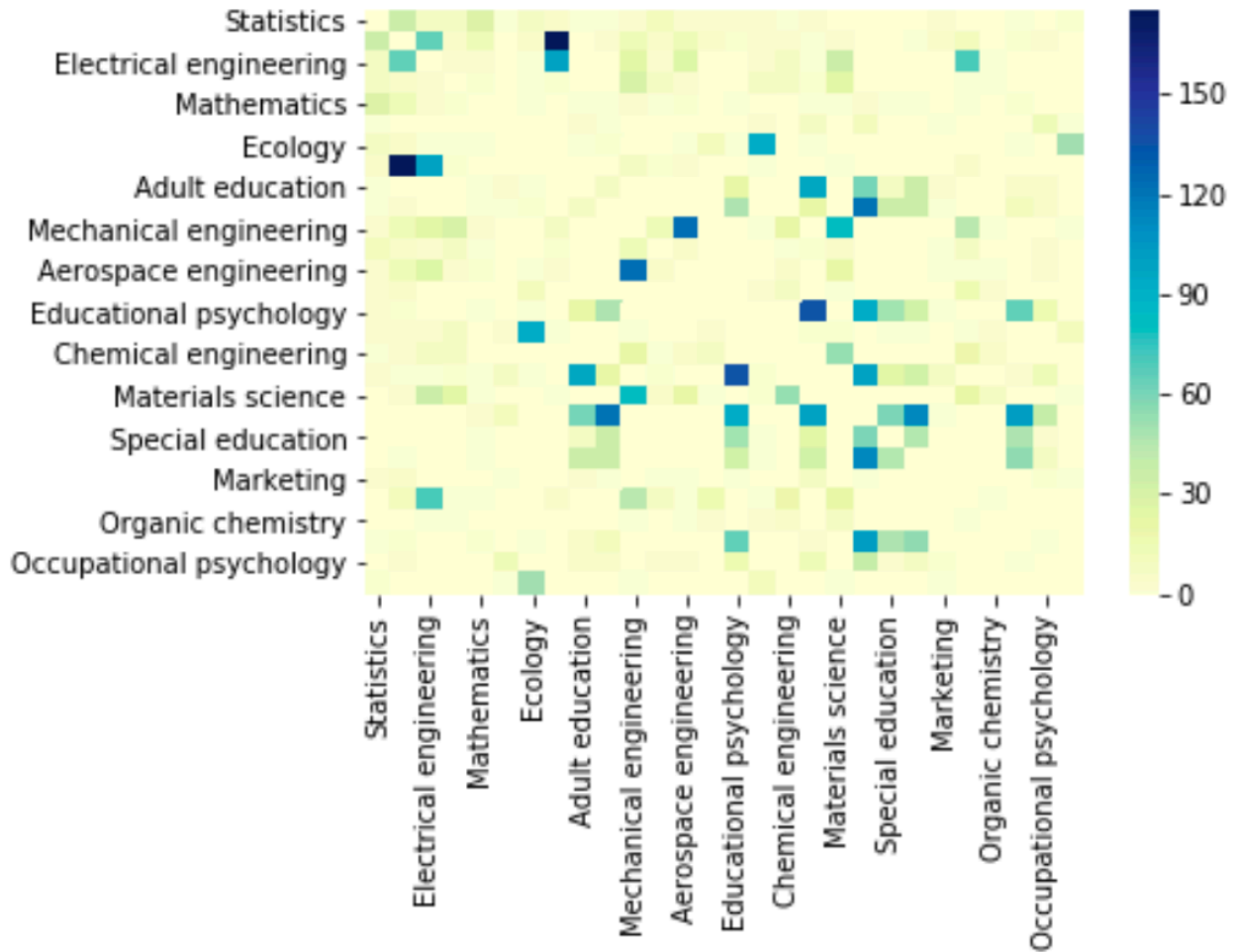


Figure 4.6: Co-occurrence heat map of the 28 subject categories (without zero values)

Most Common Departments Associated with Subject Categories

In this section, we present the top 3 most common departments that are associated with each subject category. Tables 4.19 and 4.20 display these departments. It is interesting to note that subject categories such as ‘Adult education’, ‘Educational leadership’, ‘Educational psychology’, ‘Elementary education’, ‘Higher education’, ‘Secondary education’, ‘Special education’, and ‘Teacher education’ all have ‘Education’ as the most common department. This indicates that merely using the department as a feature to the machine learning mod-

Table 4.16: Top 3 most frequent departments present in our largest 10 universities. Count of ETDs associated with each department is in brackets.

University	Dept. 1	Dept. 2	Dept. 3
California State University, Long Beach	Electrical Engineering (148)	Educational Leadership (68)	Mechanical and Aerospace Engineering (58)
The George Washington University	Special Education (33)	Education and Human Development (28)	Public Policy and Public Administration (27)
Southern Illinois University at Edwardsville	Mechanical and Industrial Engineering (51)	Chemistry (33)	Psychology (32)
Capella University	School of Education (64)	Education (28)	Harold Abel School of Social and Behavioral Sciences (27)
University of Louisiana at Lafayette	Civil Engineering (34)	Biology (26)	Computer Engineering (19)
Pepperdine University	Education (154)	Organizational Development (15)	Psychology (11)
University of Maryland, College Park	Aerospace Engineering (27)	Civil Engineering (20)	Marine-Estuarine-Environmental Sciences (14)
Lindenwood University	Education (158)	-	-
Walden University	Education (63)	Public Policy and Administration (31)	Psychology (28)
University of Phoenix	Advanced Studies (18)	Organizational Leadership (7)	School of Advanced Studies (6)

els wouldn't be sufficient as it would not be able to offer the granularity that is offered by the subject categories. It is also interesting that 'Ecology' is the third most common department for the 'Ecology' subject category while 'Biological Sciences' is the most common department with 74 ETDs. Subject categories such as 'Occupational psychology' and

Table 4.17: Part 1: Most frequent keywords in each subject category. Frequency of the keywords is represented in brackets.

Subject Category	Keyword 1	Keyword 2	Keyword 3
Adult education	Adult education (30)	Adult learning (29)	Andragogy (20)
Aerospace engineering	Computational fluid dynamics (11)	Optimization (7)	Aerodynamics (7)
Biomedical engineering	Tissue engineering (18)	Biomechanics (10)	Drug delivery (9)
Chemical engineering	Carbon dioxide (7)	Density functional theory (7)	Energy storage (7)
Civil engineering	Concrete (12)	Water quality (10)	Optimization (9)
Computer Engineering	Machine learning (17)	Security (12)	Computer vision (10)
Computer science	Machine learning (42)	Deep learning (18)	Computer vision (18)
Ecology	Climate change (28)	California (19)	Restoration (19)
Educational leadership	Leadership (49)	Professional development (24)	Higher education (20)
Educational psychology	Self-efficacy (23)	Education (16)	Stress (15)
Electrical engineering	Machine learning (13)	Image processing (12)	Optimization (10)
Elementary education	Elementary (24)	Professional development (24)	Elementary education (23)
Environmental science	Climate change (21)	Water quality (20)	Sustainability (13)
Forestry	Forest management (8)	Restoration (6)	Forestry (5)

‘Organic chemistry’ cut across a variety of departments such as ‘Psychology’, ‘Education’, ‘Social and Behavioral Sciences’ and ‘Chemistry’, ‘Chemistry and Bio-chemistry’, and ‘Civil Engineering’ respectively.

Table 4.18: Part 2: Most frequent keywords in each subject category. Frequency of the keywords is represented in brackets.

Subject Category	Keyword 1	Keyword 2	Keyword 3
Higher education	Higher education (68)	Retention (20)	Persistence (19)
Industrial engineering	Human factors (6)	Simulation (6)	Optimization (5)
Marketing	Marketing (36)	Advertising (20)	Social media (17)
Materials science	Thin films (10)	Graphene (10)	Silicon (8)
Mathematics	Mathematics (7)	Combinatorics (4)	Functional analysis (4)
Mechanical engineering	Heat transfer (13)	Optimization (11)	Computational fluid dynamics (8)
Molecular biology	Breast cancer (13)	Cancer (11)	Epigenetics (11)
Occupational psychology	Leadership (50)	Job satisfaction (45)	Burnout (31)
Organic chemistry	Synthesis (12)	Catalysis (8)	Asymmetric catalysis (7)
Public administration	Leadership (39)	Emergency management (17)	Collaboration (16)
Secondary education	High school (41)	Education (19)	Secondary education (18)
Special education	Special education (103)	Autism (49)	Inclusion (40)
Statistics	Machine learning (22)	Statistics (11)	Bayesian (8)
Teacher education	Professional development (87)	Teacher education (40)	Teacher preparation (29)

Number of Keywords per Subject Categories

Tables 4.21 and 4.22 display the number of keywords present in each subject category. Each ETD can be associated with one or more keywords. Thus, the number of keywords is much larger than the number of ETDs present in each subject category. The most number of varied keywords exist for the ‘Civil engineering’, ‘Mechanical engineering’, and ‘Environmental science’ subject categories.

Table 4.19: Part 1: Most common departments associated with each subject category. Count of ETDs associated with each department is in brackets.

Subject Category	Dept. 1	Dept. 2	Dept. 3
Adult education	Education (115)	School of Education (43)	Educational Leadership (21)
Aerospace engineering	Aerospace Engineering (93)	Mechanical and Aerospace Engineering (74)	Mechanical Engineering (23)
Biomedical engineering	Biomedical Engineering (119)	Bioengineering (43)	Electrical Engineering (33)
Chemical engineering	Chemical Engineering (113)	Chemical and Biological Engineering (20)	Mechanical Engineering (13)
Civil engineering	Civil Engineering (199)	Civil and Environmental Engineering (101)	Civil & Environmental Engineering (18)
Computer Engineering	Electrical and Computer Engineering (76)	Computer Engineering (50)	Electrical Engineering (37)
Computer science	Computer Science (122)	Electrical Engineering (33)	Computer Engineering and Computer Science (28)
Ecology	Biological Sciences (74)	Biology (59)	Ecology (20)
Educational leadership	Education (148)	Educational Leadership (102)	School of Education (21)
Educational psychology	Education (108)	Psychology (42)	Educational Leadership (25)
Electrical engineering	Electrical Engineering (238)	Electrical and Computer Engineering (92)	Electrical & Computer Engineering (17)
Elementary education	Education (188)	Educational Leadership (31)	School of Education (30)
Environmental science	Environmental Science (39)	Environmental Sciences (35)	Environmental Studies (28)
Forestry	Forestry (18)	Forest Products (5)	Geography (5)

ETDs Belonging to STEM and non-STEM Subject Categories

In this section, we examine our dataset to identify ETDs that cut across both STEM as well as non-STEM subject categories. We first categorize our subject categories as STEM or

Table 4.20: Part 2: Most common departments associated with each subject category. Count of ETDs associated with each department is in brackets.

Subject Category	Dept. 1	Dept. 2	Dept. 3
Higher education	Education (106)	Educational Leadership (56)	School of Education (18)
Industrial engineering	Industrial Engineering (30)	Industrial and Systems Engineering (25)	Mechanical and Industrial Engineering (19)
Marketing	Marketing (19)	Business Administration (18)	Communication (11)
Materials science	Materials Science and Engineering (46)	Mechanical Engineering (37)	Physics (31)
Mathematics	Mathematics (100)	Mathematics and Statistics (20)	Sciences (11)
Mechanical engineering	Mechanical Engineering (184)	Mechanical and Aerospace Engineering (80)	Mechanical and Industrial Engineering (33)
Molecular biology	Biological Sciences (54)	Biology (44)	Biochemistry and Molecular Biology (25)
Occupational psychology	Psychology (88)	Education (32)	Harold Abel School of Social and Behavioral Sciences (15)
Organic chemistry	Chemistry (142)	Chemistry and Biochemistry (15)	Civil Engineering (3)
Public administration	Public Policy and Administration (36)	Public Policy and Public Administration (27)	Public Administration (18)
Secondary education	Education (165)	Educational Leadership (44)	School of Education (29)
Special education	Education (107)	Special Education (105)	Educational Leadership (34)
Statistics	Statistics (61)	Mathematics and Statistics (45)	Biostatistics (11)
Teacher education	Education (150)	Educational Leadership (39)	School of Education (24)

non-STEM. We use the list of STEM fields indicated in [43] and [44]. Table 4.23 displays the categorization of the subject categories. Our dataset contains 17 STEM and 11 non-STEM

Table 4.21: Number of keywords associated with each subject category (alphabetic order)

Subject Category	Number of keywords
Adult education	1789
Aerospace engineering	1495
Biomedical engineering	1947
Chemical engineering	1342
Civil engineering	2046
Computer Engineering	1459
Computer science	1950
Ecology	2004
Educational leadership	1749
Educational psychology	1879
Electrical engineering	1868
Elementary education	1603
Environmental science	2002
Forestry	518
Higher education	1799
Industrial engineering	798
Marketing	1190
Materials science	1943
Mathematics	839
Mechanical engineering	2011
Molecular biology	1924
Occupational psychology	1739
Organic chemistry	962
Public administration	1812
Secondary education	1759
Special education	1464
Statistics	1522
Teacher education	1617

Table 4.22: Number of keywords associated with each subject category (ordered by count)

Subject Category	Number of keywords
Civil engineering	2046
Mechanical engineering	2011
Ecology	2004
Environmental science	2002
Computer science	1950
Biomedical engineering	1947
Materials science	1943
Molecular biology	1924
Educational psychology	1879
Electrical engineering	1868
Public administration	1812
Higher education	1799
Adult education	1789
Secondary education	1759
Educational leadership	1749
Occupational psychology	1739
Teacher education	1617
Elementary education	1603
Statistics	1522
Aerospace engineering	1495
Special education	1464
Computer Engineering	1459
Chemical engineering	1342
Marketing	1190
Organic chemistry	962
Mathematics	839
Industrial engineering	798
Forestry	518

categories.

Table 4.23: Categorization of subject categories into STEM and non-STEM

STEM	non-STEM
Aerospace engineering	Adult education
Biomedical engineering	Educational leadership
Chemical engineering	Educational psychology
Civil engineering	Elementary education
Computer Engineering	Higher education
Computer science	Marketing
Ecology	Occupational psychology
Electrical engineering	Public administration
Environmental science	Secondary education
Forestry	Special education
Industrial engineering	Teacher education
Materials science	
Mathematics	
Mechanical engineering	
Molecular biology	
Organic chemistry	
Statistics	

Figure 4.7 represents the number of ETDs that belong to STEM, non-STEM, or STEM + non-STEM subject categories. As can be seen from this pie chart, a majority of the ETDs, i.e., 57% of the total number of ETDs, belong to STEM-related subject categories. Only a very small percentage, 1% of all ETDs, belong to both STEM + non-STEM subject categories.

Tables 4.25 and 4.26 display the number of ETDs that exist for the different STEM and non-STEM category combinations. Our collection contains 66 ETDs that cut across these disciplines. The combination of ‘Marketing’ & ‘Computer Science’, and ‘Public administration’ & ‘Environmental science’ contain the maximum (4) ETDs each. The titles of the ETDs in the ‘Marketing’ and ‘Computer science’ pair include ‘Go niche or go home: Influence maximization in the presence of strong opponent’, ‘Computational Models for Scheduling in

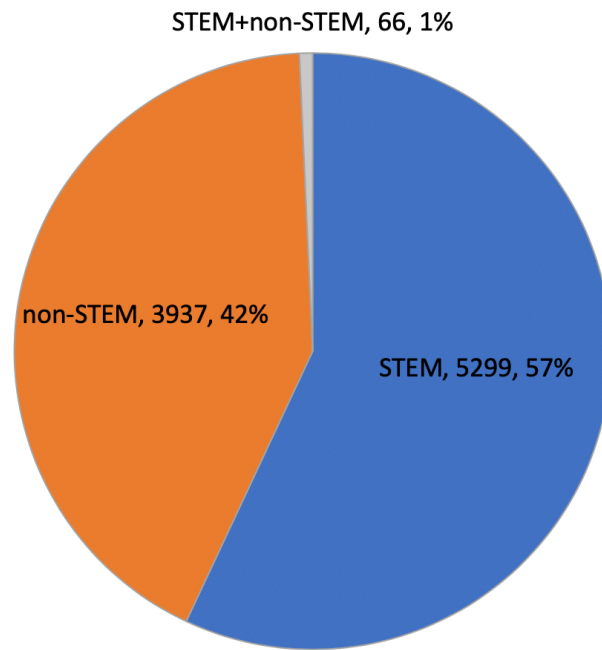


Figure 4.7: Number of STEM, non-STEM, and STEM + non-STEM ETDs

Online Advertising’, ‘A study of soft skills for IT workers in recruitment advertising’, and ‘Product reputation manipulation: The characteristics and impact of skill reviews’. Similarly, the titles of the ETDs in the ‘Public administration’ and ‘Environmental science’ pair include ‘Government-to-private sector energy programs: Identification of common elements leading to successful implementation’, ‘Nonprofit organizations and the environmental policy outcomes: A systematic inquiry into the role of different types of nonprofits to influence the processes and outcomes of environmental policy’, ‘Investigating the relationship between the policy implementation process and the utilization of information technology in a constitutional republic: The case of I-269 NEPA process’, and ‘Evaluation at EPA: Determinants of the U.S. Environmental Protection Agency’s Capacity to Supply Program Evaluation’. It is interesting to note that ‘Statistics’ and ‘Computer science’ are the most common STEM subject categories that appear in this list. Similarly, ‘Higher education’ and ‘Marketing’ are the most common non-STEM categories in this list.

STEM and Non-STEM Subject Categories in Each University

In this section, we examine the number of ETDs that belong to STEM and non-STEM subject categories in each of the largest 10 universities.

Table 4.24: Number of STEM and non-STEM ETDs associated belonging to the largest 10 universities

University	STEM ETDs	non-STEM ETDs	STEM + non-STEM ETDs
California State University, Long Beach	541	314	9
Capella University	7	208	1
Lindenwood University	0	158	0
Pepperdine University	5	175	0
Southern Illinois University at Edwardsville	212	63	0
The George Washington University	159	227	3
University of Louisiana at Lafayette	161	31	1
University of Maryland, College Park	143	28	0
University of Phoenix	11	137	0
Walden University	7	146	1

Table 4.25: Part 1: Number of ETDs that contain subject categories from STEM and non-STEM

Category 1 (STEM)	Category 2 (non-STEM)	Category 3	Number of ETDs
Computer science	Marketing	-	4
Environmental science	Public administration	-	4
Statistics	Educational psychology	-	3
Statistics	Higher education	-	3
Statistics	Marketing	-	3
Computer science	Occupational psychology	-	3
Computer science	Secondary education	-	3
Industrial engineering	Occupational psychology	-	3
Aerospace engineering	Occupational psychology	-	3
Computer science	Elementary education	-	2
Computer science	Educational psychology	-	2
Civil engineering	Higher education	-	2
Mathematics	Educational leadership	Elementary education	2
Ecology	Secondary education	-	2
Environmental science	Higher education	-	2
Statistics	Higher education	Industrial engineering	1
Statistics	Marketing	Computer science	1
Statistics	Elementary education	-	1
Statistics	Public administration	-	1
Statistics	Adult education	-	1
Statistics	Secondary education	-	1

Number of Words in Title for Each Subject Category

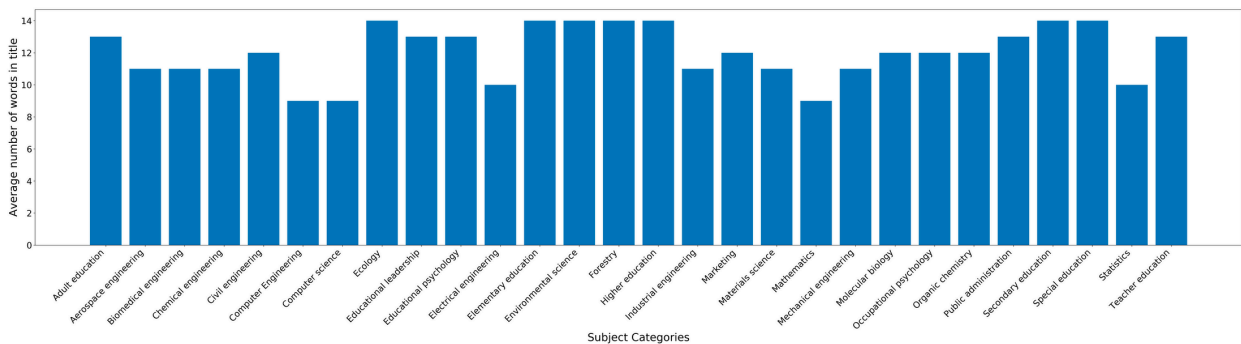


Figure 4.8: Average number of words present in the title of ETDs for each subject category

Table 4.26: Part 2: Number of ETDs that contain subject categories from STEM and non-STEM

Category 1 (STEM)	Category 2 (non-STEM)	Category 3	Number of ETDs
Computer science	Teacher education	-	1
Computer science	Higher education	-	1
Electrical engineering	Higher education	-	1
Mathematics	Higher education	-	1
Mathematics	Educational psychology	Secondary education	1
Mathematics	Teacher education	-	1
Mathematics	Adult education	-	1
Mathematics	Educational leadership	-	1
Mathematics	Special education	-	1
Forestry	Public administration	-	1
Ecology	Higher education	-	1
Ecology	Adult education	-	1
Ecology	Elementary education	Educational psychology	1
Industrial engineering	Higher education	-	1
Industrial engineering	Marketing	-	1
Aerospace engineering	Marketing	-	1
Environmental science	Teacher education	-	1
Environmental science	Special education	-	1
Forestry	Marketing	-	1

In this section, we present the average number of words that are present in the title of ETDs for a particular subject category. Figure 4.8 illustrates this count. The maximum average number of words in the title are present in ‘Ecology’, ‘Elementary education’, ‘Environmental science’, ‘Forestry’, ‘Higher education’, ‘Secondary education’, and ‘Special education’, each with a count of 14 words.

Number of Words in Abstract for Each Subject Category

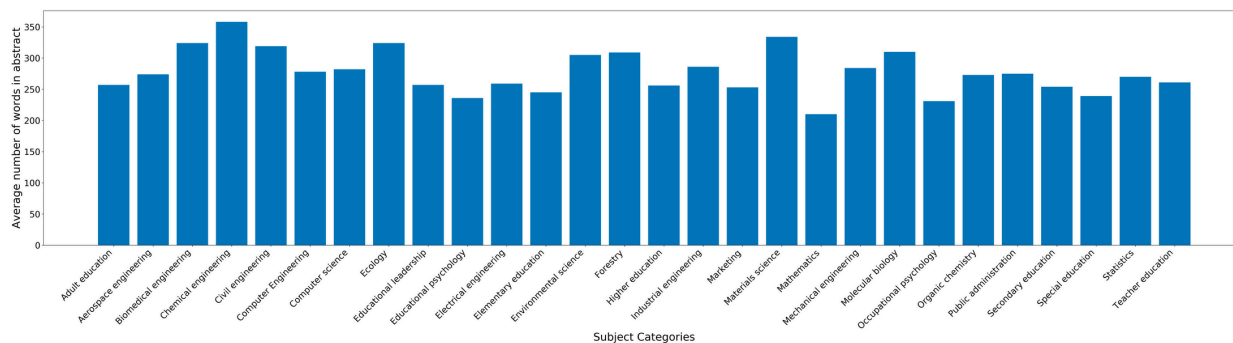


Figure 4.9: Average number of words present in the abstract of ETDs for each subject category

In this section, we present the average number of words that are present in the abstract of ETDs for a particular subject category. Figure 4.9 illustrates this count. The maximum average number of words in the abstract are present in ‘Chemical engineering’ with 358 words, followed by ‘Materials science’ with 334 words, followed by ‘Ecology’ and ‘Biomedical engineering’ with 324 words.

Number of Words in Full Text for Each Subject Category

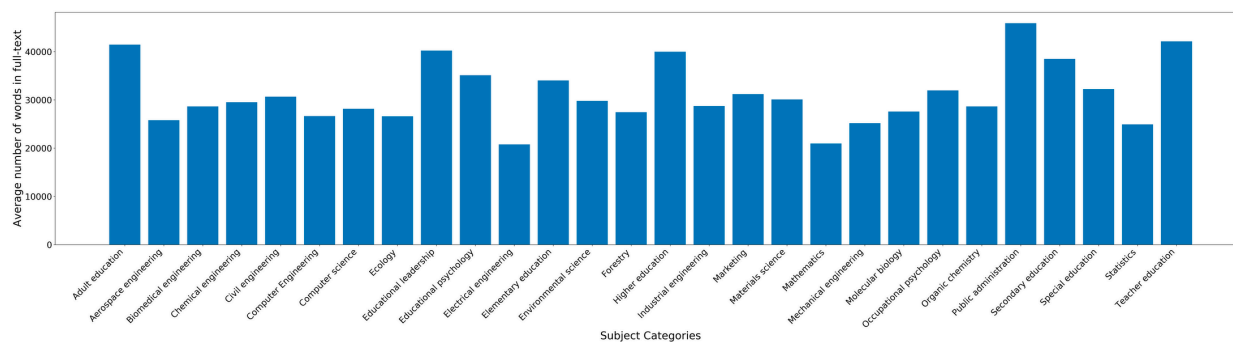


Figure 4.10: Average number of words present in the full-text of ETDs for each subject category

In this section, we present the average number of words that are present in the full-text of an ETD for a particular subject category. Figure 4.10 illustrates this count. The maximum average number of words in the full-text are present in ‘Public administration’ with 45885 words, followed by ‘Teacher education’ with 42139 words, followed by ‘Adult education’ with 41457 words and ‘Educational leadership’ with 40223 words.

4.1.4 Number of Chapters in Each Subject Category

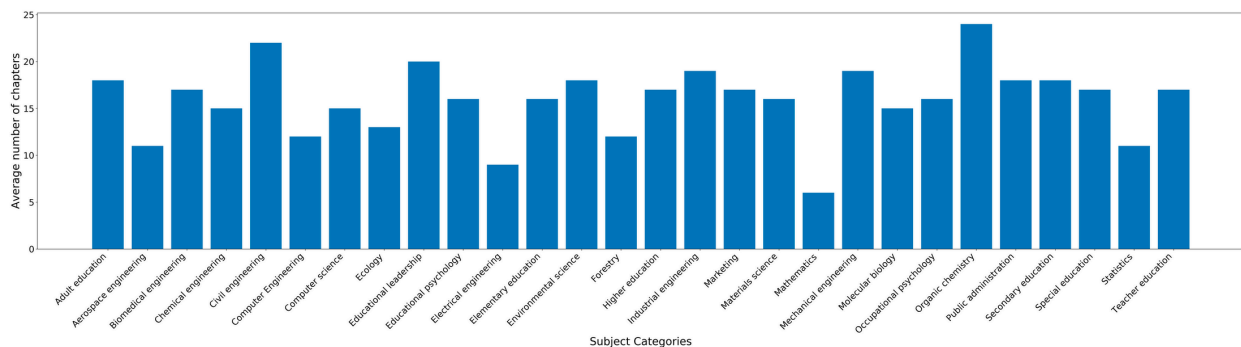


Figure 4.11: Average number of chapters in each subject category

In this section, we present the average number of chapters that exist for a particular subject category. Figure 4.11 illustrates this count. The maximum average number of chapters are present in ‘Organic chemistry’ followed by ‘Civil engineering’ and ‘Educational leadership’. The number of chapters identified are heavily dependent on the correctness of the ITCore segmentation tool.

4.1.5 Number of Words in Each Chapter for Each Subject Category

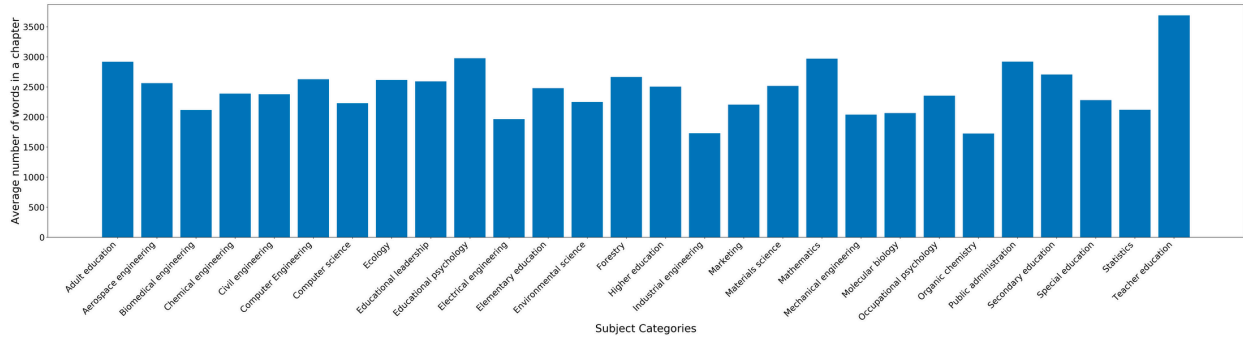


Figure 4.12: Average number of words present in chapters of ETDs for each subject category

In this section, we present the average number of words that are present in the chapters of an ETD for a particular subject category. Figure 4.12 illustrates this count. The maximum average number of words in the full-text are present in ‘Teacher education’ with 3689 words, followed by ‘Educational psychology’ with 2976 words, followed by ‘Mathematics’ with 2970 words, and ‘Public administration’ with 2920 words.

Chapter 5

Chapter Segmentation

In this chapter, we first perform a comparison of the various tools described in the Review of Literature chapter and then describe in depth the techniques that were most suited for this task. In order to get our chapters, we first needed to divide the full-text into chapter segments and then extract the text information from these segments. It is fairly challenging to perform this segmentation into chapters, since PDFs do not store documents so that the hierarchical organization of text is easy to determine.

5.1 Comparison of Text Extraction Tools from PDFs

By way of example, we considered an ETD titled ‘The Social Security Retirement Decision: Maximizing Expected Discounted Worth’ [68]. While a number of ETDs across different subject categories were randomly sampled and the results of the tool were manually inspected to determine the best tool for our purposes, we use this ETD to represent the performance of the various text extraction tools. Figure 5.1 is a snippet of the ‘Introduction’ chapter of this sample ETD. We compare each of the tools using this PDF and attempt to identify the tool that is best suited for our purposes.

1. INTRODUCTION

The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.

Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive \$730 billion in Social Security benefits [2, 3].

SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. *Retirement benefits* can be started between ages 62 and 70 and are based on the individual's earning history. The *primary insurance amount* (PIA), is the amount received when retirement benefits are started at the worker's specified *full retirement age* (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as *delayed retirement credits* are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no

Figure 5.1: Snippet from the 'Introduction' chapter of a sample ETD [68]

1. INTRODUCTION

The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.

Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive \$730 billion in Social Security benefits [2, 3]. SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. Retirement benefits can be started between ages 62 and 70 and are based on the individual's earning history. The primary insurance amount (PIA), is the amount received when retirement benefits are started at the worker's specified full retirement age (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as delayed retirement credits are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no

Figure 5.2: Snippet of the HTML generated page of the 'Introduction' chapter of a sample ETD using PDFMiner

5.1.1 PDFMiner

As mentioned in Section 3.1.1, PDFMiner has the ability to convert PDFs into different file formats including HTML and XML. We utilized the ‘pdf2txt.py’ command line tool to convert our sample PDF into HTML structure.

Figure 5.2 illustrates a sample of the HTML generated version of the PDF. Each of the lines generated on this page is written using various HTML tags. The text content has been enclosed within ‘span’ tags and contains different style attributes in order to achieve the layout that has been indicated in Figure 5.3. As illustrated by this image, the markup is fairly difficult for a human to read since each line is surrounded by HTML tags.

```

<div style="position:absolute; top:9312px;"><a name="12">Page 12</a></div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:534px; top:9349px; width:5px; height:11px;"><span
style="font-family: b'UCMZET+CMR12'; font-size:11px">1
<br></span>
</div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:245px; top:9453px; width:151px; height:14px;"><span
style="font-family: b'SVXCCK+CMBX12'; font-size:14px">1. INTRODUCTION
<br></span>
</div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:108px; top:9487px; width:432px; height:11px;"><span
style="font-family: b'UCMZET+CMR12'; font-size:11px">The United States Social Security Administration (SSA), founded through the Social
<br></span>
</div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:108px; top:9509px; width:432px; height:11px;"><span
style="font-family: b'UCMZET+CMR12'; font-size:11px">Security Act of 1935, is an independent agency of the United States Federal Govern-
<br></span>
</div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:108px; top:9531px; width:432px; height:11px;"><span
style="font-family: b'UCMZET+CMR12'; font-size:11px">ment. The SSA administers Social Security, a social insurance program where workers
<br></span>
</div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:108px; top:9552px; width:432px; height:11px;"><span
style="font-family: b'UCMZET+CMR12'; font-size:11px">pay into Social Security through taxes and receive future bene(cid:12)ts based on
earnings
<br></span>
</div>
<div style="position:absolute; border: textbox 1px solid; writing-mode:lr-tb; left:108px; top:9574px; width:432px; height:11px;"><span
style="font-family: b'UCMZET+CMR12'; font-size:11px">history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age
and
<br></span>
</div>

```

Figure 5.3: HTML markup of the ‘Introduction’ chapter of a sample ETD using PDFMiner

We had initially considered utilizing the PDFMiner tool to segment the PDF into chapters as it has the ability to generate HTML structure of a page. We had planned to utilize the BeautifulSoup library in order to parse the markup and collect all of the text tags. However, on further analysis, we identified that the HTML generated by PDFMiner did not maintain a hierarchical structure and could not be utilized to obtain chapters from the full-text. A

possible work-around would have been to collect the various style attributes used in the HTML and then identify the most frequently used style parameters to obtain the text body, and use the larger font sizes specified as part of the style attributes to identify larger titles. This could have potentially been used in order to identify the hierarchy of the document. However, since this approach could lead to multiple incorrect cases, we continued to search for better means to segment the PDF into chapters and have hence grouped PDFMiner under the bracket of tools used to extract text from PDFs.

5.1.2 PyPDF2

As described in Section 3.1.1, PyPDF2 can extract information from PDFs, page by page. We utilized the PdfFileReader class to load the PDF file object and then used the ‘getPage()’ method to select a specific page from the PDF. We ran a loop over the total number of pages present in the PDF in order to get each of the pages. The ‘extractText()’ method was used in order to extract text from the sample PDF shown in Figure 5.1.

Figure 5.4 shows the result obtained after using the ‘extractText()’ method to obtain the textual data from the ‘Introduction’ chapter. As indicated by the figure, the results obtained are fairly poor and many of the words have been merged together. This led us to conclude that this tool was not suitable to extract text from the ETD.

5.1.3 PyMuPDF

PyMuPDF 3.1.1 has the ability to extract text content from PDFs as well as to get meta-information such as Table of Contents using the ‘getToC()’ method.

Figure 5.5 shows the results obtained using the ‘getText()’ on a specific page of the sample

```

1. INTRODUCTION
TheUnitedStatesSocialSecurityAdministration(SSA),foundedthroughtheSocial
SecurityActof1935,isanindependentagencyoftheUnitedStatesFederalGovern-
ment.TheSSAadministersSocialSecurity,asocialinsuranceprogramwhereworkers
payintoSocialSecuritythroughtaxesandreceivefuturebbasedonearnings
history[1].Throughoutthethesis,SocialSecurityreferstotheFederalOld-Ageand
SurvivorsInsuranceprograms,whichprovidetoretiredworkersandspouses,
aswellassurvivingspouses.
SocialSecurity(SSB)formanimportantcomponentofretirementplan-
ningforthoselivingintheUnitedStates.AmongSocialSecuritybage60
andolder,54%ofmarriedcouplesand73%ofunmarriedpersonsreceiveatleast50%
oftheirincomefromSocialSecurity.In2011,over54millionAmericanswillreceive
$730billioninSocialSecurityb[2,3].
SSBareaseitherretirement,spousal,orsurvivorbandarede-
terminedfromearningshistoryandtheagethatthebarestarted.
Retirement
b
canbestartedbetweenages62and70andarebasedontheindividual's
earninghistory.The
primaryinsuranceamount
(PIA),istheamountreceivedwhen
retirementbarestartedattheworker'ssp
fullretirementage
(FRA),
whichisdependentondateofbirth.ThePIAisdeterminedfromthehighest35
yearsoftheworker'searnings.Althoughretirementbcanbetakenasearly
asage62,theyarereducedforeachmonthtakenearly,accordingtothemonthly
reductionfactor.DelayingretirementbpastFRAincreasesthebeach
monthas
delayedretirementcredits
areaccumulated,reachingthemaximumatage
70.Althoughretirementbcanbedelayedpastage70,delayedretirement
creditsnolongeraccumulate,sobdonotincreasefurther.Assuch,thereisno

```

Figure 5.4: Snippet of the ‘Introduction’ chapter of a sample ETD using PyPDF2

PDF. Our initial thought was to leverage the Table of Contents information and then attempt to identify the different chapters by using the data provided by the ToC. A possible strategy to achieve this would have been to use PyMuPDF to obtain the ToC along with PDFMiner (the HTML format of the document) to select chapter headings from the ToC and then use that information to identify the exact style attributes used for headings in a specific ETD.

However, when we performed further analysis, we found that there was a relatively low number of PDFs that contained ToC information. Thus, we attempted to find better tools to perform the segmentation into chapters.

1. INTRODUCTION

The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.

Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive \$730 billion in Social Security benefits [2,3].

SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. Retirement benefits can be started between ages 62 and 70 and are based on the individual's earning history. The primary insurance amount (PIA), is the amount received when retirement benefits are started at the worker's specified full retirement age (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as delayed retirement credits are accumulated, reaching the maximum at age 70.

Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no

Figure 5.5: Snippet of the 'Introduction' chapter of a sample ETD using PyMuPDF

5.1.4 Pdftotext

Pdftotext [3.1.1](#) has the ability to extract text data page by page.

It stores the data from all the pages in a list. We can specify the index of a specific page in order to get the text data from that page.

5.1.5 ABBYY Cloud OCR SDK

ABBYY Cloud OCR SDK as mentioned in Section [3.1.1](#), is a paid web based OCR service.

To utilize this tool, we performed the following steps.

1. *Register on the ABBYY Cloud OCR SDK website*

1. INTRODUCTION

The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.

Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive \$730 billion in Social Security benefits [2, 3].

SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. Retirement benefits can be started between ages 62 and 70 and are based on the individual's earning history. The primary insurance amount (PIA), is the amount received when retirement benefits are started at the worker's specified full retirement age (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as delayed retirement credits are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no

Figure 5.6: Snippet of the 'Introduction' chapter of a sample ETD using pdftotext

Here, we utilized the 'Free Trial' option given by the website. This account permitted us to process a total of '1660 pages'.

2. *Create an application*

This provided an Application ID and Application Password that was used in further steps.

3. *Clone and compile GitHub repository*

In this step, we cloned the GitHub repository <https://github.com/abbyy/cloudsdk-demo-dotnet> and then compiled the source code.

4. *Use CLI to convert to DOCX*

We follow the guidelines given in <https://www.ocrsdk.com/documentation/quick-start-guide/python-ocr-sdk/>.

(a) *Customize the URL for HTML requests*

To perform this step, we modify the file 'AbbyyOnlineSdk.py'. Here, we modify

the `ServerUrl` according to the location center. In our case this becomes the following.

```
ServerUrl = "http://cloud-westus.ocrsdk.com/"
```

(b) *Set Environmental Variables*

In this step, we set and export two environmental variables: `ABBYY_APPID` to our Application ID, and `ABBYY_PWD` to the Application Password.

(c) *Convert PDF file to DOCX* To perform this step, we run the following command

```
python process.py -docx | -txt  
/Users/palakhjude/Desktop/1501927.pdf  
/Users/palakhjude/Desktop/1501927.docx
```

Our initial plan was to use this tool to identify structural information from the PDF and thereby segment it into chapters. To perform this, we considered converting the PDF into DOCX format and then get the internal XML structure of the DOCX file to identify the hierarchy of the PDF file. Figure 5.7 illustrates the DOCX page generated for the ‘Introduction’ chapter. However, on further investigation, we noticed that this XML is not always well formed and therefore continued to look for other segmentation tools.

This tool also enables extraction of the text from a PDF in `.txt` format. Figure 5.8 illustrates the `.txt` page generated for the ‘Introduction’ chapter. While the text extracted is good, this tool is paid and the trial version allows a limited number of pages. Thus, we decided against using this for text extraction.

1. INTRODUCTION

The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.

Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive \$730 billion in Social Security benefits [2,3].

SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. *Retirement benefits* can be started between ages 62 and 70 and are based on the individual's earning history. The *primary insurance amount* (PIA), is the amount received when retirement benefits are started at the worker's specified *full retirement age* (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as *delayed retirement credits* are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no advantage to delay retirement benefits beyond age 70. *Spousal benefits* can be started between ages 62 and 70 as long the individual's spouse has started retirement benefits.

Figure 5.7: Snippet of the DOCX generated for the 'Introduction' chapter of a sample ETD using ABBYY Cloud OCR SDK

```
1
1. INTRODUCTION
The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.
Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive $730 billion in Social Security benefits [2,3].
SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. Retirement benefits can be started between ages 62 and 70 and are based on the individual's earning history. The primary insurance amount (PIA), is the amount received when retirement benefits are started at the worker's specified full retirement age (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as delayed retirement credits are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no
```

Figure 5.8: Snippet of the .txt generated for the 'Introduction' chapter of a sample ETD using ABBYY Cloud OCR SDK

5.1.6 Textract

As mentioned in Section 3.1.1, this tool offers a command line interface as well as the ability to extract text using the Python package. For our evaluation of this tool, we utilize the Python package.

1. INTRODUCTION

The United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses. Social Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive \$730 billion in Social Security benefits [2, 3]. SSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. Retirement benefits can be started between ages 62 and 70 and are based on the individual's earnings history. The primary insurance amount (PIA), is the amount received when retirement benefits are started at the worker's specified full retirement age (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as delayed retirement credits are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no

Figure 5.9: Snippet of the 'Introduction' chapter of a sample ETD using `textextract`

Figure 5.9 illustrates the result obtained after extracting text from the sample PDF. This tool has a very similar aim as compared to Apache Tika. The tool doesn't provide hierarchical information about the document layout.

We used the `process` method from the `textract` Python package. The method accepts a number of parameters including a mandatory parameter giving the path of the PDF file. It also accepts optional parameters such as 'method' which enables a user to specify a particular method to be used to parse the PDF. It contains other optional parameters such as 'encoding' and 'extension' which allow users to specify the output encoding and the file's extension if the filename does not explicitly state the name of the extension. It also allows users to specify the 'language' for other language PDFs and internally uses Tesseract OCR to extract the text.

5.1.7 Tika-Python

As mentioned in Section 3.1.1, Tika-Python provides the ability to extract text from PDFs utilizing the `parser` interface.

Since this Python library is a Python binding to the Apache Tika REST services which are in Java, it requires Java 7+ to be installed on a user's system. This is required since the Tika-Python library needs to run the Tika REST server in the background.

This `parser` interface also has the ability to output the content in the form of XHTML. However, it did not maintain hierarchical information of the document, thus we did not consider this tool for the purpose of segmenting the PDF into chapters.

```
1. INTRODUCTION\n\nThe United States Social Security Administration (SSA), founded through the Social Security Act of 1935, is an independent agency of the United States Federal Government. The SSA administers Social Security, a social insurance program where workers pay into Social Security through taxes and receive future benefits based on earnings history [1]. Throughout the thesis, Social Security refers to the Federal Old-Age and Survivors Insurance programs, which provide benefits to retired workers and spouses, as well as surviving spouses.\n\nSocial Security Benefits (SSB) form an important component of retirement planning for those living in the United States. Among Social Security beneficiaries age 60 and older, 54% of married couples and 73% of unmarried persons receive at least 50% of their income from Social Security. In 2011, over 54 million Americans will receive $730 billion in Social Security benefits [2, 3].\n\nSSB are classified as either retirement, spousal, or survivor benefits, and are determined from earnings history and the age that the benefits are started. Retirement benefits can be started between ages 62 and 70 and are based on the individual's earning history. The primary insurance amount (PIA), is the amount received when retirement benefits are started at the worker's specified full retirement age (FRA), which is dependent on date of birth. The PIA is determined from the highest 35 years of the worker's earnings. Although retirement benefits can be taken as early as age 62, they are reduced for each month taken early, according to the monthly reduction factor. Delaying retirement benefits past FRA increases the benefit each month as delayed retirement credits are accumulated, reaching the maximum at age 70. Although retirement benefits can be delayed past age 70, delayed retirement credits no longer accumulate, so benefits do not increase further. As such, there is no
```

Figure 5.10: Snippet of the 'Introduction' chapter of a sample ETD using Tika-Python

Figure 5.10 illustrates the result obtained after extracting text from the sample PDF. After testing the Tika-Python library on multiple PDFs, we found that it gave consistently good results. Thus, we make use of this library for the purpose of extracting text from the ETDs.

5.2 Comparison of Segmentation Tools

For the purpose of this research, it was necessary for us to identify a tool that would enable us to segment our full-text ETD document into chapters. Considering the varied styles that authors employ while drafting ETDs, this task is non-trivial. Older heuristic based methods attempted to identify the boundaries of chapters by utilizing font information such as font style and font size. However, these methods do not work in all cases.

In this section, we evaluate the performance of two such tools. The first is Grobid, which is a machine learning library, while the second is ITCore (Intelligent Textbooks Core) that has been developed in Java.

5.2.1 Grobid

As mentioned in Section 3.1.2, Grobid is a machine learning library that extracts and parses PDFs into XML/TEI documents. The output XML generated should maintain the hierarchy of the document.

As part of the project for the course *CS6604: Digital Libraries* [6], we extracted and parsed data from 7033/13071 dissertations and 11674/17891 theses from the Virginia Tech collection.

As part of a post-processing step, we identified the boundaries of the chapters from the XML/TEI documents that were generated by Grobid. Figure 5.11 indicates the number of

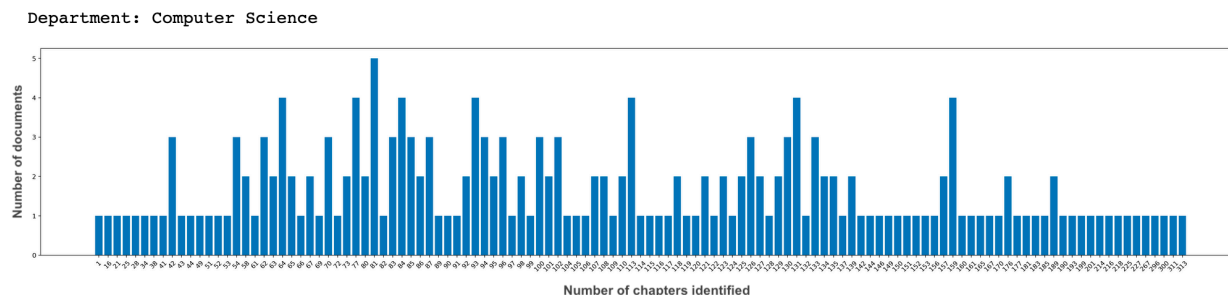


Figure 5.11: Number of chapters identified for the Computer Science department using Grobid

chapters identified for the Computer Science department from the Virginia Tech collection. The x-axis indicates the number of chapters and the y-axis indicates the number of documents. As seen, the number of chapters extracted range from 1 to 313. A similar range of chapters were extracted across all the various departments that exist in the Virginia Tech ETD corpus. This indicated to us that Grobid was not the best tool for this task as it segmented the documents into much smaller chunks than chapters.

5.2.2 ITCore (Intelligent Textbooks Core)

ITCore (Intelligent Textbooks Core) [5] is a Java based tool that uses information from the Table of Contents to segment the full-text PDF into small segments which are also stored as PDFs.

The authors of the repository provide a web interface <https://intextbooks.science.uu.nl/> that can be used to quickly test the working of this tool. However, this web interface only returns the tei.xml file and does not give us the segmented PDF files. The link to send the resultant tei.xml file is emailed to the address provided by the user.

We cloned the code base from the GitHub repository <https://github.com/intextbooks/ITCore>. In order to run this code base locally, we had to ensure that we had Java and MySQL set

Contents

Abstract	ii
Acknowledgments	iv
List of Tables	xi
List of Figures	xiii
1 Introduction	1
1.1 Problems of Interest	1
1.1.1 Robotic Minimally Invasive Surgery (RMIS)	1
1.1.2 Temporal Textures in Video	3
1.2 Problems We are Trying to Solve	4
1.3 Organization	5
1.4 Contributions of this Dissertation	12
2 Hidden Markov Models	14
2.1 Probabilistic Markov Models	14

CONTENTS

2.2 Emitting Probabilistic Markov models	15
2.3 Hidden Markov Model	16
2.4 Inference in a HMM	17
2.4.1 Inferring the Distribution of $s_{1:N}$	17
Forward Pass	19
Backward Pass	19
2.4.2 Inferring the most Probable State Sequence $s_{1:N}$	20
2.5 Parameter Estimation for HMM ; the Baum Welch Algorithm	22
2.6 Chapter Summary	27
3 Factor Analyzed Hidden Markov Models	28
3.1 Probabilistic Principal Component Analysis	29
3.1.1 Inference in PPCA	30
Efficient Inference in PPCA	31
3.1.2 Maximum Likelihood Estimation of Θ via EM	34
3.2 Factor Analyzed HMMs	39
3.2.1 Inferring the joint distribution of $s_{1:N}$ and $x_{1:N}$	39
3.2.2 Learning the Parameters of a FA-HMM via EM	40
3.2.3 Tied Estimation of the Loading Matrix : $H^t = H$	45
3.2.4 Tied Estimation of the Loading Parameters : $H^t = H$	46
3.3 Connection to LDA, HLDA and Semi-Tied Covariance Modeling	47
3.3.1 EM Estimation of the Parameters of HLDA	48
3.4 Chapter Summary	50

vi

vii

Figure 5.12: Table of Contents of the sample ETD titled ‘Learning And Inference Algorithms for Dynamical System Models Of Dextrous Motion’ [69]

up on our system. We ran the `intextbooks_db.sql` SQL file to generate the database and table schema as per the requirements of the project. The code base is a Maven Project. We ensured that all the dependencies were downloaded appropriately on our local system. It was also necessary for us to add the ‘`stanford-corenlp-3.9.2-models.jar`’ file to our classpath in addition to the other dependencies present in the `pom.xml` file.

The default implementation leverages the information present in the Table of Contents using a rule-based approach with the aim of identifying potential segments from the PDF. This approach was developed with the intention of segmenting chunks from books. The output of this tool is a number PDFs of each of the segments that have been identified. We tested this tool on the sample ETD utilized to compare the extraction techniques in Section 5.1, however, the tool was unable to process this PDF. Thus, we selected another ETD titled ‘Learning And Inference Algorithms for Dynamical System Models Of Dextrous Motion’ [69]

```

<body>
  <div type="contents">
    <list>
      <item>1 introduction<ref target="seg_1">1</ref>
      </item>
      <list>
        <item>1.1 problems of interest<ref target="seg_3">1</ref>
        </item>
        <list>
          <item>1.1.1 robotic minimally invasive surgery (rmis)<ref
            ↪ target="seg_5">1</ref>
          </item>
          <item>1.1.2 temporal textures in video<ref
            ↪ target="seg_7">3</ref>
          </item>
        </list>
        <item>1.2 problems we are trying to solve<ref
          ↪ target="seg_9">4</ref>
        </item>
        <item>1.3 organization<ref target="seg_11">5</ref>
        </item>
        <item>1.4 contributions of this dissertation<ref
          ↪ target="seg_13">12</ref>
        </item>
      </list>
    </list>
  </div>
</body>

```

Listing 1: Snippet of the TEI XML file as generated by ITCore for Chapter 1

to evaluate the performance of this tool.

Figure 5.12 illustrates the first two pages of the Table of Contents of our sample PDF. As is visible in this image, there are three chapters on these pages: ‘Introduction’, ‘Hidden Markov Models’, and ‘Factor Analyzed Hidden Markov Models’. Each of these chapters contains sections and sub-sections that have been indicated by indentations in the Table of Contents.

As indicated by Listing 1 and Figure 5.12, the tool accurately identified the hierarchy of the chapter and its sections. However, as per the default implementation, it considered each of the identified elements present in the TEI/XML file to be a new segment. Thus, it generated multiple PDFs (some with overlapping content in cases where the new section did not appear on a new page), one for each of the segments present in the TEI/XML file. This tool identified segments that are more granular than chapter-level segments.

Considering that for the purpose of our research we are required to segment the ETD at the chapter level, we inspected the code in an attempt to identify a way to change this level of granularity. We found that the method `performCleanPDFsplitHelper` was responsible for determining the level of the segment. Thus, we modified the code to ensure that it proceeds further to generate the segmented PDFs only if this level is less than or equal to 1. While this did not modify the resultant TEI/XML file, it ensured that the segmented PDF files generated were restricted to only those segments that appeared at the topmost level of the hierarchy. We hypothesize that most well formatted table of contents pages would ensure that only the chapter names appear at the topmost level.

Since we were dealing with thousands of ETDs, we decided to create a shell script that would loop over a given directory and generate segmented chapter PDFs for each for the ETDs present in that directory; see Listing 2.

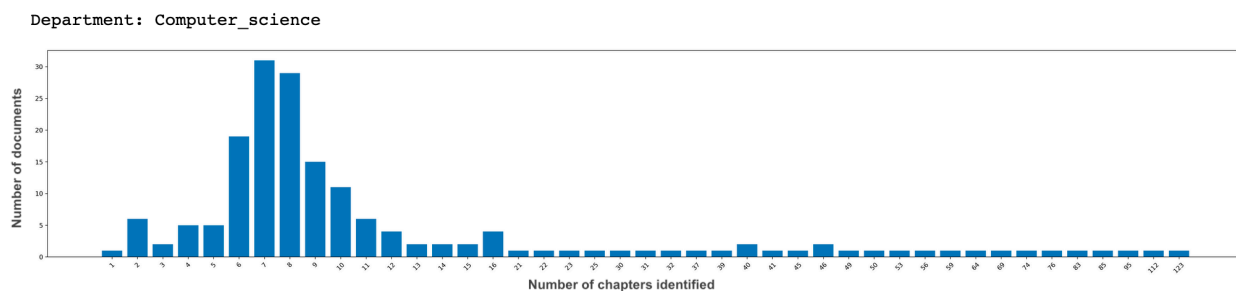


Figure 5.13: Number of chapters identified for the Computer Science subject category using IT Core

```
#!/bin/bash

BASE_DIR="/Users/palakhjude/Desktop/PQDT_Full_Text_Data/${1}"

BASE_OUT_DIR="/Users/palakhjude/Desktop/PQDT_Segments/${1}"

dirlist=(`find "$BASE_DIR" -type f -name "*.pdf"`)

for FILENAME in "${dirlist[@]}"
do
    echo "$FILENAME"
    java -Dfile.encoding=UTF-8 -classpath $FILENAME $BASE_OUT_DIR

    sleep 5s
done
```

Listing 2: Shell script to iterate over all folders in a directory and run the ITCore tool

We ran this script for all of the files in our PQDT data collection. Since this tool uses a rule-based approach to identify chapters and is heavily dependent on the table of contents, a large number of our ETD files were skipped. Some of the documents were skipped due to a lack of page numbers, some due to a missing/inappropriate Table of Contents page, and some due to a Table of Contents page that did not contain the correct title as prescribed by the list of titles that could be present such as ‘Contents’, ‘Index’, ‘Table of Contents’, etc. However, given that segmentation is a complex research problem in its own right, for the purpose of this research, we utilize the ones that were extracted as part of this tool.

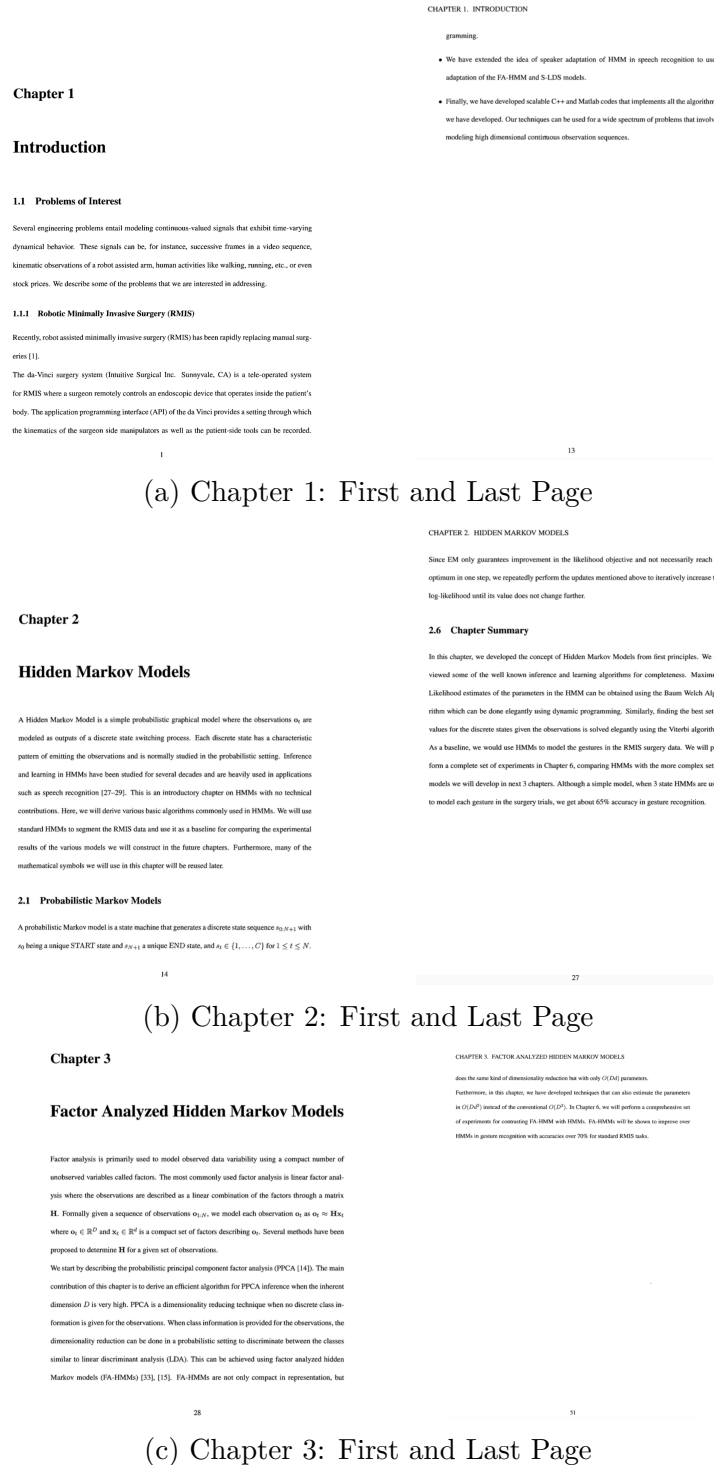


Figure 5.14: First and last pages of the chapter segmented PDFs of the sample ETD titled ‘Learning And Inference Algorithms for Dynamical System Models Of Dextrous Motion’ [69] using the ITCore tool

Figure 5.13 illustrates the number of chapters extracted for the ‘Computer Science’ subject category from the PQDT dataset. The x-axis indicates the number of chapters while the y-axis indicates the number of documents. As can be seen, the number of chapters extracted range from 1 to 123. However, if we look closely, the bulk of the number of chapters extracted is between 6-10 chapters. This is a reasonable number and thus, we decided to utilize this tool for our chapter segmentation.

Figure 5.14 illustrates the first and last pages of the first three chapters as identified and segmented by this tool. As is seen here, the performance of the tool is good when the Table of Contents is well formatted as is the case with this sample ETD. A randomly selected manual inspection of multiple PDFs indicated that the quality of the other extractions was also fairly good.

Chapter 6

Summarization

As part of some of our initial experiments, we utilized the full-text data from the Electronic Theses and Dissertations to train our models. However, some of the documents are very long. This could lead to a loss of important information when we generate lower dimensional Doc2Vec data. Thus, we wanted to evaluate the performance of the classification model given a summarized version of the full-text data.

To generate these summaries, we summarize each of the chapters that were segmented from the ETD and then concatenate these summaries. We do not summarize the full-text as a whole since each chapter may contain varied information and we wanted to get the key aspects from each of the chapters to ensure that all of the key concepts from the ETD as a whole were represented.

To generate these summaries we use the summarization implementations provided by `gensim` [54] and `sumy` [40]. As discussed in Section 3.2.1, we generate extractive summaries using TextRank using the implementation from `gensim`. Similarly, other types of extractive summaries – from LexRank, Latent Semantic Analysis, and Luhn’s Algorithm – were generated using the `sumy` library.

In this chapter, we present the results obtained for each of these techniques. We consider a sample ETD titled ‘Beech Bark Disease: The Relationship between Scale and Neonectria Lesion Densities in an Aftermath Forest’ [27].

CHAPTER 1: INTRODUCTION AND HYPOTHESES

At the turn of the last century, American beech was faced with a foreign disease complex from Europe, which has left heavy beech mortality in its wake. Beech bark disease (BBD) is a disease complex that affects American beech throughout much of its range in the United States and Canada. Initial studies by Ehrlich (1934) in the early decades of the 20th century in northeastern forests led to the hypothesis that the disease is caused by the interaction between the beech scale (*Cryptococcus fagisuga* Lindinger) and ascomycetous fungi in the genus *Neonectria*. Ehrlich's work in 1934 and Houston's work in the 1970's and 1980's have long been the basis for the hypothesis that *Neonectria* infection is dependent upon predisposing scale infestation. However, work by Parker, Lonsdale, Perrin and others in the early 1980s suggests that other factors can lead to *Neonectria* infection, even in the absence of scale (Parker, 1977; Lonsdale, 1980; Perrin, 1980). These later reports as well as my own observations in the aftermath forests of Central New York have lead me to question the classic hypothesis of scale predisposition followed by *Neonectria* infection. It is common to find trees with evidence of long-term scale infestation without subsequent fungal infection. It is also common to find *Neonectria*-infected trees without scale colonization. These inconsistencies in the purported causation of BBD have prompted this research study.

In order to examine the relationship between disease agents, I used image analysis to measure scale densities and track the development of *Neonectria* lesions. I also attempted to reduce scale populations on treatment trees through the use of a systemic insecticide. Scale densities were compared to track changes within treatments and lesion formation was measured to determine any relationships to previous and current scale densities. Two treatment groups, a one-year injection and a two-year injection with ImicideTM (a systemic insecticide), were

developed to study effective residual times of the insecticide treatment, and tree diameters were recorded to observe interactions with scale density. This study is the first to my knowledge to use a quantitative, as opposed to a qualitative, method to measure scale and *Neonectria* population densities on beech bark to study the relationship between them. I will not attempt to explain the cause of each agent, but will quantify their occurrence with one another.

My objectives are:

1. To determine the effectiveness of systemic Imicide application upon beech scale population density.
2. To compare different treatment times to determine the effective residual time of Imicide upon scale.
3. To observe the relationship between host tree diameter and scale population density.
4. To observe the effect of scale presence on future lesion density.
5. To determine the relationship between beech scale and *Neonectria* population densities.

My hypotheses are:

1. Imidacloprid treatment will significantly reduce scale density of infested beech trees.
2. Annual re-injection of imidacloprid is necessary to maintain significantly reduced scale density.
3. Current scale density is significantly related to host tree diameter.
4. Scale presence will have no significant affect on future lesion formation.
5. Scale density and lesion presence are not directly correlated in an aftermath forest.

For each of the methods mentioned in this section, we generate summaries separately and store to disk each of the summaries across all of the various techniques. The concatenation of the summaries is performed as part of a later step in the pipeline.

6.1 `TextRank`

In this section, we present the summaries generated for the sample ETD by changing different parameters such as the word count and ratio of text to be kept while generating the summary. We also present the results obtained from the top set of keywords used by the `TextRank` algorithm.

6.1.1 Summaries

We use the summarization methods from `gensim`'s summarizer package. The parameters that this method accepts include:

1. **text**: This parameter accepts the text to be summarized.
2. **ratio**: This parameter accepts a number between 0 and 1. It represents the proportion of the text sentences from the original text that should be retained for the summary. The default value is 0.2. We experiment with this default value and with a value of 0.5.
3. **split**: This parameter accepts a Boolean value. If True, it returns a list of sentences, otherwise, joined strings are returned.
4. **word_count**: This parameter represents the number of words to be present in the output summary. If this parameter is provided along with the ratio value, the ratio

value will be ignored.

Figure 6.3 displays the summary generated using the default parameter values of the summarizer. The default parameters include a ratio of 0.2, split value of False, and no word count provided.

for the hypothesis that *Neonectria* infection is dependent upon predisposing scale infestation. factors can lead to *Neonectria* infection, even in the absence of scale (Parker, 1977; Lonsdale, measure scale densities and track the development of *Neonectria* lesions. reduce scale populations on treatment trees through the use of a systemic insecticide. densities were compared to track changes within treatments and lesion formation was measured developed to study effective residual times of the insecticide treatment, and tree diameters were population densities on beech bark to study the relationship between them. 3. To observe the relationship between host tree diameter and scale population density. 4. To observe the effect of scale presence on future lesion density. 5. To determine the relationship between beech scale and *Neonectria* population densities. 1. Imidacloprid treatment will significantly reduce scale density of infested beech trees.

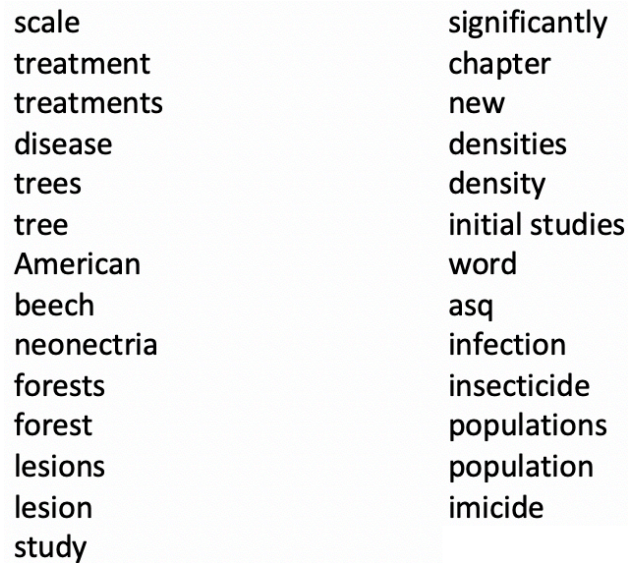
Figure 6.3: Summary of the sample ETD generated using TextRank with ratio=0.2

For this set of summaries, we set the word count to 100, split value to False, and have no ratio value. Figure 6.4 displays the summary generated using these parameters.

measure scale densities and track the development of *Neonectria* lesions. reduce scale populations on treatment trees through the use of a systemic insecticide. developed to study effective residual times of the insecticide treatment, and tree diameters were population densities on beech bark to study the relationship between them. 3. To observe the relationship between host tree diameter and scale population density. 4. To observe the effect of scale presence on future lesion density. 5. To determine the relationship between beech scale and *Neonectria* population densities. 1. Imidacloprid treatment will significantly reduce scale density of infested beech trees.

Figure 6.4: 100 word count summary of the sample ETD generated using TextRank

6.1.2 Keywords



scale	significantly
treatment	chapter
treatments	new
disease	densities
trees	density
tree	initial studies
American	word
beech	asq
neonectria	infection
forests	insecticide
forest	populations
lesions	population
lesion	imicide
study	

Figure 6.5: Keywords of the sample ETD generated using TextRank

We use the keywords method from `gensim`'s summarizer package. The parameters that this method accept include:

1. **text**: This parameter accepts the text for which keywords are to be generated.

Figure 6.5 represents the keywords generated using this method.

6.2 LexRank

We use the `LexRankSummarizer` from the `sumy.summarizers.lex_rank` module along with the `PlaintextParser` from the `sumy.parsers.plaintext` module and the `EnglishTokenizer` from the `sumy.nlp.tokenizers` module.

The LexRankSummarizer accepts the document text to be summarized as well as the number of sentences to be generated by the summarizer. For the purpose of these summaries, we set the number of sentences to 10.

Figure 6.6 represents the summary generated using this method.

northeastern forests led to the hypothesis that the disease is caused by the interaction between the However, work by Parker, Lonsdale, Perrin and others in the early 1980s suggests that other reduce scale populations on treatment trees through the use of a systemic insecticide. Scale developed to study effective residual times of the insecticide treatment, and tree diameters were This study is the first to my knowledge to population densities on beech bark to study the relationship between them. To determine the relationship between beech scale and Neonectria population densities. Imidacloprid treatment will significantly reduce scale density of infested beech trees. Scale density and lesion presence are not directly correlated in an aftermath forest.

Figure 6.6: Summary of the sample ETD generated using LexRank

6.3 Luhn's Algorithm

We use the LuhnSummarizer from the `sumy.summarizers.luhn` module along with the PlaintextParser from the `sumy.parsers.plaintext` module and the English Tokenizer from the `sumy.nlp.tokenizers` module.

The LuhnSummarizer accepts the document text to be summarized as well as the number of sentences to be generated by the summarizer. For the purpose of these summaries, we set the number of sentences to 10.

Figure 6.7 represents the summary generated using this method.

northeastern forests led to the hypothesis that the disease is caused by the interaction between the Ehrlich's work in 1934 and Houston's work in the 1970s and 1980s have long been the basis. However, work by Parker, Lonsdale, Perrin and others in the early 1980s suggests that other factors can reduce scale populations on treatment trees through the use of a systemic insecticide. We developed a study to observe effective residual times of the insecticide treatment, and tree diameters were measured to study population densities on beech bark to study the relationship between them. To observe the relationship between host tree diameter and scale population density. To observe the effect of scale presence on future lesion density. To determine the relationship between beech scale and *Neonectria* population densities. Imidacloprid treatment will significantly reduce scale density of infested beech trees.

Figure 6.7: Summary of the sample ETD generated using Luhn's Algorithm

6.4 Latent Semantic Analysis (LSA)

We use the `LsaSummarizer` from the `sumy.summarizers.lsa` module along with the `PlainTextParser` from the `sumy.parsers.plaintext` module and the `EnglishTokenizer` from the `sumy.nlp.tokenizers` module.

We extract two forms of summaries using this technique, one with stopwords and one without.

The `LsaSummarizer` accepts the document text to be summarized as well as the number of sentences to be generated by the summarizer. For the purpose of these summaries, we set the number of sentences to 10. This default setup does not make use of stopwords.

Figure 6.8 represents the summary generated using this method.

disease complex that affects American beech throughout much of its range in the United States beech scale (*Cryptococcus fagisuga* Lindinger) and ascomycetous fungi in the genus *Neonectria*. for the hypothesis that *Neonectria* infection is dependent upon predisposing scale infestation. However, work by Parker, Lonsdale, Perrin and others in the early 1980s suggests that other factors can lead to *Neonectria* infection, even in the absence of scale (Parker, 1977; Lonsdale, reduce scale populations on treatment trees through the use of a systemic insecticide. densities were compared to track changes within treatments and lesion formation was measured developed to study effective residual times of the insecticide treatment, and tree diameters were explain the cause of each agent, but will quantify their occurrence with one another. Scale density and lesion presence are not directly correlated in an aftermath forest.

Figure 6.8: Summary of the sample ETD generated using LSA

For the alternative approach that uses stopwords, we use the Stemmer from the `sumy.nlp.stemmers` module along with the `get_stop_words` method from the `sumy.utils` module.

Figure 6.9 represents the summary generated using this method.

At the turn of the last century, American beech was faced with a foreign disease complex disease complex that affects American beech throughout much of its range in the United States beech scale (*Cryptococcus fagisuga* Lindinger) and ascomycetous fungi in the genus *Neonectria*. factors can lead to *Neonectria* infection, even in the absence of scale (Parker, 1977; Lonsdale, of Central New York have lead me to question the classic hypothesis of scale predisposition developed to study effective residual times of the insecticide treatment, and tree diameters were To observe the relationship between host tree diameter and scale population density. Imidacloprid treatment will significantly reduce scale density of infested beech trees. Current scale density is significantly related to host tree diameter. Scale density and lesion presence are not directly correlated in an aftermath forest.

Figure 6.9: Summary of the sample ETD generated using LSA with stopwords

Chapter 7

Classification

In this chapter, we first discuss an overview of the system pipeline. This is followed by a discussion of the various embedding models that were generated for the classification tasks performed on the PQDT dataset. Next, we describe the different methods used to work with the multi-labeled nature of our classification task. We then discuss the Machine Learning baselines that were created as part of the *CS6604: Digital Libraries* [6] class project. The dataset used for this study comprised of ETDs only from Virginia Tech. We then discuss the Machine Learning and Deep Learning models that were trained using the PQDT dataset that is comprised of ETDs from various universities. We also generate summaries of the chapters of ETDs with the intention of combining the summaries and using the result as input to our models. These summaries were also embedded using the techniques discussed in Section 7.2. We conclude by describing the various evaluation metrics that will be used for different tasks.

7.1 System Design Overview

In this section, we describe the pipeline used to train and test our classifiers using the PQDT dataset. All of the relevant steps have been discussed in Section 7.4.

We divide the pipeline into two main processes:

1. PDF extraction and embedding process
2. Classification models training and testing process

7.1.1 PDF Extraction and Embedding Process

The first process of our pipeline involves segmenting the ETD into chapters, extracting data from the full-text and chapter ETDs, and embedding this data.

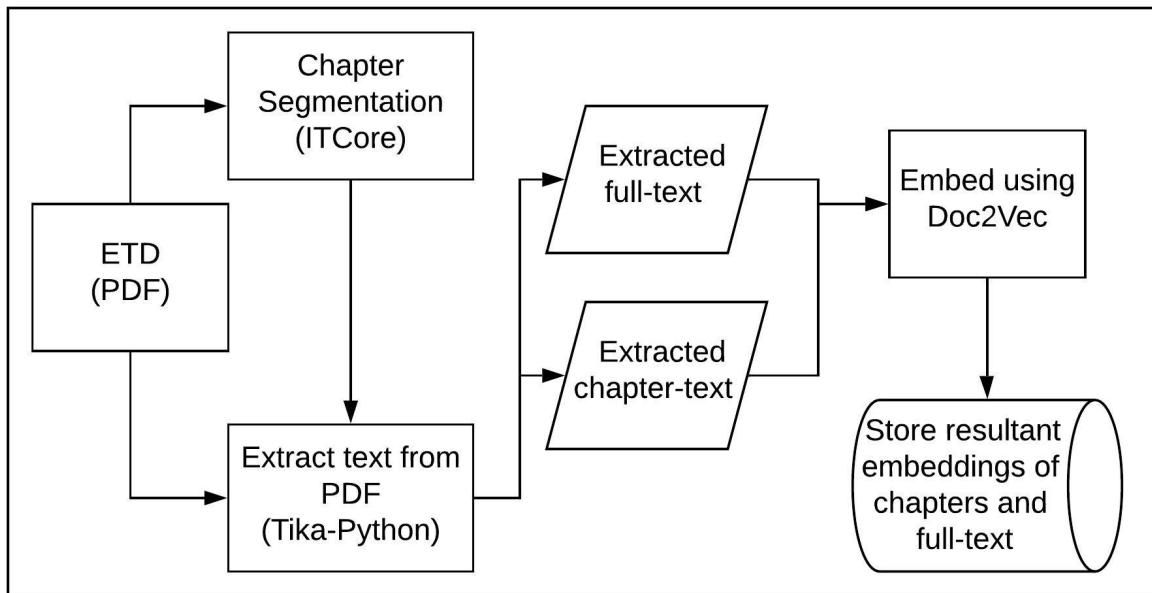


Figure 7.1: System overview: Extraction and embedding process

Figure 7.1 is a diagrammatic representation of this process. As indicated by this figure, given an ETD in PDF format, we obtain chapters from the ETD using the ITCore tool. We then extract the text from both the full-text ETD as well as the chapters using Tika-Python. Details about these methods have been discussed in depth in Chapter 5.

Once this text data is obtained, we embed this data using Doc2Vec, and the resultant

embeddings of both the chapters as well as the full-text are stored to disk. This data is retrieved in a later step while training or testing the classification model. We further describe our embedding process in Section 7.2.

7.1.2 Classification Models Training and Testing Process

This section describes the flow of our classification training and testing process.

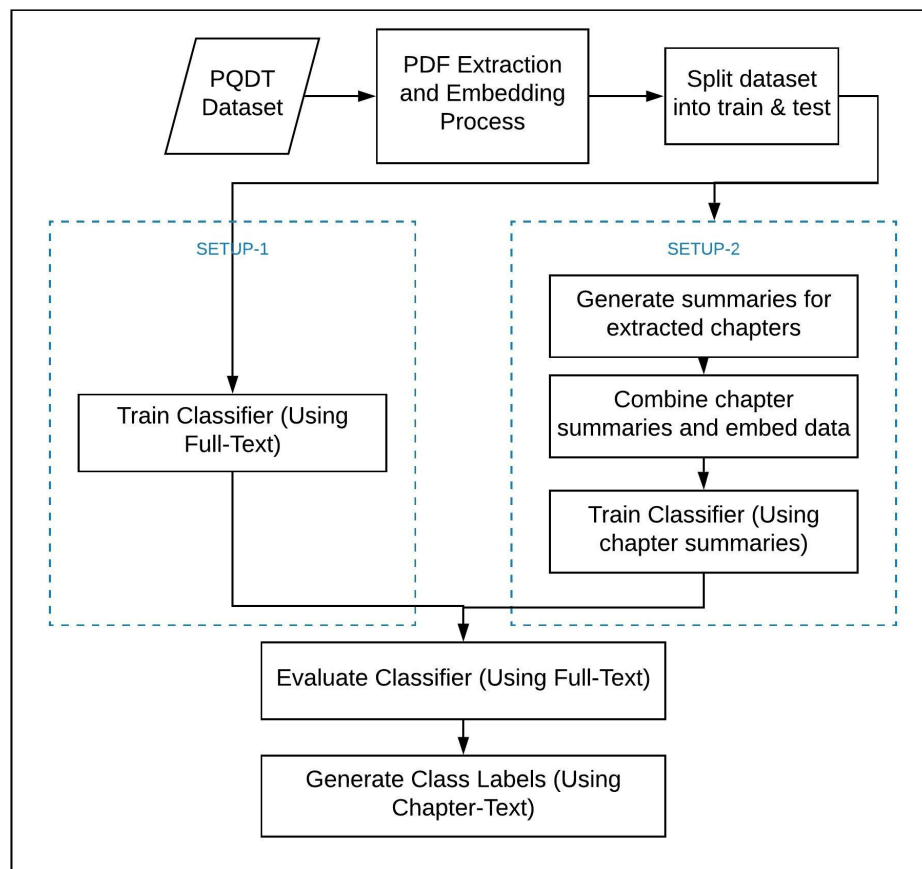


Figure 7.2: System overview: Classification workflow

Figure 7.2 gives a detailed representation of the sub-steps involved in this process. Given the entire PQDT Dataset, we first extract the full-text and chapter data from the PDF of

all of the ETDs and embed them as described in Section 7.1.1. We then split this data into train and test sets such that there is an 80-20% split in each subject category.

Our pipeline consists of two main experimental setups. The first setup uses the full-text data alone to train the classifiers. This is performed in two variants, one with 50% subset of randomly sampled ETDs (per subject category) and the other consisting of all the ETDs in our corpus. The second setup uses a smaller subset of the dataset. This subset only consists of ETDs for which chapters were successfully extracted with the ITCore tool.

Table 7.1: Counts of ETDs in train-test splits for the different experimental setups

Experimental Setup	Total ETDs	Train set	Test set
Setup 1: Full-text (All data)	9298	6964	2334
Setup 1: Full-text (Half data)	6458	4124	2334
Setup 2: Chapter subset data	4034	3009	1025

Table 7.1 lists the number of ETDs present in the train/test splits for each of the different setups. Since we randomly sample ETDs per subject category while generating the training dataset for **Setup 1: Full-text (Half data)**, the number of unique ETDs picked up as per our sampling causes the number of ETDs present in our halved dataset (4124) to be greater than half the total number of unique ETDs in the full dataset (3482). We keep the same test set throughout for Setup 1.

In the second setup, we generate summaries for each of the extracted chapters of the ETD and then combine them together. These combined summaries are then embedded and saved to disk. The saved embedded summaries are used to train the different classifiers.

In both of the setups, we test our classifier performance against the full-text ETD labels to determine the best models. We then use these best performing models to generate subject

category labels for chapters of ETDs that are present in the test set for each of the individual experimental setups.

7.2 Embeddings

In this section, we discuss the word and document embedding models we train as part of this research. Due to the nature of our ETD corpus which contains a vast amount of domain specific terms, we wanted to compare the performance of our classifiers with both pre-trained word/document embedding models and corpus specific re-trained word/document embedding models.

As discussed in Section 2.1.1, the fastText word embeddings [24] [7] have the best performance among the three word embeddings described. Thus, we utilize the fastText word embeddings to create vector representations of our keywords, title, and other metadata fields.

As indicated in Section 2.1.2, the word embedding models are not ideal for documents of longer length such as ETDs, since the order of words is not taken into consideration causing loss of semantic information. Thus, for longer fields such as full-text and abstract of the ETDs, we use Doc2Vec [26] embeddings.

For training these embeddings, we use the Penn State University dataset, the University of Illinois at Urbana-Champaign dataset, and the Virginia Tech ETD dataset. The pre-processing needed for fastText, as well as Doc2Vec, differs; we illustrate the approaches used for both.

With an intent to compare the performance of our classifiers with different data parts, we divide the PQDT dataset into three different segments. The data from each of these subsets is embedded using either the pre-trained or re-trained fastText or Doc2Vec embeddings before

being utilized by the classifiers. These subsets of the PQDT dataset include:

1. All of the full-text data from the PQDT dataset
2. Randomly sampled 50% of the training dataset that was formed for the previous subset of full-text data from the PQDT dataset
3. A subset of the original dataset that has chapter segments that were identified by the ITCore tool as described in 5.2.2. Additionally, we perform experiments using the chapter summaries combined together for the chapters in this subset.

7.2.1 Data Pre-processing

Here, we discuss the data pre-processing required before the fastText and Doc2Vec models are trained on our corpus.

FastText

The full-text extracted using *Tika-Python* contained extra characters that are not needed while training the fastText word embeddings. Thus, we use regular expressions in order to strip the text of these unwanted characters. The characters that are removed include non-alphabet and non-newline characters. In addition to this, we convert all of the text to lower case. The output of this step is saved to disk and utilized to train the fastText model.

Doc2Vec

Here we utilize `gensim`'s 'simple_preprocess' method from the `utils` package. It tokenizes each given sentence, converts it to lower case, and returns a list of strings as the final result.

The output of this step is saved to disk and utilized to train the Doc2Vec model.

7.2.2 Embedding Parameters

In order to help us determine the most suitable set of parameters for our classification task, we train multiple sets of fastText and Doc2Vec embeddings by varying the parameters. We divide the data into three groups:

1. Virginia Tech ETDs
2. ETDs from Penn State and University of Illinois at Urbana-Champaign
3. ETDs from Penn State, Virginia Tech, and University of Illinois at Urbana-Champaign

FastText

Table 7.2: Set of parameter variations for fastText

Parameter Name	List of values
Epochs	25, 50, 100
Word N Grams	3, 5
Dimensions	100, 200

Each set of parameters in Table 7.2 is used on each dataset permutation mentioned earlier for a total of 3 x 12 models.

Doc2Vec

Each set of parameters in Table 7.3 is used on each dataset permutation mentioned earlier for a total of 3 x 12 models.

Table 7.3: Set of parameter variations for Doc2Vec

Parameter Name	List of values
Epochs	25, 50, 100
DM	0, 1
Dimension	100, 200

7.3 Multi-Label Classification

In the case of text classification, a single document can belong to multiple categories at once. In the Virginia Tech ETD collection as well as the PQDT dataset, a single thesis/dissertation can possibly belong to multiple ProQuest subject categories at once. For example, a dissertation can be classified under ‘Statistics’ as well as ‘Computer Science’. Thus, the classification problem that we attempt to solve as part of this research extends beyond multi-class classification wherein a single document can only belong to a single class-label. Most of the traditional Machine Learning algorithms have been built for single-label classification problems. Thus, some of the techniques used to solve the multi-label classification problem involve converting this multi-label problem into multiple single-label problems and then utilizing the existing single-label algorithms. For this research, we use the BinaryRelevance and LabelPowerset approaches for the Virginia Tech dataset, and the LabelPowerset approach for the PQDT dataset. Details about each of these techniques is described in Section 2.2.

7.4 Virginia Tech Dataset

In this section, we discuss the methodology used to train different Machine Learning classifiers as part of the course project for *CS6604: Digital Libraries* [6]. The methods used here form the basis for further experiments performed.

Metadata and MongoDB

We represent an individual ETD exported from the Virginia Tech repository as a folder consisting of several content and metadata files.

The example ETD represented by Figure 7.3 contains two descriptive metadata files: `dublin_core.xml` and `metadata_thesis.xml`. The first is described with Qualified Dublin Core (QDC) metadata terms, while the second supplements the first with thesis-specific terms not available in the QDC element set. In this example, the number 83391 comes from the handle, or persistent ID, of the item in the repository¹. The `contents` file describes the remaining non-metadata files. It is described in Section 7.4 with multiple examples.

```
83391
├── contents
├── handle
├── Koehn_TE_D_2016.pdf
├── Koehn_TE_D_2016.pdf.jpg
├── Koehn_TE_D_2016.pdf.txt
├── metadata_thesis.xml
└── dublin_core.xml
```

Figure 7.3: Folder contents of the dissertation with handle number 83391

To process the metadata efficiently, we combined the two metadata files and then stored the result in a MongoDB database within a `metadata` collection. We further supplement the metadata with two additional elements, `searchTitle` and `searchAuthorStr`, which contain normalized versions of the title and author strings, for easier searching. These fields were created by removing all punctuation from the original `title` and `contributor-author` fields

¹The persistent handle for this item is <http://hdl.handle.net/10919/83391>

and then converting the entire string to lower case. Figure 7.4 is an example of a snippet of the metadata entry for a thesis.

```
{
  "contributor-author": "Aatique, Muhammad ",
  "title-none": "Evaluation of TDOA Techniques for Position Location in CDMA
  ↪ Systems ",
  "searchAuthorStr": "aatique muhammad ",
  "searchTitle": "evaluation of tdoa techniques for position location in cdma
  ↪ systems "
}
```

Figure 7.4: Entry in MongoDB with search fields for a sample ETD

Cross Referencing ETDs in VTechWorks and PQTD

We made use of three methods for connecting an ETD in VTechWorks with its counterpart in the PQTD.

1. Utilizing `searchTitle`

Either a full string or partial string match (by matching either the initial 60 characters or the trailing 60 characters) was performed to identify the handle number of the record from MongoDB. The handle number was then used to copy the full-text PDF as well as additional metadata files from VTechWorks.

2. Utilizing `searchAuthorStr`

A full string match was performed to identify the handle number of the document for the given ProQuestID. The method ensured that there was only a single result for the given ProQuestID author match—since some authors may have published both a master’s thesis and a doctoral dissertation. These cases were resolved separately.

A potential improvement to better identify documents by the author’s name would be

to make use of author name disambiguation and then store the results of the same as separate fields in MongoDB.

3. Manually identifying the handle number to copy full-text

A small subset of the ProQuest metadata files contained titles that were inconsistent with those in the Virginia Tech ETD collection. In one example, we found an ETD in the ProQuest collection with the title *Case Study of Academic Achievement Teams in Stellar County*. It had no match in the VTechWorks Collection. So we manually searched VTechWorks for documents using that title as our search string. We found the correct document in VTechWorks because the ProQuest title was the same as the title on the cover sheet of the document. But as it turned out, the item metadata in VTechWorks listed the title as *District Leadership Practices that Enhance and Sustain Student Achievement at the Elementary School Level Through the Use of the Academic Achievement Team*, which is completely different. Such cases failed to be retrieved by the aforementioned methods. In order to deal with this case, once this title was matched, the handle number was used to copy all the ETD data from the Virginia Tech ETD collection into the ProQuest folders.

Prepare PDF Files for Full-text Extraction

We use Grobid to extract full text from PDF documents. Some of the Virginia Tech ETDs have more than one PDF file associated with the ETD. This could be either because they have supporting documents, Institutional Review Board (IRB) protocols, separated chapters, and/or separated front-matter or back-matter. We identified three cases: single-PDF items, single-PDF items with supporting documents, and multiple-PDF items. Each is described below.

Single PDF items

The easiest case to process is an ETD with only one PDF. An example is shown in Figure 7.5. These items contain only a single PDF file that can be easily identified. For each of these PDF files, we use Grobid to extract full text.

```
Nanjundappa_M_D_2015.pdf    bundle:ORIGINAL primary:true
Nanjundappa_M_D_2015.pdf.txt bundle:TEXT      description:Extracted text
Nanjundappa_M_D_2015.pdf.jpg bundle:THUMBNAIL description:IM Thumbnail
```

Figure 7.5: Contents file of an ETD folder with a single PDF

Single PDF items with supporting documents

As shown from Figure 7.6, some items contain more than a single PDF file. The additional PDF files are typically supporting documents that do not contribute to the full-text of the ETD.

```
73236
├── Wagstaff_JF_D_2015.pdf
└── Wagstaff_JF_D_2015_support_3.pdf
```

Figure 7.6: PDF files of the dissertation with handle number 73236 with support PDF

```
Buffardi_KJ_D_2014.pdf    bundle:ORIGINAL primary:true
Buffardi_KJ_D_2014_support_1.pdf    bundle:ORIGINAL description:Supporting documents
Buffardi_KJ_D_2014.pdf.txt    bundle:TEXT      description:Extracted text
Buffardi_KJ_D_2014_support_1.pdf.txt    bundle:TEXT      description:Extracted text
Buffardi_KJ_D_2014.pdf.jpg    bundle:THUMBNAIL description:Generated Thumbnail
Buffardi_KJ_D_2014_support_1.pdf.jpg    bundle:THUMBNAIL description:Generated Thumbnail
```

Figure 7.7: Contents file of an ETD folder with a single PDF along with supporting documents

As illustrated by Figure 7.7, the `contents` file identifies the main PDF with the `bundle:ORIGINAL primary:true` tag. For each of these PDF files, we use Grobid to extract full text as we did with the single-PDF items.

```

Spruill_NR_D_2011.pdf    bundle:ORIGINAL
Spruill_NR_D_2011_IRB.pdf    bundle:ORIGINAL
Spruill_NR_D_2011.pdf.txt    bundle:TEXT    description:Extracted text
Spruill_NR_D_2011_IRB.pdf.txt    bundle:TEXT    description:Extracted text
Spruill_NR_D_2011.pdf.jpg    bundle:THUMBNAIL    description:Generated Thumbnail
Spruill_NR_D_2011_IRB.pdf.jpg    bundle:THUMBNAIL    description:Generated Thumbnail

```

Figure 7.8: Special Case: `contents` file of an ETD folder with a single PDF along with supporting documents

Figure 7.8 shows a special subset of this category wherein the `contents` file does not directly indicate the full-text PDF document. In this example, there are two PDFs annotated with the `bundle: ORIGINAL` tag. These additional files could be IRB documents, the author's CV, permission files, or fair use documents. To identify the correct document for this case, the `contents` file was manually processed and the additional documents were removed.

Multiple PDF items

Some items contain multiple PDF files wherein each file represents a chapter of the ETD, or a group of chapters is coupled together. Most of these files contribute to the full-text of the ETD. Figure 7.9 is an example of one such ETD that contains separated chapters. For this research, the appendices have not been included for any of the ETDs (if present separately).

```

26928
├── 01Chapters1--2--3.pdf
├── 02Chapter4.pdf
├── 03Chapter5.pdf
└── 04Appendices.pdf

```

Figure 7.9: PDF files of the dissertation with handle number 26928 with multiple PDFs

To collect these PDFs together for full-text extraction, the `contents` file was manually viewed and the individual PDFs were verified to ensure that only relevant PDFs were being collected for processing. These files were then renamed to have the handle number added as

```

04Appendices.pdf      bundle:ORIGINAL
01Chapters1-2-3.pdf  bundle:ORIGINAL
03Chapter5.pdf       bundle:ORIGINAL
02Chapter4.pdf       bundle:ORIGINAL
04Appendices.pdf.txt bundle:TEXT      description:Extracted text
01Chapters1-2-3.pdf.txt bundle:TEXT      description:Extracted text
03Chapter5.pdf.txt   bundle:TEXT      description:Extracted text
02Chapter4.pdf.txt   bundle:TEXT      description:Extracted text
04Appendices.pdf.jpg bundle:THUMBNAIL description:Generated Thumbnail
01Chapters1-2-3.pdf.jpg bundle:THUMBNAIL description:Generated Thumbnail
03Chapter5.pdf.jpg   bundle:THUMBNAIL description:Generated Thumbnail
02Chapter4.pdf.jpg   bundle:THUMBNAIL description:Generated Thumbnail

```

Figure 7.10: Contents file of an ETD folder with multiple PDFs

a prefix to the document name before we extract the full text.

7.4.1 Data Pre-Processing

To prepare the data for our classification models, we need to perform certain pre-processing steps to ensure that there are no stop words, the correct features have been selected, etc.

Formatting Classification Labels

The metadata contained classification subjects from the PQTD system that were not restricted to the top 30 categories selected by us. Due to this, we needed to modify the ProQuest subject categories assigned to each of the metadata records to ensure that only categories from the top 30 categories are used.

Splitting Data

We split the data into an approximate 80–20 distribution per category. It was necessary to ensure that there is equal representation from each subject category. Thus, the split was

performed within each category and the training and test files were then added together. In addition to this, due to the multi-labeled nature of the subject categories, it was necessary to perform a check to ensure that the file does not get included in both the train and test sets. For example, an ETD may appear under Statistics, and as part of the random selection for the training set. At the same time, if this record also exists under Computer Science, it may be selected randomly to be part of the test set. Such situations must be avoided. In order to achieve this, we grouped all the ETDs that had multiple labels (and existed under different subject categories) and ensured that only this subset would be picked for the training set. The test set was selected from the remaining data set. While the test set data contains ETDs with a single-label, the model was trained using multiple labels and is therefore able to predict multiple labels for ETDs.

Storing Consolidated Metadata in MongoDB

The previous `metadata` collection that was created contained only the VTechWorks metadata. To perform analysis on all our metadata, we created a new `metadata_consolidated` collection. This consolidated collection contains all of the fields from both ProQuest metadata as well as VTechWorks metadata. The ProQuest metadata fields have been prefixed by “PQ” whereas the VTechWorks metadata fields have been prefixed by “VT.” Certain fields such as title and degree exist in the metadata of both ProQuest and VTechWorks. Figure 7.11 is an example of a snippet of the metadata entry for a dissertation from the `metadata_consolidated` collection.

```
{
  "PQ_Classification" : "['0463: Statistics', '0478: Forestry', '0799: Remote
  ↪ sensing']"
  "PQ_Title" : "Fourier Series Applications in Multitemporal Remote Sensing
  ↪ Analysis using Landsat Data",
  "VT_Title" : "Fourier Series Applications in Multitemporal Remote Sensing
  ↪ Analysis using Landsat Data",
  "VT_degree_level" : "doctoral",
  "PQ_Degree" : "Ph.D."
}
```

Figure 7.11: Entry in the `metadata_consolidated` MongoDB collection for a sample ETD

Data Cleaning

To ensure that the data fed to the classification models is clean, we performed certain data cleaning steps:

1. Ensure that all of the date fields are represented in the same ISO format `yyyy-MM-dd'T'HH:mm:ss.SSS'Z`.
2. Remove all stop words using the set of English stop words that are part of the NLTK [\[52\]](#) package as well as remove all punctuation.
3. Perform lemmatization using the `WordNetLemmatizer` from NLTK. The lemmatization was performed on selected features of the ETD such as title and abstract. Features such as author, department, degree name were not lemmatized. These features were manually selected.
4. Convert all the text to lower case.

Dealing with Missing Values

After splitting the data and counting the number of missing values, it was found that the “PQ_Advisor” field and the “VT_contributor_committeechair” had a large number of missing values. After performing a comparison, it was found that only 1 ETD had no value for each of these fields. Thus, we created a “Derived_Advisor” field that captures the non-null values from either of these fields. If both exist, the value from the “VT_contributor_committeechair” field was selected.

For other fields, the number of missing values was not very large. For these fields, we used “U” to fill in the null value.

Feature Selection

We performed manual feature selection to decide which attributes are to be retained in order to train the model. The final features that were used to train the model are listed in Table 7.4.

Table 7.4: Set of features used to train the model

PQ Metadata	VT Metadata
PQ_Author	VT_Subjects
PQ_Degree	VT_Title
PQ_Identifier_keyword	VT_Type
PQ_Number_of_pages	VT_contributor_author
PQ_Publication_year	VT_contributor_department
PQ_Title	VT_degree_level
	VT_degree_name
Derived_Advisor*	VT_description_abstract

Formatting class labels for the classification task

To utilize the given multi-label ProQuest subject categories along with any of the multi-label

Table 7.5: Example ETD with multiple subject categories

<u>Title</u>	<u>Primary subject category</u>
<i>A history of the outplacement industry, 1960–1997: From job search counseling to career management. A new curriculum of adult learning</i>	Adult education (0516)
	<u>Secondary subject categories</u>
	Continuing education (0516) Management (0454) American history (0337)

approaches mentioned above, it was necessary to ensure that the data was formatted in a manner that could be understood by the classifiers. For this purpose, we transformed a given “PQ_Classification” field. As part of this process, we generated 30 class label columns for each of the ETD records which represented the top 30 subject categories. Figure 7.12 shows 5 transformed records with the class labels represented as “1” if present and “0” if absent. Similarly, Figure 7.13 shows five transformed records wherein instead of simply indicating the presence or absence of a class label, we assign a probability score to each of the class labels present for a given ETD. As indicated in Table 7.5, this could further be extended to give a higher score to the “Primary subject category” and a lower equally divided score to the remaining “Secondary subject categories.”

	Statistics	Computer_science	Environmental_science	Forestry	PQ_Classification
0	1	0	0	1	Statistics, Forestry
1	1	0	1	0	Statistics, Environmental science
2	1	0	0	0	Statistics
3	1	0	0	0	Statistics
4	1	0	0	0	Statistics

Figure 7.12: Sample 5 ETD records with transformed class labels

	Statistics	Computer_science	Environmental_science	Forestry	PQ_Classification
0	0.5	0.0	0.0	0.5	Statistics, Forestry
1	0.5	0.0	0.5	0.0	Statistics, Environmental science
2	1.0	0.0	0.0	0.0	Statistics
3	1.0	0.0	0.0	0.0	Statistics
4	1.0	0.0	0.0	0.0	Statistics

Figure 7.13: Sample 5 ETD records with transformed class labels (with probability scores)

Machine Learning Algorithms

In text classification, some of the common algorithms used include Logistic Regression, Support Vector Machine, and Random Forest. For this dataset, we perform experiments using these three algorithms.

Logistic Regression

Due to some issues with the parameters when training for LabelPowerset, we have only considered Logistic Regression in the case of the BinaryRelevance approach. However, there were some convergence issues when using this approach with GridSearch. Thus, we have used default values as indicated in Table 7.6.

Table 7.6: Default parameter values utilized for Logistic Regression

Parameter Name	Value
Penalty	L2
C	1.0
Solver	liblinear
Maximum Iterations	100

Support Vector Machine

The parameters selected by GridSearch for the LabelPowerset approach have been indicated in Table 7.7. The parameters for which values have not been mentioned in the table have

default values assigned.

Table 7.7: Parameter values selected for Support Vector Machine

Parameter Name	Value
Kernel	Linear
C	5
Gamma	Auto
Class_Weight	Balanced

Random Forest

The parameters selected by GridSearch for the LabelPowerset approach have been indicated in Table 7.8. The parameters for which values have not been mentioned in the table have default values assigned.

Table 7.8: Parameter values selected for Random Forest

Parameter Name	Value
n_estimators	10
Criterion	Gini
Random State	10

7.5 PQDT Dataset

In this section, we discuss the methods and system design for the Machine Learning and Deep Learning models trained on the PQDT Dataset. This dataset contains the full-text PDF along with metadata information. The additional auxiliary step of cross referencing the ETDs in VTechWorks and PQDT is not performed. Similarly, full-text extraction is performed using `Tika-Python` as described in Section 5.1.7.

Due to the relatively poor performance of the probability scores variant of our experiments with the Virginia Tech dataset, we decided to focus on the presence-absence variant of the

experiments for this dataset. An example, for 5 ETDs with transformed labels, appeared in the previous section in Figure 7.12.

7.5.1 Data Pre-processing

In this section, we describe the various data pre-processing steps that are performed on the PQDT dataset.

Similar to the steps performed for the Virginia Tech dataset, as discussed in Section 7.4.1, we perform the `Formatting Classification Labels` and `Splitting Data` steps for this dataset as well. However, here we only consider 28 out of the original 30 subject categories due to a limited number of ETDs being present in the two categories that are dropped. These categories are: ‘Mechanics’, and ‘Management’. A list of the 28 subject categories along with the number of ETDs associated with each of them has been illustrated in Table 4.9.

Data Cleaning

The steps performed to clean the PQDT dataset include the following.

1. Remove all stop words using the set of English stop words that are part of the NLTK package, as well as remove all punctuation.
2. Perform lemmatization using the `WordNetLemmatizer` from NLTK. The lemmatization is performed only on non-NER fields.
3. Convert all the text to lower case.

Feature Selection

We perform manual feature selection to identify the ones to be included to train the models.

The final set of features is shown in Table 7.9.

Table 7.9: Set of features used to train the model

Advisor	Committee
Department	Keywords
School	Abstract
Author	Degree_obtained
Full Text	Number of Pages
Title	Year

For the models built using the combined summarized chapter data, we do not use the ‘Full Text’ as a feature, but instead, use one of the different summary variants. The list of these summary features appears in Table 7.10

Table 7.10: Set of combined chapter summary features used to train the model

<code>gensim's TextRank Generated Summary with ratio of 0.2</code>
<code>gensim's TextRank Generated Summary with 100 words</code>
<code>sumy's LexRank Generated Summary</code>
<code>sumy's Generated Summary using Luhn's Algorithm</code>
<code>sumy's Generated Summary using LSA</code>
<code>sumy's Generated Summary using LSA with stopwords</code>

Dealing with Missing Values

None of the ETDs present in this corpus has missing values for the abstract and full-text fields. For the metadata fields that were included as indicated in Table 7.9, the number of missing values is not very large. For these fields, we use ‘U’ to fill in the null value.

Machine Learning Algorithms

For this dataset, we perform experiments using Support Vector Machine and Random Forest.

Support Vector Machine

Based on the parameters selected by GridSearch for the LabelPowerset approach on the Virginia Tech dataset, we use the parameters mentioned in Table 7.11. The parameters for which values are not mentioned in the table have default values assigned.

Table 7.11: Parameter values selected for Support Vector Machine

Parameter Name	Value
Kernel	Linear
C	5
Gamma	Auto
Class_Weight	Balanced

Random Forest

Based on the parameters selected by GridSearch for the LabelPowerset approach on the Virginia Tech dataset, we use the parameters mentioned in Table 7.12. The parameters for which values are not mentioned in the table have default values assigned.

Table 7.12: Parameter values selected for Random Forest

Parameter Name	Value
n_estimators	10
Criterion	Gini
Random State	10

Deep Learning Algorithms

As discussed in Sections 3.3.2 and 3.3.2, the authors of [4] and [70] indicate the benefits of simpler deep learning architectures over more complex ones in the domain of long document

classification.

Thus, for this dataset, we perform experiments using different deep learning architectures including Gated Recurrent Units (GRU), Bidirectional Gated Recurrent Units (BiGRU), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BiLSTM).

Based on our experiments with different hyperparameters, that were performed using 5-fold cross validation, we found that our models perform well when trained for 100 epochs, with a state size of 1024, dropout of 0.2, and batch size of 512. The optimizer that gave the best performance was the ‘Adam’ optimizer.

7.6 Evaluation Metrics

Since we are performing multi-label classification, the evaluation metrics of accuracy, precision, recall, and F1-score mentioned in Section 3.3.3 need to be tweaked such that we can average over the classes. For this purpose, the commonly used averaging strategies are *micro-averaging* and *macro-averaging* [42]. In the case of *micro-averaging*, we calculate metrics globally by counting the total number of true positives, false negatives, and false positives. Alternatively, in the case of *macro-averaging*, we calculate metrics for each label and find their unweighted mean. Scikit-learn [13] also has a provision for *weighted-averaging* wherein we calculate the metrics for each of the labels present and find their average values based on their support (i.e., the number of instances for each label that are present). This helps to overcome the drawback of *macro-averaging* which is unable to deal with label imbalance.

Similarly, to compute the Hamming loss, Jaccard index, and exact match score, we use the ‘MultiLabelBinarizer’ from `scikit-learn` to convert our predicted labels into the correct

format that is expected by these metrics. We use the implementation of the metrics provided by `scikit-learn`.

Chapter 8

Results and Discussion

In this chapter, we discuss the results obtained for the different experiments that were performed. We first present the results obtained on the Virginia Tech dataset, followed by the PQDT dataset. We also provide analysis of the results obtained.

8.1 Virginia Tech Dataset

The results presented in this section were completed as part of the project for the *CS6604: Digital Libraries* [6] course.

The various experiments conducted have been listed as part of Figure 8.1. In addition to the LabelPowerset experimental setup that was performed with the presence-absence labels, another set of experiments was conducted by making use of probability scores for each of the subject categories labels. Each of the classes that were associated with a given ETD were assigned an equal probability score.

The training time has been calculated using the `time.process_time()` function provided by Python. This function returns a float value of the time taken in seconds. The other evaluation metrics have been discussed in Section 7.6.

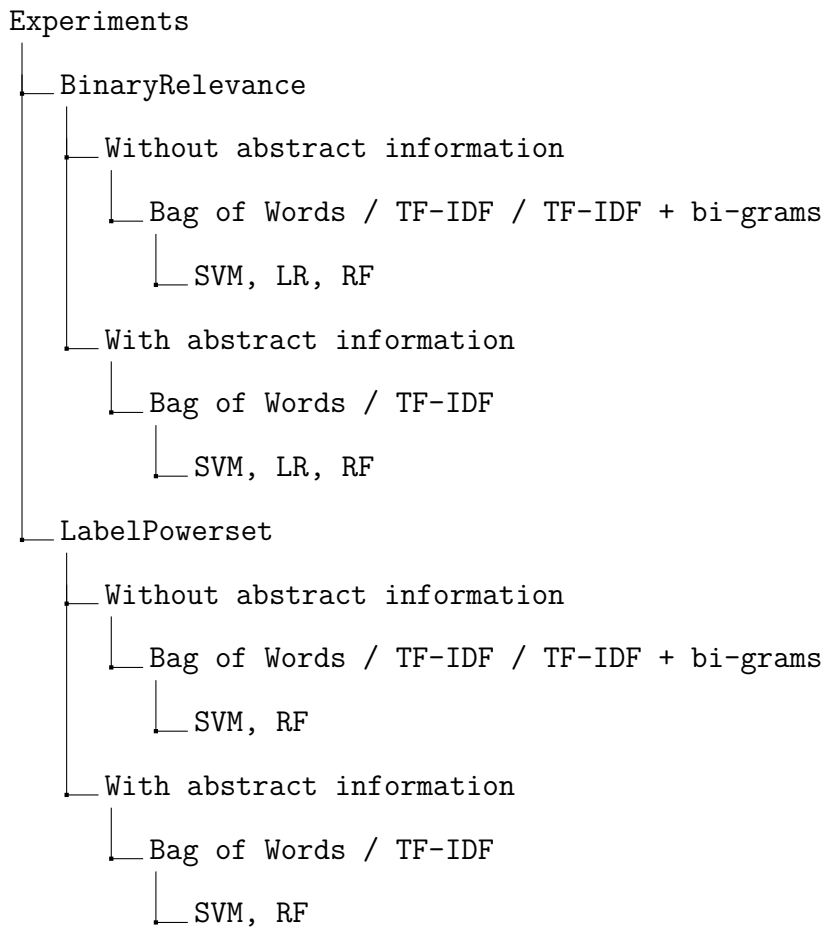


Figure 8.1: Various experimental setups

8.1.1 BinaryRelevance

Here we present the results obtained by using the BinaryRelevance approach. We did not perform experiments with the BinaryRelevance approach and probability scores class labels. All results presented in this section are with the presence-absence class labels.

Analysis

As indicated by Figure 8.2, the first, third, and last records include additional labels as com-

Table 8.1: Performance of various experimental setups using BinaryRelevance (without abstract)

Experiment setup	Accuracy	Precision	Recall	F1-score	Training Time
TF-IDF(bi) + SVM	63.3	77.6	80.5	77.7	1718.4
TF-IDF(bi) + RF	24	47.9	91	57.7	29.3
TF-IDF(bi) + LR	54.6	71.8	87.6	75.6	19.1
TF-IDF + SVM	63.9	77.4	82.4	78.5	448.5
TF-IDF + RF	20.2	32.6	83.6	44.6	13.1
TF-IDF + LR	50.7	68.2	86.6	73.0	7.99
BOW + SVM	55.4	69.4	79.9	72.6	355.2
BOW + RF	17.5	30.3	86.8	43	13.1
BOW + LR	40.1	52.8	82.0	62.3	11.9

Table 8.2: Performance of various experimental setups using BinaryRelevance (with abstract information)

Experiment setup	Accuracy	Precision	Recall	F1-score	Training Time
TF-IDF + SVM	64.63	79.30	83.0	79.5	818.2
TF-IDF + RF	11.8	23	94.2	35	18.5
TF-IDF + LR	50.5	67.5	86.9	73.1	17.3
BOW + SVM	36.9	54.3	65.8	57.9	634.96
BOW + RF	12.8	21.8	93.5	33.8	17.4
BOW + LR	31.2	44.3	68.3	52.4	24.7

PQ_Identifier_keyword	PQ_Title	VT_Subjects	VT_Title	VT_contributor_department	Expected_Labels	Predicted_Labels
education academic achievement ethnicity men persistence race self efficacy socioeconomic status	predict academic achievement male college student	ses self efficacy race ethnicity academic achievement academic persistence and men family	predict academic achievement male college student	educational leadership and policy studies	Higher_education	Educational_psychology, Higher_education
applied sciences dynamics fluid mechanics rowing boats sensitivity analysis	multi physic multilevel fidelity modeling analysis olympic row boat dynamic	sensitivity analysis rowers biomechanics water forces rowing rigid body motions	multi physic multilevel fidelity modeling analysis olympic row boat dynamic	engineering science and mechanics	Mechanics	Mechanical_engineering
social sciences women managers	relationship african american european american woman hotel management career	hotel manager career race gender	relationship african european american woman hotel management career	hospitality and tourism management	Marketing	Management, Marketing
biological sciences melospiza georgiana salt marshes swamp sparrows tidal marshes	life history divergence tidal salt marsh adaptation coastal plain swamp sparrow	melospiza georgiana optimal habitat clutch size extrapair fertilization sexual selection conspecific attraction	life history divergence tidal salt marsh adaptation coastal plain swamp sparrow	biological sciences	Ecology	Ecology, Forestry

Figure 8.2: Analysis of 5 differently classified ETDs using the BinaryRelevance approach

pared to the one assigned to the record as per the ground truth. After examining the fields associated with the first record, we find terms such as “education” and “academic achievement,” which could explain the category of “Educational_psychology” that is assigned to this record. Similarly, for the third record, the department associated with this field is “Hospitality and Tourism Management,” and there are keywords such as “hotel management” that are present in the metadata of this record which could explain the label “Management” that was assigned to this record.

In the case of the experiments that utilized the abstract information, we find that in both the records indicated by Figure 8.3, the class labels predicted by the model are the same as the department name. This indicates a high correlation between the department name and the predicted class labels.

8.1.2 LabelPowerset

Here we present the results obtained by using the LabelPowerset approach.

PQ_Identifier_key word	PQ_Title	VT_Subjects	VT_Title	VT_contributor_ department	VT_description_abstract	Expected_Labels	Predicted_Labels
pure sciences applied sciences additive manufacturing inkjet photopolymers physical unctionable functions polyjets quantum dots	effect quantum dot nanoparticles polyjet direct 3d printing process	additive manufacturing polyjet inkjet physical unctionable functions quantum dot photopolymer	effect quantum dot nanoparticles polyjet direct 3d printing process	mechanical engineering	additive manufacturing unique method fabrication contrast traditional manufacturing method build object layer layer ability partner 3d scan clone physical object raise concern area intellectual property ip address issue goal dissertation characterize model method incorporate unique security feature within build add optically detectable nanoparticles transparent medium physical uncloneable function pufs embed build serve anti counterfeit measure nanoparticle select work quantum dot qd absorbs uv light emits light visible spectrum unique interaction light make qds ideal security system since challenge uv light different signal response visible light emit qds polyjet process select work utilizes inkjet deposit photopolymer layer cure uv light investigation visibility qds within print polyjet medium reveal qds produce puf pattern visible via fluorescent microscopy furthermore rheological data show ink jet property print medium significantly affect qds sufficient concentration produce pufs final objective study characterize effect qds photocuring mathematical model predict critical exposure qd dope photopolymer utilizes light scatter theory qd characterization result photopolymer cure characterization result mathematical representation contribute toward body knowledge area additive manufacturing nanomaterials photopolymers overall work embodies first investigation effect qds rheological characteristic ink jetted medium effect qds cure photopolymer medium visibility nanoparticles within printed medium first attempt incorporate security feature within build finally major scientific contribution work theoretical model develop predict effect qds cure property photopolymers	Civil_engineering	Mechanical_engineering
education computer science education conceptual learning engagement object oriented peer review	peer review cs2 assessment effect attitudes engagement conceptual learning	computer science education peer review peer concept learning	peer review cs2 effect attitudes engagement conceptual learning	computer science	computer science student could benefit exposure critical programming concept multiple perspective peer review one method allow student experience authentic us concept non program manner peer review provide student opportunity evaluate peoplevc work allow rich learn experience much know peer review many us discipline literature especially computer science spend much time perspective benefit reviewer work examine implement peer review process early object orient computer science course way develop reviewersvc high level think skill increase knowledge specific program concept improve attitude help engage activity specifically explore peer review effect abstraction decomposition encapsulation type review student review peer review material instructor influence effect also look studentsvc attitude relate engagement benefit review study idea use peer review exercise two cs2 class local university course semester divide student three group one group review peer one group review instructor one group complete small design cod exercise measure studentsvc attitude conceptual understanding semester survey test concept map collect complete review well found review help student learn decomposition especially review instructorvc program find evidence improvement studentsvc level thinking semester significant change attitude however data show student assign review peer less likely complete assignment student overall peer review valuable method teach decomposition cs2 student use alternative way learn object orient programming concept	Higher_education	Computer_science

Figure 8.3: Analysis of 5 differently classified ETDs using the BinaryRelevance approach (with abstract)

Using Presence-absence Class Labels

Tables 8.3 and 8.4 represent the results obtained using presence-absence class labels. The parameters used to train these models were selected using the Grid Search method. When we attempted to perform a Grid Search to identify the best parameters to train the experimental setup comprising of TF-IDF with bi-grams and abstract information, it took a very large amount of time to train. Thus, the results for this experiment are not included in this report.

Analysis

In this section, we focus on the best classifier (based on F1-score) in each subset of experimental setups and analyze the performance of these classifiers.

Figure 8.4 represents the expected and predicted labels for the Random Forest classifier + BOW approach trained using LabelPowerset (without probability scores).

Table 8.3: Performance of various experimental setups using LabelPowerset (without abstract)

Experiment setup	Accuracy	Precision	Recall	F1-score	Training Time
TF-IDF(bi) + SVM	60.3	74.4	64	66.3	256.5
TF-IDF(bi) + RF	64.0	79.1	67.7	69.6	4.8
TF-IDF + SVM	61.3	75.6	65	67.6	97.9
TF-IDF + RF	57.2	73.5	60	63.5	0.8
BOW + SVM	58.9	73.9	62.5	64.4	85.5
BOW + RF	63.7	80.4	67.9	70.7	8.8

Table 8.4: Performance of various experimental setups using LabelPowerset (with abstract)

Experiment setup	Accuracy	Precision	Recall	F1-score	Training Time
TF-IDF + SVM	61.9	76.4	65.4	68.1	169.2
TF-IDF + RF	60.1	78	64.1	66.2	15.1
BOW + SVM	44.8	64.0	49.4	51.3	162.7
BOW + RF	65.2	79.6	68.3	70.4	8.2

PQ_Identifier_keyword	PQ_Title	VT_Subjects	VT_Title	VT_contributor_department	Expected_Labels	Predicted_Labels
pure sciences education differential item functioning mantel haenszel procedures	effect unequal sample size power dif detection irt base monte carlo study sibtest mantel haenszel procedure	monte carlo simulation combination ratios sample size nominal p value dif detection dif magnitude statistical power differential item functioning	effect unequal sample size power dif detection irt base monte carlo study sibtest mantel haenszel procedure	educational leadership and policy studies	Statistics	Secondary_education
education pure sciences computer experience internet primacy effect satisficing surveys	contribution respondent computer experience primacy effect satisfice internet survey	computer experience response effects internet surveys online surveys primacy effect satisficing	contribution respondent computer experience primacy effect satisfice internet survey	research and analysis	Statistics	Mathematics
applied sciences equivalent plate analysis finite element wings	efficient method structural analysis build wing	mindlin plate theory sensitivity continuum model equivalent plate model ritz method neural network structural analysis built up wing	efficient method structural analysis build wing	aerospace and ocean engineering	Mechanical_engineering, Mechanics	Aerospace_engineering
applied sciences autonomous vehicles compressed sensing data compression haar wavelet transform occupancy grids slam simultaneous localization and mapping sparse signal reconstruction underwater autonomous vehicles	real time slam use compressed occupancy grid low cost autonomous underwater vehicle	autonomous vehicles slam occupancy grids haar wavelet transform compressed sensing sparse signal reconstruction data compression	real time slam use compressed occupancy grid low cost autonomous underwater vehicle	mechanical engineering	Computer_science	Electrical_engineering, Mechanical_engineering
health and environmental sciences communication and the arts social sciences business clothing disabilities innovations specialized product development women workers	innovation improvisation study specialized product development focus business clothing woman physical disability	working women clothing manufacturing co design universal product development clothing physical disabilities	innovation improvisation study specialized product development focus business clothing woman physical disability	near environments	Marketing	Environmental_science

Figure 8.4: Analysis of 5 incorrectly classified ETDs using the LabelPowerset approach

If we analyze records 1, 3, and 5, it appears as though the “VT_contributor_department” plays an important role in determining the labels that have been predicted for these records.

If we were to observe record number 4, the expected label is meant to be “Computer_science.” However, if we were to look at the “VT_contributor_department,” the predicted label “Mechanical_engineering” is not entirely incorrect as this research would in fact have some relation to Mechanical Engineering. Similarly, if we were to observe some of the “VT_Subjects” or “PQ_Identifier_keyword,” we see terms such as “sparse signal reconstruction” which could justify the predicted label of “Electrical_engineering.”

Based on the results obtained, it could be said that the classifiers built as part of this study help to add to discovery and aid in the identification of other categories that could be assigned for these ETDs which gives us more information than the original ProQuest subject categories.

PQ_Identifier_keyword	PQ_Title	VT_Subjects	VT_Title	VT_contributor_department	VT_description_abstract	Expected_Labels	Predicted_Labels
pure sciences education differentiation item functioning mantel haenszel procedures	effect unequal sample size power dif detection in base monte carlo study sibstest mantel haenszel procedure	monte carlo simulation combination ratios value dif detection dif magnitude statistical power differential item functioning	effect unequal sample size power dif detection in base monte carlo study sibstest mantel haenszel procedure	educational leadership and policy studies	html head title risper akelo avuor abstract title meta http equiv content type content text html charset iso 8859 1 head body p simulation study focus determine effect unequal sample size statistical power sibstest mantel haenszel procedure detection dif moderate large magnitude item parameter estimate generate 2pm use wingent2 han 2006 nutisim use simulate ability estimate generate response data analyse sibstest sibstest procedure regression correction use calculate dif statistic namely dif effect size statistical significance bias old sibstest use calculate dif statistic h procedure sa provide environment ability parameter simulate response data generate dif analysis conduct test item observe determine priori manipulate item demonstrate dif study result indicate unequal sample ratio h good type error rate control sibstest result also indicate ratio also sample size magnitude dif influence behavior sibstest h regard error rate behavior small sample moderate dif magnitude type ii error commit h sibstest reference focal group sample size ratio 1 10 due low observe statistical power inflate type error rate p body html	Statistics	Statistics
applied sciences earth sciences floating treatment wetlands nutrient control nutrient harvest performance assessment retention ponds stormwater treatment sustainability	float wetland urban stormwater treatment	retention ponds nutrient control performance assessment nutrient harvest sustainability	float wetland urban stormwater treatment	biological systems engineering	float treatment wetland fw ecological approach seek reduce point nonpoint source pollution instal substrate root plant grown float mat open water relatively novel fw use increase review literature identify several research gap include 1 assessment treatment performance fws 2 evaluation fws u particularly within wet pond receive urban runoff 3 plant temporal nutrient distribution plant growth rate long term persistence fws 4 temperate region periodic ice encasement assessment model fw model develop parameter fit base data 14 publish fw study first research topic estimate median fw apparent uptake velocity 95 confidence interval 0 048 0 038 0 059 0 027 0 036 0 040 day total phosphorus tp total nitrogen tn respectively fw model provide accurate prediction nutrient removal two common performance metricsremoval rate mg m2 day removal efficiency second research topic result mesocosm experiment indicate fws 63 coverage plant pickerelweed pontederia cordata 1 softstem bulrush schoenoplectus tabernaemontani significantly improve tp tn removal efficiency control treatment 8 2 18 2 respectively pickerelweed exhibit significantly high phosphorus nitrogen removal softstem bulrush water temperature great 25 deg c field observation third research topic find pickerelweed demonstrate high phosphorus removal performance 7 58 mg plant softstem bulrush 1 62 mg plant base observe seasonal change phosphorus distribution harvest ground vegetation recommend conduct twice year june september plant perennial macrophytes successfully adapt stress low dissolve oxygen concentration minimum 1 2 mg l ice encasement relatively low nutrient concentration water median10 15 mg l to 1 15 mg l 1st systematic observation wildlife activity indicate eight class organisms inhabit forage breed nurse rest fws recommendation fw design suggestion research make base upon finding	Civil_engineering	Ecology
applied sciences assistive technology human computer interaction participatory design user interface visual impairment wayfinding wearable computer	environmental user interface user framework comvey participatory design user interface visual impairment wayfinding severe visual impairment interactive system design	assistive technology user interface framework comvey participatory design user interface visual impairment human computer interaction	environmental user interface eui framework comvey environmental contexts interactive system design	industrial and systems engineering	world health organization estimate 488 million people worldwide suffer visual impairment 327 million severe visual impairment individual severe visual impairment navigate orient independently well know surroundings even people independent navigation orientation likely challenge unfamiliar place overcome challenge assistive technology develop support independent wayfinding task however severe visual impairment often experience frustration try use assistive technology since technology lack address environmental factor influence independent wayfinding research develop evaluate efficacy framework call environmental user interface eui particular research explore whether propose eui framework effective user centered design ucd design wayfinding system capture environmental requirement thus aid severe visual impairment two study first consist requirement elicitation second usability test conduct study reveal eui framework indeed effective conventional ucd design method alone identifying environmental factor participant severe visual impairment prefer use prototype design use ucd eui framework propose eui framework find effective way enhance design process play important role elicit great number environmental factor hence produce device prefer user visual impairment prototype influence well wayfinding task perform five participant severe visual impairment prototype implement base requirement elicit ucd eui framework much prefer participant	Computer_Engineering	Industrial_engineering
biological sciences health and environmental sciences aquatic invertebrates eichornia crassipes freshwater water hyacinth invasive species lake chapala mexico water hyacinth waterbirds	ecological effect eichornia ecology aquatic invertebrates freshwater ecology invasive species	water hyacinth eichornia crassipes waterbirds community ecology aquatic invertebrates freshwater ecology invasive species	ecological effect water hyacinth eichornia crassipes lake chapala mexico	fisheries and wildlife sciences	water hyacinth eichornia crassipes float non native plant reoccurring lake chapala jalisco mexico 100 year research explore effect water hyacinth freshwater ecosystem worldwide specifically lake chapala p chapter 1 review study conduct water hyacinth worldwide find effect water hyacinth water quality similar magnitude effect dependent percent cover potentially spatial configuration water hyacinth mat water hyacinth effect aquatic invertebrate fish waterbird less predictable dependent condition prior invasion chapter 2 test relationship percent water hyacinth cover waterbird abundance specie diversity community composition habitat use general find weak positive relationship relationship variable chapter 3 monitor habitat use american coot fulca americana variety habitat around lake chapala find time spent water hyacinth positively correspond percent water hyacinth cover time foraging water hyacinth positively related time spent water hyacinth p chapter 4 compare invertebrate assemblage open water within edge water hyacinth mat emergent vegetation submerge tree also examine invertebrate assemblage within root water hyacinth plant compare assemblage patch shoreline water hyacinth plant find density taxonomic richness water column invertebrates generally high association water hyacinth mean percent cover water hyacinth affect magnitude difference among habitat vegetation type find significant difference root invertebrate density taxonomic richness patch shoreline water hyacinth plant chapter 5 discuss water hyacinth affect dissolved oxygen water transparency small localize scale dive factor seasonal difference overall result suggest water hyacinth minimal ecological effect lake chapala study	Environmental_science	Ecology

Figure 8.5: Analysis of 5 incorrectly classified ETDs using the LabelPowerset approach (with abstract information)

As indicated by Table 8.5, the first record has been correctly classified as “Statistics,” which was previously incorrectly classified by the RF classifier built using BOW but without abstract information. In case of the second record, the ground truth label is “Civil_engineering,” whereas the model predicted the class to be “Ecology.” However, on examining the features of this record, “Ecology” appears to be a valid label as well.

Using Probability Scores Class Labels

Tables 8.5 and 8.6 represent the results obtained using the probabilistic class labels. Details about this formatting have been discussed in Section 7.4.1. The parameters used to train these models were selected based on the best parameters chosen for the presence-absence

class labels. This was done since the Grid Search method from scikit-learn [13] was giving an error due to the probability values associated with the multi-label classes of the training set. Further analysis can be performed on the same to find a way to perform a Grid Search in this case as well, which may lead to an improvement in the performance of the models. The predictions generated by these models provided a probability score for each of the 30 subject categories.

Table 8.5: Performance of various experimental setups using LabelPowerset (without abstract) with probability scores class labels

Experiment setup	Precision	Recall	F1-score	Training Time
TF-IDF(bi) + SVM	85.4	28.8	39.9	1030.8
TF-IDF(bi) + RF	87.2	30.3	43.1	4
TF-IDF + SVM	85.2	28.7	39.9	397.1
TF-IDF + RF	85.5	29.6	42.5	2.3
BOW + SVM	88.13	29.6	41.7	352.6
BOW + RF	89	30.8	44.3	2.3

Table 8.6: Performance of various experimental setups using LabelPowerset (with abstract) with probability scores class labels

Experiment setup	Precision	Recall	F1-score	Training Time
TF-IDF + SVM	84.7	28.7	39.9	676.2
TF-IDF + RF	77.2	26.7	38.2	3.95
BOW + SVM	81.3	26.7	38.3	679.6
BOW + RF	78.8	26.8	38.6	2.9

Formulation of class labels

As indicated by Figures 8.6 and 8.7, the labels predicted by the classifiers in this set of experimental setups is a probability score. However, since we have 30 potential labels, due to the values generated by the classifier, with varied probabilities, it is difficult to determine

a fixed threshold value to pick the labels for the input. Thus, for the purpose of this study, we have chosen the top 3 subject categories as the labels to be generated for a given input record.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0.1	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.1	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.2	0.6	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	

Figure 8.6: Probability values predicted by the classifier (BOW + RF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0.03	0.05	0.07	0.05	0.07	0.08	0.14	0.04	0.06	0.05	0.11	0.04	0.04	0.03	0.05	0.06	0.04	0.02	0.08	0.03	0.14	0.04	0.04	0.03	0.02	0.03	0.04	0.04	0.06	0.04
0.05	0.03	0.04	0.05	0.03	0.03	0.04	0.07	0.03	0.07	0.04	0.04	0.07	0.14	0.07	0.05	0.05	0.04	0.05	0.04	0.06	0.06	0.06	0.05	0.02	0.03	0.04	0.05	0.04	0.05
0.05	0.03	0.04	0.06	0.04	0.03	0.04	0.04	0.02	0.05	0.04	0.04	0.07	0.06	0.07	0.06	0.07	0.16	0.06	0.03	0.06	0.04	0.03	0.06	0.02	0.05	0.04	0.04	0.02	0.06

Figure 8.7: Probability values predicted by the classifier (TF-IDF with bi-grams + SVM)

Figure 8.8 represents the resultant set of values for the probability scores given in Figure 8.6 for the BOW + RF classifier. The probability values have been replaced by presence (1) or absence (0) values since the evaluation metrics within scikit-learn are not currently equipped to work with multi-label continuous output values. However, since the model probability scores are still generated, we can still associate a probabilistic score for each of the top 3 labels.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	

Figure 8.8: Predicted labels after picking the top 3 subject categories (BOW + RF)

Figure 8.9 illustrates the corresponding top 3 subject category names for the categories with presence ('1') values represented in Figure 8.8.

Analysis

In this section, we focus on the best classifier (based on F1-score) in each subset of experi-

Expected_Labels	Predicted_Labels
Computer_science	Computer_science, Higher_education, Educational_psychology
Forestry	Forestry, Ecology, Molecular_biology
Management	Marketing, Management, Public_administration

Figure 8.9: Expected and Predicted labels after picking the top 3 subject categories (BOW + RF)

mental setups and analyze the performance of these classifiers.

Figure 8.10 represents the expected and predicted labels for the Random Forest classifier + BOW approach trained using LabelPowerset (with probability scores).

If we were to examine record 1 in Figure 8.10, we can justify the additional labels of “Higher_education” and “Educational_psychology” by examining some of the keywords. Words such as “education” and “pedagogical” present among the words in record 1, indicate education and thus, these new labels can be possible new categories that could be assigned to this ETD.

Similarly, in the case of record 2, the keywords microbial, nutrient, etc. indicate “Molecular_biology” which is one of the additional labels. Keywords such as soil, earth sciences, etc. could very well justify the presence of the label “Ecology.”

As was inferred from the results discussed in Section 8.1.2, the “VT_contributor_department” plays an influential role in the labels obtained, and each of the records represented in Figure 8.10 testify to this inference.

PQ_Identifier_keyword	PQ_Title	VT_Subjects	VT_Title	VT_contributor_department	Expected_Labels	Predicted_Labels
applied sciences education corgis computing comutational thinking data science datasets motivation	motivate introductory compute student pedagogical datasets	motivation introductory computing computational thinking engagement corgis datasets data science data pedagogical datasets computer science education engagement	motivate introductory compute student pedagogical datasets	computer science	Computer_science	Computer_science, Higher_education, Educational_psychology
biological sciences earth sciences decomposition hardwood forests microbial biomass nutrient cycling soil heterogeneity	soil resource heterogeneity site quality southern appalachian hardwood forest impact decompose stump geology salamander abundance	fine root dynamics appalachian hardwoods nutrient cycling soil heterogeneity ground penetrating radar salamanders decomposing stumps microbial biomass	soil resource heterogeneity site quality southern appalachian hardwood forest impact decompose stump geology salamander abundance	forestry	Forestry	Forestry, Ecology, Molecular_biology
social sciences brand equity customer equity hospitality businesses	create validate measure customer equity hospitality business link shareholder value return marketing	relationship equity brand equity value equity servperf firm valuation customer lifetime value shareholder value customer equity	create validate measure customer equity hospitality business link shareholder value return marketing	hospitality and tourism management	Management	Marketing, Management, Public_administration

Figure 8.10: Analysis of 5 incorrectly classified ETDs (with probability scores)

8.2 PQDT Dataset

In this section, we present the results obtained for the models built using the PQDT dataset. First, we discuss the results obtained for the Machine Learning models followed by a discussion of the results for the deep learning models. We use the LabelPowerset approach to convert the multi-labels into multi-class as the training time needed for this was less than that of the BinaryRelevance approach. Additionally, the probability models reported in the previous section had lower F1-scores as compared to the presence-absence labels, thus, for this dataset, we focused on the use of presence-absence labels.

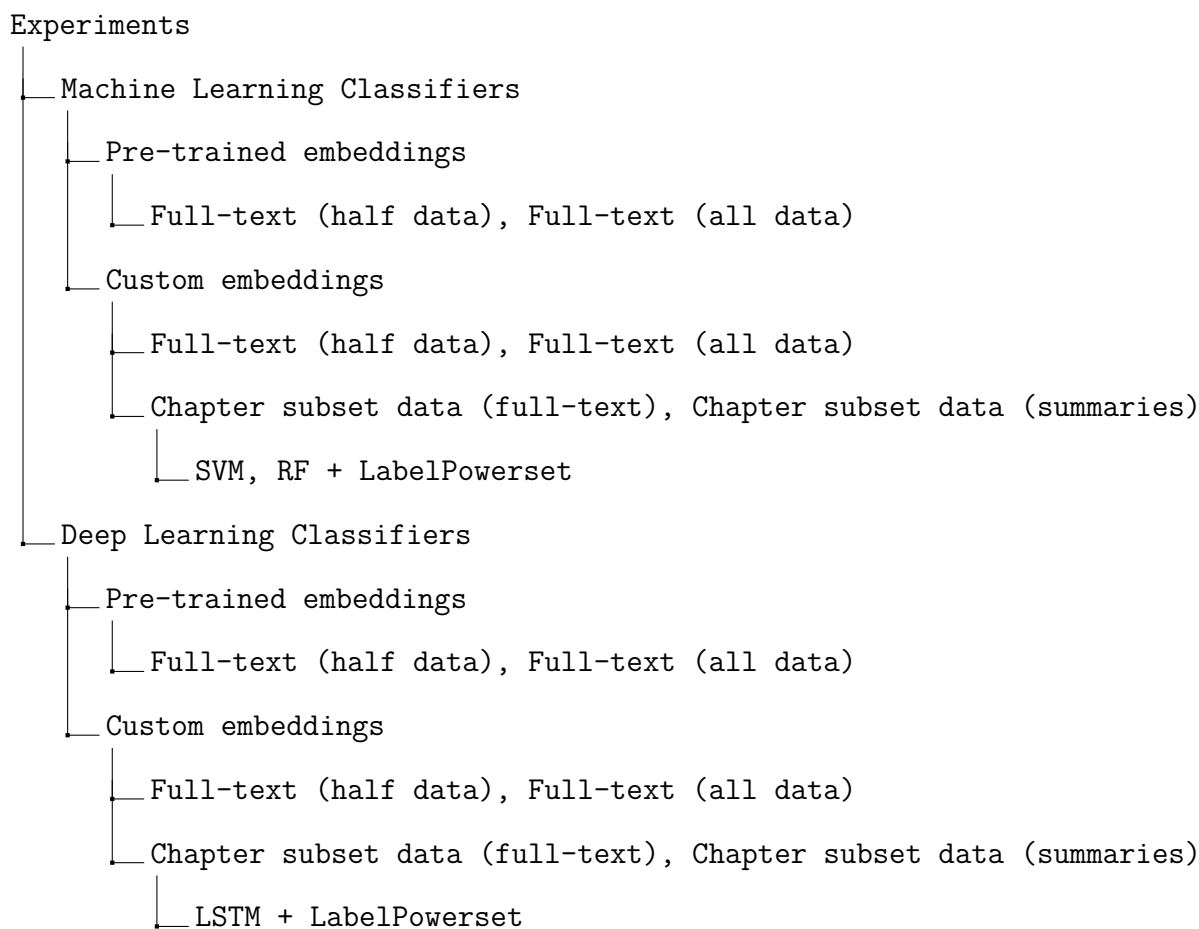


Figure 8.11: Various experimental setups

First, we present the performance of the pre-trained word and document embeddings for each category of our models. As indicated in Tables 7.2 and 7.3 we trained different word embedding and document embedding models in an attempt to better understand the effect of these embeddings on the performance of the classifiers. Thus, we train different models to identify the best possible custom embeddings for that given model and data subset and also contrast the performance of the classifiers against that of the pre-trained word embeddings trained using Wikipedia. After the best models in each category have been identified, we present the chapter-level labels generated by the best models.

The following subsections discuss each of the three different subsets of the data as indicated by Table 7.1. Figure 8.11 gives an overview of the different experimental setups that were performed on this dataset.

8.2.1 Machine Learning Models

For the purpose of this study, the Machine Learning algorithms that we focus on are Support Vector Machine (SVM) and Random Forest (RF) algorithms.

Pretrained Embeddings

In this section, we present the results obtained for the SVM and RF models when we use the pre-trained fastText and Doc2Vec embeddings that were trained on the Wikipedia and Common Crawl datasets.

Table 8.7: Performance of SVM and RF built using pre-trained embeddings

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	35.7	51.5	37.8	42.8	Full-text (half data)
SVM	46.7	68.9	52.8	58.9	Full-text (half data)
RF	40.4	53.8	41.7	46.2	Full-text (All data)
SVM	51.4	70.1	55.9	61.6	Full-text (All data)

Custom Embeddings

We trained a total of 12 (all combinations of fastText) x 12 (all combinations of Doc2Vec) as indicated by Tables 7.2 and 7.3 under each data subset as well as for both the algorithms.

In the following sections, we present the best performing models in each subset. We present the Accuracy, Precision, Recall, and F1-scores for each model. The values of these scores are in the form of percentages.

For each of the tables in the following section, the specifics of the embedding models have been indicated in the ‘Notes’ column where E stands for number of epochs, D stands for number of dimensions, N stands for word n-grams, and dm stands for ‘distributed memory’, where a value of 1 implies PV-DM, whereas 0 means distributed bag of words (PV-DBOW).

Full-text (Half data)

To begin with, we experimented with half the training set and trained multiple models with different combinations of the fastText and Doc2Vec embeddings.

Table 8.8 represents the scores obtained on this dataset for the RF and SVM models with different embeddings. From the table, we see that the SVM model consistently outperforms the RF model in each case. This is contrary to the base results we had as indicated in Tables 8.3 and 8.4 where the RF models outperformed the SVM models. Further details about the top 10 best performing models in each case have been added to Appendix Tables B.1, B.2, B.3, B.4, B.5, and B.5

Our best performing model across the cases is an SVM embedding using the PSU + Illinois dataset. The fastText parameters are 50 epochs, 5 N word n-grams, and 200 dimensions. Similarly, the Doc2Vec parameters are 0 DM (which indicates PV-DBOW), 200 dimensions, and 25 epochs. This model achieves an F1-score of 65.5.

Full-text (All data)

In an attempt to get a better sense of the effect of the number of samples in the training set, we experimented with all of the data in the training set and trained multiple models with different combinations of the fastText and Doc2Vec embeddings.

Table 8.8: Performance of best RF and SVM models on the Full-text (Half data)

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	38.4	56.7	41	46.6	PSU-Illinois embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)
SVM	53.3	75.1	59.2	65.5	PSU-Illinois embeddings (fastText: 50 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	37.8	57	41	46.7	VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	52.4	74.9	58.5	64.9	VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	39.2	56.4	41.5	47	PSU-Illinois + VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 100 D, 25 E)
SVM	52.7	75.1	58.8	65.1	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)

Table 8.9 represents the scores obtained on this dataset for the RF and SVM models with different embeddings. In keeping with the results obtained on the ‘Full-text (half data)’ data subset, the SVM models outperform the RF models. We also see an improvement in the F1-score values for both the RF and SVM models. Further details about the top 10 best performing models in each case have been added to Appendix Tables B.7, B.8, B.11, B.12, B.9, and B.10.

For this data subset, our best performing model across the cases, once again, is an SVM embedding using the PSU + Illinois dataset. The fastText parameters are 50 epochs, 3 N word n-grams, and 200 dimensions. Similarly, the Doc2Vec parameters are 0 DM (which

Table 8.9: Performance of best RF and SVM models on the Full-text (All data)

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	43.1	59.6	46.3	51.3	PSU-Illinois embeddings (fastText: 50 E, 3 N, 100 D Doc2Vec: 0 dm, 100 D, 25 E)
SVM	58.4	77.2	63.1	68.9	PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	42.9	59.5	45.7	50.8	VT embeddings (fastText: 100 E, 3 N, 200 D Doc2Vec: 0 dm, 100 D, 100 E)
SVM	57.9	77	62.6	68.5	VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	43.7	59.2	45.2	50.4	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 100 D Doc2Vec: 0 dm, 100 D, 50 E)
SVM	57.5	76.9	62.4	68.3	PSU-Illinois + VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)

indicates PV-DBOW), 200 dimensions, and 25 epochs. This model achieves an F1-score of 68.9.

These results indicate that the 200 dimension representation of the data appears to be better than the 100 dimension representation for the SVM models.

Chapter subset data

Since we wanted to gauge the impact of chapter summaries combined together to form the full-text for training on the performance of our classification model, we first train our Machine Learning models using the full-text data from this data subset and train models using the

combined summary data. We then compare the performance of these two methodologies. Again, all the experiments performed for this data subset were with all possible combinations of embedding models to ensure that we get the best possible F1-score.

Using full-text data

For this set of experiments, we use the full-text data of the ETDs present in this subset. Table 8.10 represents the scores obtained on this dataset for the RF and SVM models with different embeddings.

Table 8.10: Performance of best RF and SVM models on the chapter subset data (full-text data)

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	46.3	57.7	46.5	50.8	PSU-Illinois embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	63.2	74.3	64.9	68.9	PSU-Illinois embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	46.5	58	46.7	50.8	VT embeddings (fastText: 100 E, 5 N, 100 D Doc2Vec: 0 dm, 100 D, 100 E)
SVM	62.5	74.7	64.1	68.5	VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	46.6	56.7	46.6	50.4	PSU-Illinois + VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 0 dm, 100 D, 50 E)
SVM	63.5	76	65	69.5	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 50 E)

Further details about the top 10 best performing models in each case have been added to

Appendix Tables [B.13](#), [B.14](#), [B.17](#), [B.18](#), [B.15](#), and [B.16](#).

For this data subset as well, the SVM models outperform the RF models. Our best performing model across each case is an SVM embedding using the PSU + Illinois + VT dataset which is unlike our previous result where the best embedding dataset was the PSU + Illinois dataset. The fastText parameters are 50 epochs, 3 N word n-grams, and 200 dimensions. Similarly, the Doc2Vec parameters are 0 DM (which indicates PV-DBOW), 200 dimensions, and 50 epochs. This model achieves an F1-score of 69.5 which is better than all the F1-scores previously recorded.

These results further corroborate that the 200 dimension representation of the data appears to be better than the 100 dimension representation for our SVM models.

Using summary data

Based on the results obtained on the chapter subset dataset (full-text data), we selected the fastText and Doc2Vec parameters of the top 5 best performing models in each case and trained different Machine Learning models using the summarization techniques illustrated in Table [7.10](#). Here, we present the results obtained for each of these models.

Table [8.11](#) represents the scores obtained on this dataset for the RF and SVM models using `gensim`'s TextRank Generated Summary with ratio of 0.2.

Appendix Tables [B.19](#), [B.20](#), [B.23](#), [B.24](#), [B.21](#), and [B.22](#) represent the scores of the top 5 models obtained on the chapter subset dataset for the RF and SVM models with different embeddings trained on the combined summaries of the chapters of the ETDs present in this subset using `gensim`'s TextRank Generated Summary with ratio of 0.2.

Table [8.12](#) represents the scores obtained on this dataset for the RF and SVM models using `gensim`'s TextRank Generated Summary with 100 words.

Table 8.11: Performance of best RF and SVM models on the chapter subset data (summary data) using `gensim`'s TextRank Generated Summary with ratio of 0.2

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	46.6	56.6	46.452	50.295	PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 100 E)
SVM	62	74.8	64.0	68.6	PSU-Illinois embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 100 D, 25 E)
RF	45.4	55.6	44.9	49	VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 1 dm, 100 D, 25 E)
SVM	63.2	75	64.7	69	VT embeddings (fastText: 100 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	45	56.4	45.3	49.4	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 100 D, 25 E)
SVM	62.6	74.8	64.2	68.6	PSU-Illinois + VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 50 E)

Appendix Tables [B.25](#), [B.26](#), [B.29](#), [B.30](#), [B.27](#), and [B.28](#) represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings trained on the combined summaries of the chapters of the ETDs present in this subset using `gensim`'s TextRank Generated Summary with 100 words.

Table [8.13](#) represents the scores obtained on this dataset for the RF and SVM models using `sumy`'s LexRank Generated Summary.

Appendix Tables [B.31](#), [B.32](#), [B.35](#), [B.36](#), [B.33](#), and [B.34](#) represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings trained

Table 8.12: Performance of best RF and SVM models on the chapter subset data (summary data) using `gensim`'s TextRank Generated Summary with 100 words

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	45.8	57.3	45.6	49.7	PSU-Illinois embeddings (fastText: 100 E, 3 N, 100 D Doc2Vec: 0 dm, 100 D, 50 E)
SVM	63.5	73.6	64.3	68.3	PSU-Illinois embeddings (fastText: 100 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	44.1	55.8	44.4	48.5	VT embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 0 dm, 100 D, 100 E)
SVM	63.4	74.5	64.8	69	VT embeddings (fastText: 50 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	43.4	55	44	48.1	PSU-Illinois + VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 100 D, 100 E)
SVM	62.0	73.2	63.4	67.6	PSU-Illinois + VT embeddings (fastText: 50 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)

on the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s LexRank Generated Summary.

Table 8.14 represents the scores obtained on this dataset for the RF and SVM models using `sumy`'s Generated Summary using Luhn's Algorithm.

Appendix Tables B.37, B.38, B.41, B.42, B.39, and B.40 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings trained on the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using Luhn's Algorithm.

Table 8.13: Performance of best RF and SVM models on the chapter subset data (summary data) using `sumy`'s LexRank Generated Summary

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	45.5	56.4	45.5	49.3	PSU-Illinois embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	62	73.7	63.5	67.8	PSU-Illinois embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	43.6	56.2	43.6	48.3	VT embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 0 dm, 100 D, 100 E)
SVM	62.6	74.3	63.9	68.1	VT embeddings (fastText: 100 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	45.7	56.4	44.4	48.7	PSU-Illinois + VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 0 dm, 100 D, 50 E)
SVM	62.5	73.9	63.6	67.7	PSU-Illinois + VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)

Table 8.15 represents the scores obtained on this dataset for the RF and SVM models using `sumy`'s Generated Summary using LSA.

Appendix Tables B.43, B.44, B.47, B.48, B.45, and B.46 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings trained on the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using LSA.

Table 8.16 represents the scores obtained on this dataset for the RF and SVM models using `sumy`'s Generated Summary using LSA with stopwords.

Table 8.14: Performance of best RF and SVM models on the chapter subset data (summary data) using `sumy`'s Generated Summary using Luhn's Algorithm

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	44.9	55.2	44.6	48.5	PSU-Illinois embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	62.8	73	60	67.8	PSU-Illinois embeddings (fastText: 100 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	45.2	57.3	45.3	50	VT embeddings (fastText: 25 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	64.4	76	65.7	70.1	VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	45	55.6	45	49.1	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 100 D, 25 E)
SVM	63	74.3	64.4	68.5	PSU-Illinois + VT embeddings (fastText: 50 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)

Appendix Tables [B.49](#), [B.50](#), [B.53](#), [B.54](#), [B.51](#), and [B.52](#) represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings trained on the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using LSA with stopwords.

From the results presented in this section, we see that the SVM models continue to outperform the RF models. Similarly, we see that most of the extractive summarization algorithms result in F-1 scores similar to those with the full-text data. However, two of the summarization techniques, `sumy`'s Generated Summary using LSA and `sumy`'s Generated Summary

Table 8.15: Performance of best RF and SVM models on the chapter subset data (summary data) using `sumy`'s Generated Summary using LSA

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	44.5	55.2	44.3	48.3	PSU-Illinois embeddings (fastText: 100 E, 3 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	62.3	74	64.6	68.6	PSU-Illinois embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	43.3	55	44.2	48.4	VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 1 dm, 100 D, 25 E)
SVM	63.9	75.7	65.8	70	VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	40	55.9	44.3	48.7	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 100 D, 25 E)
SVM	60	74.7	65.1	69.1	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 50 E)

using Luhn's Algorithm, yield F1-scores of 70 and 70.1, respectively for the SVM models. The fastText parameters of the best model are 25 epochs, 3 N word n-grams, and 200 dimensions. Similarly, the Doc2Vec parameters are 0 DM (which indicates PV-DBOW), 200 dimensions, and 25 epochs.

On the other hand, the best F-1 score among all of the RF models trained using the summary data, yielded a score of 50.3 which was less than the best F-1 score of 50.8 obtained using the full-text data.

Table 8.16: Performance of best RF and SVM models on the chapter subset data (summary data) using `sumy`'s Generated Summary using LSA with stopwords

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	45.2	55.3	44.5	48.6	PSU-Illinois embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	61.9	73.0	63.4	67.5	PSU-Illinois embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	40	55.4	44.4	48.4	VT embeddings (fastText: 25 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)
SVM	62.9	73.4	63.9	68	VT embeddings (fastText: 100 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)
RF	44.9	56	44.9	49	PSU-Illinois + VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 0 dm, 100 D, 50 E)
SVM	61.6	73.4	63.5	67.6	PSU-Illinois + VT embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 0 dm, 200 D, 50 E)

Analysis

Table 8.17 gives the best scoring ML models and their performance scores.

It is interesting to note, that for the RF models, the best Doc2Vec model comprised of 100D vectors, contrary to our observation in case of SVM models that performed best with 200D vector representations.

The best model for our ‘Full-text’ dataset is an SVM with an F-1 score of 68.9 using the PSU-Illinois custom trained embeddings. For our ‘Chapter subset dataset’ it is an SVM trained

Table 8.17: Performance of various Machine Learning experimental setups

ML Algorithm	Accuracy	Precision	Recall	F1-score	Notes
RF	43	59.6	46.3	51.3	Full-text (All data) + PSU-Illinois embeddings
SVM	58.4	77.2	63.1	68.9	Full-text (All data) + PSU-Illinois embeddings
RF	46.5	58	46.7	50.8	Chapter subset data (full-text data) + VT embeddings
SVM	64.4	76	65.7	70.1	Chapter subset data (summary data using Luhn’s Algorithm) + VT embeddings

on the combined chapter summaries generated using Luhn’s Algorithm with an F-1 score of 70.1 using the custom trained VT embeddings. In both cases, these values are greater than those obtained for the pre-trained models. This answers **RQ1**; our custom embeddings give better F-1 scores in case of the Machine Learning models. Similarly, as an answer to **RQ2**, we see that the extractive summaries generated using Luhn’s Algorithm resulted in models that had a greater F-1 score when compared to those trained using the full-text.

Chapter-level Labels

To answer **RQ4**, we will present an Exact Match score, Hamming loss, and Jaccard index values that will give us the value of the overlap of the multi-labels. This will let us know if there are new labels added by our models’ predictions of the chapter labels.

We present the chapter labels generated for the best SVM model described in Table 8.17. We select the model trained on the chapter subset data with summaries generated using `sumy`’s Luhn’s Algorithm and embeddings trained on the VT dataset.

We perform transfer learning from the full-text to the chapter-level. Here, we compare the predicted model results at the chapter level against the ground truth labels present at the full-text level.

Table 8.18: Performance of best Machine Learning algorithms at the chapter-level

F1-score	Hamming loss	Jaccard index	Exact match score	Notes
64.7	0.027	52.3	54.2	Full-text (All data) + PSU-Illinois embeddings
65.5	0.026	50.9	61.5	Chapter subset data (summary data using Luhn’s Algorithm) + VT embeddings

Department	Keywords	chapter_text	Expected_Labels	Predicted_Labels
Philosophy	Computer modeling, Glass, Quasi-one-dimension, Soft matter	<p>Chapter 2: Figure-8 Model</p> <p>In this chapter we describe calculations of slow dynamics in the simplest quasi-one-dimensional model—the “Figure-8” model [62]. In later chapters, we will apply and generalize the concepts developed in this chapter to calculate analytically the structural relaxation in quasi-one-dimensional models with an arbitrary number of junctions.</p> <p>Glassy dynamics is characterized by several common features [3]. For example, the viscosity and structural relaxation times diverge super-Arrheniusly near the glass transition, and the long-time self-diffusion constant becomes extremely small. Correspondingly, a plateau develops in the particle mean-square displacement (MSD). As the system approaches the glass transition, the plateau extends to longer and longer times [31, 65, 66], signaling kinetic arrest associated with the formation of cages of neighbors around each particle. Structural relaxation occurs through a series of rare cage-breaking events, in which particles in the nearest-neighbor and further shells move cooperatively so that caged particles can escape [16]. The last feature is emphasized in Fig. 2.1, where we show results from simulations of a 50 – 50 bidisperse mixture of hard disks with diameter ratio 1.4 in the supercooled liquid regime. In Fig. 2.1 (a), we plot displacements of a focus particle as it moves between three cages a, 8, and 7. In Fig. 2.1 (b)-(f), we monitor the focus particle and its neighbors as it breaks out of cage a and becomes trapped in B. Two processes are required for cage-breaking to occur.</p>	Statistics	Statistics, Computer science
Philosophy	Computer modeling, Glass, Quasi-one-dimension, Soft matter	<p>Chapter 3: Multi-Junction Models</p> <p>In the previous chapter, we focused on the figure-8 system with a single junction [62] and N hard rods. We found that the structural relaxation time diverges as a power-law with increasing packing fraction, ϕ_S is the packing fraction at which kinetic arrest occurs and $a = N/2 - 1$ can be calculated exactly. The most likely configurations in figure-8 systems are those with $IV/2$ particles in both the top and bottom lobes, and no particles in the junction. However, to is controlled by rare ‘junction-crossing’ events, in which a particle from the bottom (top) lobe, crosses the junction, enters the top (bottom) lobe from one side of the junction, and another particle exits the top (bottom) lobe and enters the bottom (top) lobe from the other side of the junction. Thus, to undergo structural relaxation, the system transitions from a relaxed configuration with half of the particles in each lobe to a rare, squeezed configuration with an extra particle in one of the lobes, and back to a relaxed configuration.</p>	Statistics	Statistics, Mathematics
Philosophy	Computer modeling, Glass, Quasi-one-dimension, Soft matter	<p>Chapter 4: Microstate Network</p> <p>4.1 Graph-theoretic description of dynamics in quasi-one- dimensional models</p> <p>To describe the structural relaxation mechanisms in Q1-D systems, it is convenient to map all of the configurations of the system onto a set of discrete microstates, which provides a complete description of the system. A microstate can be defined by a specific series of $3J$ integers (i.e. the microstate b identifier). The first and last of these integers (labelled l^* and l^*) represent the number of particles in the end lobes. The second and second to last integers (labeled by J) represent the number of particles that occupy the intersections adjacent to the end lobes. Since the intersections are the same size as the particles, a minimum of zero and a maximum of two particles can occupy each intersection. (If any portion of the particle is in the intersection, it is considered occupied.)</p>	Statistics	Statistics, Computer science

Figure 8.12: Chapter-level labels for the ETD titled ‘Quasi-one-dimensional models for glassy dynamics’

Table 8.18 presents the evaluation of the two best SVM models. As can be seen from this table, the loss of labels between the full-text and the chapters tends towards zero which indicates that most of the predicted labels contain the ground truth labels as well. Similarly, the strict exact match values are 54.2 and 61.5, respectively. This indicates that over 50% of the labels exactly match in both cases. This does, however, indicate to us that with respect

to **RQ4**, there does exist a difference between the labels predicted at the chapter-level when compared to the labels associated with the ETD as a whole.

Figure 8.12 depicts the chapter-level labels generated for an ETD entitled ‘Quasi-one-dimensional models for glassy dynamics’. While the full-text label is ‘Statistics’, we see that the labels generated for Chapter 2 and 4 include ‘Statistics, Computer science’ whereas the label for Chapter 3 has been predicted to be ‘Statistics, Mathematics’. These labels were predicted using the SVM model trained on the Full-text (All data).

	precision	recall	f1-score
Statistics	0.67	0.67	0.67
Computer science	0.77	0.85	0.81
Electrical engineering	0.69	0.94	0.79
Civil engineering	0.82	0.94	0.88
Mathematics	0.77	0.82	0.79
Public administration	0.55	0.82	0.66
Ecology	0.71	0.77	0.74
Computer Engineering	0.76	0.56	0.65
Adult education	0.48	0.45	0.47
Secondary education	0.34	0.51	0.41
Mechanical engineering	0.69	0.58	0.63
Industrial engineering	0.27	0.39	0.32
Aerospace engineering	0.70	0.89	0.78
Molecular biology	0.82	0.95	0.88
Educational psychology	0.43	0.39	0.41
Environmental science	0.80	0.86	0.83
Chemical engineering	0.40	0.40	0.40
Higher education	0.47	0.55	0.50
Materials science	0.68	0.61	0.64
Educational leadership	0.52	0.74	0.61
Special education	0.78	0.61	0.69
Teacher education	0.40	0.53	0.46
Marketing	0.85	0.89	0.87
Biomedical engineering	0.77	0.75	0.76
Organic chemistry	0.67	0.95	0.79
Elementary education	0.39	0.44	0.41
Occupational psychology	0.73	0.80	0.76
Forestry	0.99	0.65	0.79

Figure 8.13: Performance of SVM trained on Chapter subset data (summary data using Luhn’s Algorithm) per subject category at the chapter-level

Figure 8.13 represents the performance of our best performing SVM model trained using the Chapter subset data with summaries generated using Luhn’s Algorithm. As can be seen from this figure, the categories with the lowest F-1 score include ‘Industrial Engineering’ with a

score of 32, followed by ‘Chemical Engineering’ with an F-1 score of 40. On further inspection, we found that ‘Industrial Engineering’ was alternatively being classified as ‘Aerospace Engineering’ and ‘Chemical Engineering’ was alternatively being classified as ‘Biomedical Engineering’ and ‘Molecular Biology’. We suspect that this was due to similar words that were used in case of each of these subject categories which indicates that these disciplines may co-occur together more frequently.

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry	
Statistics	229	1	0	0	0	2	0	0	0	0	0	43	0	0	10	20	0	0	0	0	0	0	0	0	0	0	0	38	0
Computer science	10	622	4	0	0	52	0	2	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	1	0
Electrical engineering	0	23	480	0	0	4	0	1	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
Civil engineering	1	0	0	707	0	7	0	0	0	0	7	0	0	0	0	9	3	0	12	0	0	0	4	0	0	0	0	0	0
Mathematics	1	0	0	0	122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	25	0	0	
Public administration	53	0	0	1	0	458	0	6	1	0	0	0	0	0	3	13	0	0	0	9	0	0	5	0	0	0	7	1	
Ecology	0	0	0	0	0	2	401	0	0	0	0	0	0	4	0	18	0	0	0	0	0	0	0	7	89	0	0	0	
Computer Engineering	8	130	93	0	0	28	0	12	0	0	1	6	5	4	0	0	0	0	6	0	0	0	9	0	0	0	0	0	
Adult education	0	1	0	0	0	90	0	0	302	3	0	0	5	0	71	0	0	30	0	98	2	1	0	0	0	4	59	0	
Secondary education	0	0	0	0	0	0	0	13	234	0	0	0	0	0	16	0	0	67	0	27	12	40	0	0	0	50	0	0	
Mechanical engineering	0	0	19	111	25	0	0	0	0	0	472	77	28	0	0	0	3	0	0	0	0	0	0	77	0	0	0	0	
Industrial engineering	1	12	7	10	0	0	0	0	0	0	13	54	35	0	0	0	0	0	0	0	0	0	1	0	0	0	6	0	
Aerospace engineering	0	0	1	0	0	1	0	0	0	0	103	1	208	0	0	2	2	0	0	0	0	0	0	0	0	0	4	0	
Molecular biology	0	0	0	0	0	0	0	0	0	0	0	0	0	520	0	0	28	0	0	0	0	0	0	0	0	0	0	0	

Figure 8.14: Part 1: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the SVM model trained using the Chapter subset data (summary data using Luhn’s Algorithm)

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry
Educational psychology	0	6	0	0	0	16	0	0	99	223	0	0	0	0	342	2	0	47	0	20	26	46	1	0	0	5	37	0
Environmental science	3	0	0	14	0	11	44	0	0	0	6	0	0	2	0	549	4	0	4	0	0	0	0	0	21	0	0	0
Chemical engineering	1	0	3	2	7	0	0	0	0	0	27	0	0	27	0	0	53	0	9	0	0	0	0	17	7	0	0	0
Higher education	5	4	0	0	0	51	0	0	38	45	0	1	0	0	80	0	0	397	0	143	0	54	3	0	2	1	3	0
Materials science	0	0	20	4	4	0	0	0	0	0	53	0	0	0	0	51	0	195	0	0	0	0	0	7	42	0	0	0
Educational leadership	1	0	0	0	0	25	0	4	52	0	0	0	0	0	17	2	0	128	0	863	3	77	0	0	0	16	28	0
Special education	0	0	0	0	0	14	0	0	54	16	0	0	0	0	107	0	0	6	0	48	404	72	0	0	0	13	2	0
Teacher education	0	0	0	0	0	1	0	0	43	68	0	0	0	0	44	0	0	2	0	135	14	282	0	0	0	37	11	0
Marketing	7	0	0	3	0	7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	355	0	0	25	0
Biomedical engineering	11	2	36	0	0	0	0	3	0	0	0	0	3	74	0	5	2	40	0	0	0	0	0	393	0	0	0	0
Organic chemistry	0	0	0	7	0	0	0	0	0	0	0	0	0	1	7	0	0	0	0	0	0	0	0	2	331	0	0	0
Elementary education	3	0	0	0	0	2	0	0	58	39	0	0	0	0	48	11	0	1	0	54	50	54	3	0	0	185	2	0
Occupational psychology	10	0	0	0	0	60	0	0	10	0	0	2	1	0	63	2	0	0	4	2	0	3	0	0	0	629	0	
Forestry	0	0	0	0	0	7	120	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	10

Figure 8.15: Part 2: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the SVM model trained using the Chapter subset data (summary data using Luhn’s Algorithm)

Figures 8.14 and 8.15 indicate the number of times subject categories were commonly predicted to co-occur together. There were a total of 15047 chapters as part of this test set. These results have been presented for our best performing SVM model trained using the Chapter subset data with summaries generated using Luhn’s Algorithm.

Figure 8.16 represents the number of STEM and non-STEM subject categories that were

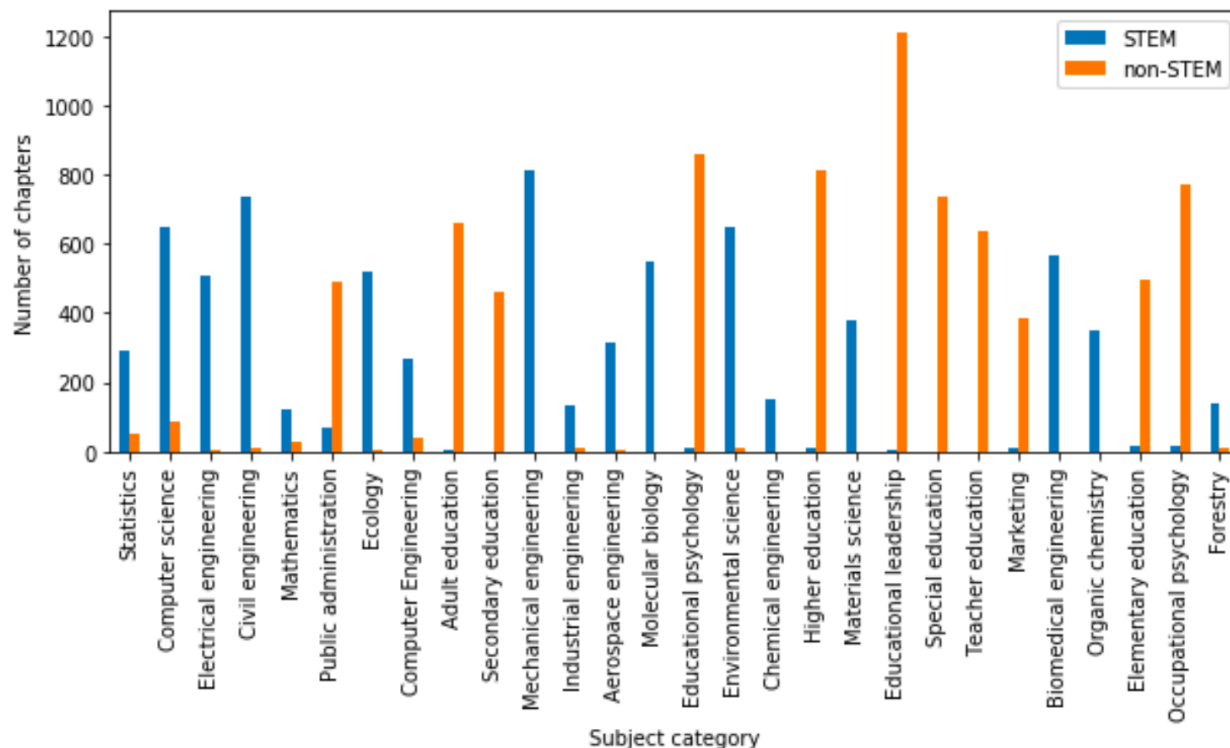


Figure 8.16: STEM and non-STEM categories predicted for chapters of ETDs by the SVM model trained using the Chapter subset data (summary data using Luhn’s Algorithm)

assigned to each of the individual subject categories. The values on the X-axis represent the ground truth subject categories for each of the chapters of the ETDs considered. As can be seen from this figure, categories such as ‘Educational leadership’, and ‘Educational psychology’ which are non-STEM categories contain chapters that also predominantly belong to non-STEM categories.

8.2.2 Deep Learning Models

For the purpose of this study, the Machine Learning algorithms that we focus on are LSTM, GRU, BiLSTM, and BiGRU. For the first part of this study, we only consider the performance of the LSTM models. Then, we fine tune the LSTM models and compare the performance against the other models.

Pretrained Embeddings

Table 8.19: Performance of LSTM built using pre-trained embeddings on the Full-text (half data)

Model parameters	Accuracy	Precision	Recall	F1-score
500 E, 64 b, validation_split=0.2	28.9	62.8	37.2	45.3
500 E, 64 b	31.3	61.4	35.3	40.5
500 E, 128 b, validation_split=0.2	33.8	62.9	39.1	46.8
500 E, 128 b	41.3	56.3	44.6	48.8
1000 E, 128 b	37.2	59.9	43.2	49.4
2500 E, 128 b	36.4	58.9	43.1	48.8
5000 E, 128 b	22.5	60.5	24	31.8

In this section, we present the results obtained for the LSTM models when we use the pre-trained fastText and Doc2Vec embeddings that were trained on the Wikipedia and Common Crawl datasets. We train and test these embeddings on the Full-text (half data) data subset with different hyperparameter values. In the tables of this section, ‘E’ represents number of epochs while ‘b’ represents the batch size. All of these models had a state size of 512 and a dropout of 0.2. Table 8.19 presents the performance of the LSTM built using pre-trained embeddings. Since the performance of the models without a validation split was better than the one with a validation split, we performed more analysis on the former setup. According to the table, after 1000 Epochs, the model performance starts to dwindle. This could be due to overfitting.

According to Figure 8.17, the accuracy and loss values do not have any substantial gain after 1000 epochs. Since the training time for 1000 epochs was large, we limited the number of epochs while identifying the best possible custom embeddings to 15 epochs. Additionally, we change the metric used to evaluate the performance while fitting the model to Precision, Recall, and F-1 scores since there exists an imbalance in our dataset. Table 8.20 presents

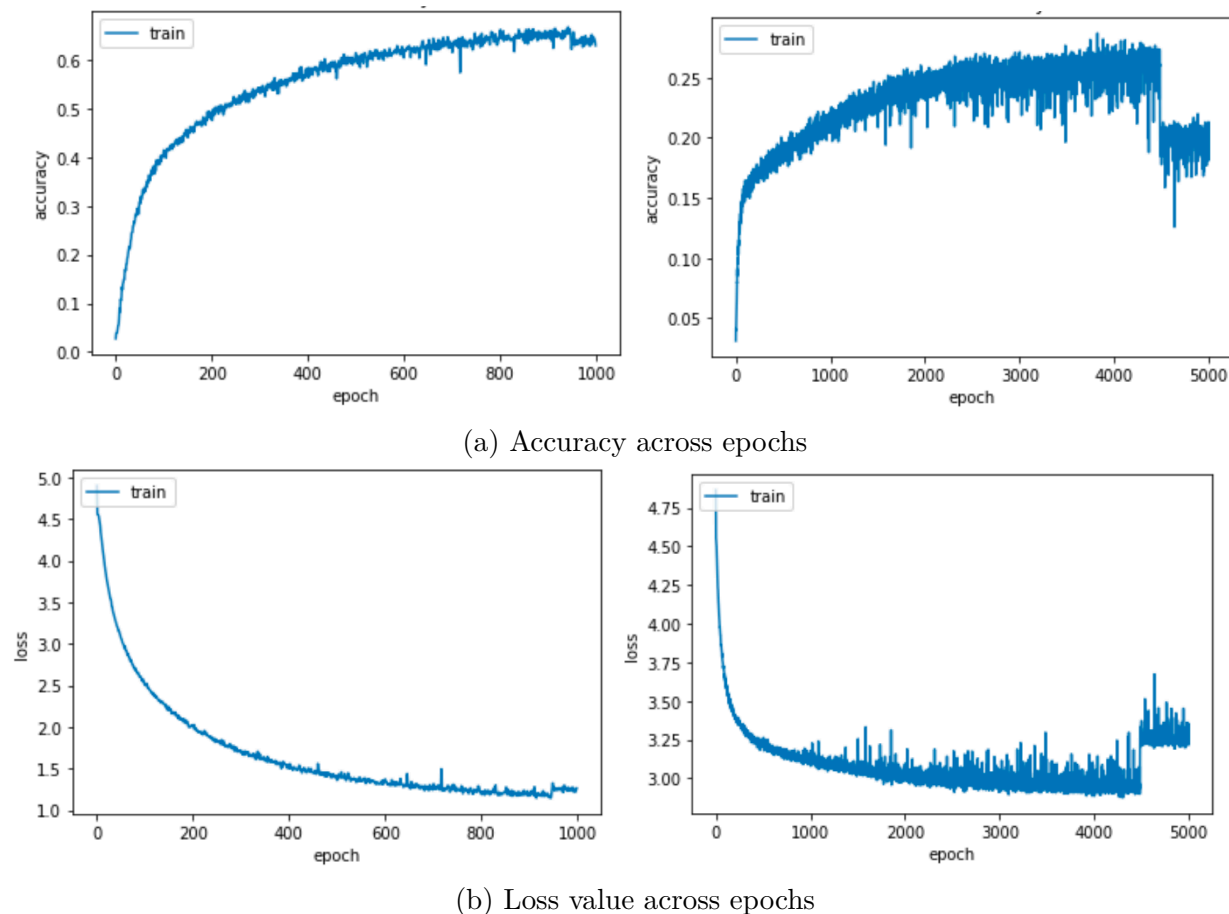


Figure 8.17: Visual representation of accuracy and loss values across epochs

the scores of the LSTMs trained with these hyperparameters and will be used to compare the performance of our models using custom embeddings.

Table 8.20: Performance of LSTM built using pre-trained embeddings

Model parameters	Accuracy	Precision	Recall	F1-score	Notes
15 E, 512 b	10.4	65.8	10.4	17	Full-text (half data)
15 E, 512 b	36.3	70.5	37.3	44.9	Full-text (all data)

Custom Embeddings

Based on the performance of the SVM and RF models trained, we picked the fastText and Doc2Vec parameters of the top 5 models in each subset and trained our deep learning models with those parameters. This section is divided in a similar manner as the section discussing the Machine Learning models. For the first portion of these results, we only make comparisons between the LSTM models. As mentioned earlier, in an attempt to identify the best embeddings using LSTMs, we ran our models for a limited number of epochs. After selecting our best performing models, we tuned the hyperparameters and then compared the tuned LSTM with the performance of our ML models.

Full-text (Half data)

Table 8.21 represents the scores obtained on this dataset for the LSTM trained using with different embeddings. Further details about the top 10 best performing models in each case have been added to Appendix Tables B.55, B.57, and B.56.

Each of these models were trained with a state size of 1024, a dropout of 0.2, batch size of 512, and 15 epochs.

Table 8.21: Performance of best LSTM models on the Full-text (Half data)

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	30.5	80.4	33.1	41.6
VT embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 1 dm, 200 D, 50 E)	38.4	77.1	40.3	48.8
PSU-Illinois + VT embeddings (fast-Text: 25 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	34.8	67.7	37.0	44.1

Our best performing model is an LSTM using embeddings built from the VT dataset. The

fastText parameters are 25 epochs, 3 N word n-grams, and 200 dimensions. Similarly, the Doc2Vec parameters are 1 DM (which indicates PV-DM), 200 dimensions, and 50 epochs. This model achieves an F1-score of 48.8. It is interesting to note that these parameter values differ from those of the best model in case of the Machine Learning counterpart of this experimental setup.

Full-text (All data)

Table 8.22 represents the scores obtained on this dataset for the LSTM trained with different embeddings. Further details about the top 10 best performing models in each case have been added to Appendix Tables B.58, B.59, and B.60.

Similar to the hyperparameter values in the previous case, each of these models was trained with a state size of 1024, a dropout of 0.2, batch size of 512, and 15 epochs.

Table 8.22: Performance of best LSTM models on the Full-text (All data)

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 25 E)	52.1	76.8	54.2	60.7
VT embeddings (fastText: 100 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	50.3	72.3	52.8	58.6
PSU-Illinois + VT embeddings (fast-Text: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 100 D, 50 E)	48.1	74.4	50.5	56.7

Our best performing model is an LSTM using embeddings built from the PSU + Illinois dataset. The fastText parameters are 25 epochs, 3 N word n-grams, and 200 dimensions. Similarly, the Doc2Vec parameters are 1 DM (which indicates PV-DM), 100 dimensions, and 25 epochs. This model achieves an F1-score of 60.7 which is a considerable improvement over the 48.8 F1-score obtained with half the training data. This indicates that our model has scope to improve with more data and that it hasn't suffered from overfitting. It is interesting

to note that the `fastText` parameter values are the same as the *Full-text (half-data)* case but the `Doc2Vec` values differ.

Chapter subset data

Similar to the experiments described in 8.2.1, we present the results obtained for the deep learning models. Each of these models were trained with a state size of 1024, a dropout of 0.2, batch size of 512, and 25 epochs.

Using full-text data

For this set of experiments, we use the full-text data of the ETDs present in this subset. Table 8.23 represents the scores obtained on this dataset for the LSTM models with different embeddings.

Further details about the top 10 best LSTM models have been added to Appendix Tables B.61, B.62, and B.63.

Table 8.23: Performance of best LSTM models on the chapter subset data (full-text data)

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 100 E)	60	74.1	60	64.4
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	61.1	73.1	61.1	64.8
PSU-Illinois + VT embeddings (fast-Text: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	56.4	71.9	56.9	61.6

Our best performing model is an LSTM using embeddings built from the VT dataset. It has a marginally better F1-score when compared to the model built using the PSU + Illinois dataset. The `fastText` parameters are 25 epochs, 3 N word n-grams, and 100 dimensions. Similarly, the `Doc2Vec` parameters are 1 DM (which indicates PV-DM), 100 dimensions, and

100 epochs. This model achieves an F1-score of 64.8.

Using summary data

Once again, based on the results obtained on the chapter subset dataset (full-text data), we selected the fastText and Doc2Vec parameters of the top 5 best performing models in each case and trained LSTM models using summarization techniques illustrated in Table 7.10. Here we present the results obtained for each of these models.

Table 8.24 represents the scores obtained on this dataset for the LSTM models using gensim's TextRank Generated Summary with ratio of 0.2.

Table 8.24: Performance of best LSTM models on the chapter subset data (summary data) using gensim's TextRank Generated Summary with ratio of 0.2

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 1 dm, 100 D, 50 E)	58.9	72.6	59	63.3
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	61.5	72.0	61.6	64.8
PSU-Illinois + VT embeddings (fast-Text: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	57.8	70.1	57.8	61.4

Further details about the top 5 best performing LSTM models has been added to Appendix Tables B.64, B.65, and B.66.

Table 8.25 represents the scores obtained on this dataset for the LSTM models using gensim's TextRank Generated Summary with 100 words.

Further details about the top 5 best performing LSTM models has been added to Appendix Tables B.67, B.68, and B.69.

Table 8.26 represents the scores obtained on this dataset for the LSTM models using sumy's

Table 8.25: Performance of best LSTM models on the chapter subset data (summary data) using `gensim`'s TextRank Generated Summary with 100 words

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 25 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 50 E)	57.1	74.1	57.1	61.8
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	57.4	70.8	57.4	61.3
PSU-Illinois + VT embeddings (fastText: 50 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 50 E)	51.3	77.3	51.3	57.9

LexRank Generated Summary.

Table 8.26: Performance of best LSTM models on the chapter subset data (summary data) using `sumy`'s LexRank Generated Summary

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 100 E)	53.7	73.8	53.4	58.3
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	54	72.8	54.2	59
PSU-Illinois + VT embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	55.7	68.4	55.7	59.3

Further details about the top 5 best performing LSTM models have been added to Appendix Tables [B.70](#), [B.71](#), and [B.72](#).

Table [8.27](#) represents the scores obtained on this dataset for the LSTM models using `sumy`'s Generated Summary using Luhn's Algorithm.

Further details about the top 5 best performing LSTM models have been added to Appendix Tables [B.73](#), [B.74](#), and [B.75](#).

Table 8.27: Performance of best LSTM models on the chapter subset data (summary data) using `sumy`'s Generated Summary using Luhn's Algorithm

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 100 E)	53.7	77	53.7	59.5
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	58.7	74.3	58.8	63.8
PSU-Illinois + VT embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	55.8	72.1	55.8	60.1

Table 8.28 represents the scores obtained on this dataset for the LSTM models using `sumy`'s Generated Summary using LSA.

Table 8.28: Performance of best LSTM models on the chapter subset data (summary data) using `sumy`'s Generated Summary using LSA

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 100 E)	55.803	74.5	55.8	61
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	55.6	73.3	56.3	61.1
PSU-Illinois + VT embeddings (fastText: 25 E, 5 N, 200 D Doc2Vec: 0 dm, 200 D, 25 E)	51.8	74.2	51.8	57.8

Further details about the top 5 best performing LSTM models have been added to Appendix Tables B.76, B.77, and B.78.

Table 8.29 represents the scores obtained on this dataset for the LSTM models using `sumy`'s Generated Summary using LSA with stopwords.

Further details about the top 5 best performing LSTM models have been added to Appendix

Table 8.29: Performance of best LSTM models on the chapter subset data (summary data) using `sumy`'s Generated Summary using LSA with stopwords

Model details	Accuracy	Precision	Recall	F1-score
PSU-Illinois embeddings (fastText: 50 E, 3 N, 200 D Doc2Vec: 1 dm, 100 D, 100 E)	56.5	73.5	56.6	61.1
VT embeddings (fastText: 25 E, 3 N, 100 D Doc2Vec: 1 dm, 200 D, 50 E)	57.6	71.9	57.9	61.8
PSU-Illinois + VT embeddings (fastText: 50 E, 5 N, 100 D Doc2Vec: 1 dm, 100 D, 100 E)	55.2	72.5	55.2	60.4

Tables [B.79](#), [B.80](#), and [B.81](#).

Analysis

Table [8.30](#) gives the performance of the best LSTM models.

Table 8.30: Performance of various deep learning experimental setups

Notes	Accuracy	Precision	Recall	F1-score
Full-text (All data) + PSU-Illinois embeddings	52.1	76.8	54.2	60.7
Chapter subset data (full-text) + VT embeddings	60	74.1	60	64.8
Chapter subset data (summary data using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2) + VT embeddings	61.5	72	61.6	64.8

The best model for our ‘Full-text’ dataset had an F-1 score of 60.7 using the PSU-Illinois custom trained embeddings in 15 epochs. For our ‘Chapter subset dataset’, the model trained using the full-text data obtains a score of 64.8 which marginally outperforms the model trained using summaries generated by `gensim`'s TextRank Algorithm with ratio of

0.2. In both these models, the embedding used was trained on the VT dataset with fastText parameters of 25 epochs, 3 word n-grams, and 100 dimensions, and Doc2Vec parameters of 1 DM (which indicates PV-DM), 200 dimensions, and 50 epochs.

Here again, these values are greater than those obtained for the pre-trained models. This answers **RQ1**; our custom embeddings give better F-1 scores in case of the Machine Learning models.

With respect to **RQ2**, we see that the score obtained for the chapter subset data (full-text data) is only marginally more than the extractive summaries generated using TextRank. Considering that these models were trained for only 25 epochs, we further train these models and perform hyperparameter tuning to better understand which of them obtains a higher F1-score.

We train the chapter subset data (full-text data) and chapter subset data (summary data using gensim's TextRank Generated Summary with ratio of 0.2) models by varying the number of epochs in an attempt to better understand the performance of these two models.

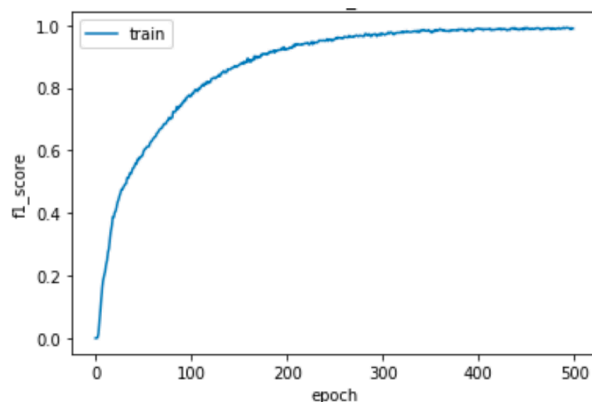
As indicated by Table 8.31, we find that the LSTM trained on the Full-text (All data) for 100 epochs performs better than all of the other models. This model achieves an F1-score of 67.2. We find that the value of the F1-score of this model reduces as we increase the number of epochs. Figure 8.18 visually represents the F1-score and loss value across 500 epochs. Based on this, we decided to select the 100 epochs model as our best model.

With respect to **RQ3**, we find that our deep learning models do not achieve an F1-score that is higher than our best SVM model. The best SVM model on the Full-text (All data) had an F1-score of 68.9. Since this difference is not extremely large, we believe that our deep learning models will be able to perform better with a larger training dataset.

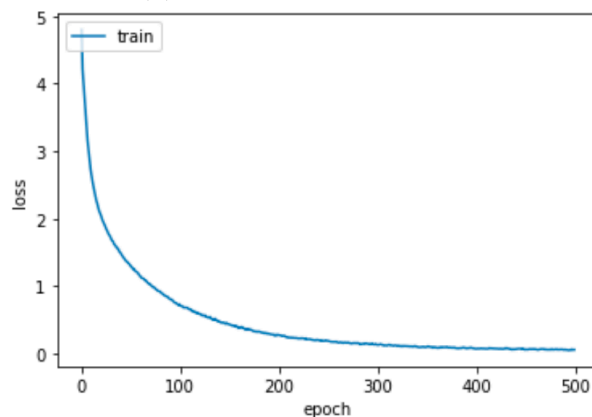
Table 8.31 also indicates that contrary to our findings with the Machine Learning models,

Table 8.31: Performance of various deep learning experimental setups across different number of epochs

Number of Epochs	Accuracy	Precision	Recall	F1-score	Notes
10	32.1	75.6	32.1	40.1	Chapter subset data (full-text)
10	34.3	85.1	34.3	40.5	Chapter subset data (summary data using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2)
100	58.2	75.4	62.3	67.2	Full-text (All data)
100	59.5	71.4	61.8	65.6	Chapter subset data (full-text)
100	58.3	70.9	60.4	64.8	Chapter subset data (summary data using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2)
200	56.2	75.1	61	66.6	Full-text (All data)
200	58	73.3	60.9	66	Chapter subset data (full-text)
200	57	71.2	59.8	64.6	Chapter subset data (summary data using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2)
500	52.7	72.8	57.3	63.6	Full-text (All data)
500	57	69.5	59.2	63.5	Chapter subset data (full-text)
500	57	70.1	58.9	63.7	Chapter subset data (summary data using <code>gensim</code> 's TextRank Generated Summary with ratio of 0.2)



(a) F1-score across epochs



(b) Loss value across epochs

Figure 8.18: Visual representation of F1-score and loss values across epochs for the best performing LSTM on the Full-text (All data)

our deep learning models trained using summary data does not perform as well as the chapter subset data (full-text) model. However, this difference is marginal. With respect to **RQ2**, we conclude that in case of Machine Learning models, our hypothesis holds true, whereas in case of the deep learning models, it does not.

In an attempt to further improve the performance of our models, we train a GRU, Bidirectional LSTM, and Bidirectional GRU for 100 epochs on the Full-text (All data) since our best F1-score was achieved on this data subset. As indicated by Table 8.32, our LSTM model performs the best.

Table 8.32: Performance of various deep learning architectures

Model details	Accuracy	Precision	Recall	F1-score
LSTM	58.2	75.4	62.3	67.2
GRU	56.7	73.7	60.9	66
BiLSTM	56	74	60.7	66.1
BiGRU	56.4	75.9	61.1	66.9

Chapter-level Labels

Once again, considering **RQ4**, we will present an Exact Match score, Hamming loss, and Jaccard index that will give us a measure of the overlap of the multi-labels. This will let us know if there are new labels added by our models' predictions of the chapter labels.

We perform transfer learning from the full-text to the chapter-level. Here we compare the predicted model results at the chapter level against the ground truth labels present at the full-text level.

Table 8.33: Performance of best deep learning algorithms at the chapter-level

F1-score	Hamming loss	Jaccard index	Exact match score	Notes
62.3	0.0286	49.1	55.4	Full-text (All data) + PSU-Illinois embeddings
60	0.0299	44.8	55.7	Chapter subset data (full-text) + VT embeddings

Table 8.33 presents the results obtained for the same. In both cases, our best models are LSTMs. According to this table, the loss of labels between the full-text and the chapters tends towards zero which indicates that most of the predicted labels contain the ground truth labels as well. Similarly, the strict exact match values are 55.4 and 55.7, respectively. This indicates that over 50% of the labels exactly match in both cases. This does, however,

indicate to us that, with respect to **RQ4**, there does exist a difference between the labels predicted at the chapter-level when compared to the labels associated with the ETD as a whole.

	precision	recall	f1-score
Statistics	0.60	0.74	0.66
Computer science	0.83	0.58	0.69
Electrical engineering	0.72	0.86	0.78
Civil engineering	0.84	0.94	0.89
Mathematics	0.86	0.82	0.84
Public administration	0.57	0.76	0.65
Ecology	0.76	0.72	0.74
Computer Engineering	0.26	0.66	0.37
Adult education	0.36	0.61	0.45
Secondary education	0.28	0.55	0.37
Mechanical engineering	0.75	0.74	0.74
Industrial engineering	0.62	0.60	0.61
Aerospace engineering	0.69	0.85	0.76
Molecular biology	0.78	0.95	0.86
Educational psychology	0.43	0.35	0.38
Environmental science	0.64	0.79	0.71
Chemical engineering	0.74	0.67	0.70
Higher education	0.38	0.31	0.34
Materials science	0.77	0.84	0.81
Educational leadership	0.38	0.59	0.47
Special education	0.70	0.76	0.73
Teacher education	0.50	0.33	0.40
Marketing	0.74	0.73	0.74
Biomedical engineering	0.73	0.74	0.73
Organic chemistry	0.74	0.92	0.82
Elementary education	0.30	0.49	0.37
Occupational psychology	0.73	0.82	0.77
Forestry	0.57	0.29	0.38

Figure 8.19: Performance of LSTM trained on Full-text (All data) at the chapter-level

Figure 8.19 represents the performance of our best performing LSTM model trained using Full-text (All data). As can be seen from this figure, the categories with the lowest F-1 score include ‘Higher education’ with a score of 34, followed by ‘Secondary education’ with an F-1 score of 37, followed by ‘Forestry’ with an F-1 score of 38. On further inspection, we found that ‘Higher education’ was alternatively being classified as ‘Adult education’ and ‘Secondary education’, and ‘Secondary education’ was alternatively being classified as ‘Elementary education’ and ‘Adult education’. We think this could be attributed to the similar vocabulary used in these subject categories as well as overlapping departments and author-

provided keywords. ‘Forestry’ was alternatively being classified as ‘Environmental science’ and ‘Ecology’. Once again, on inspection of a few of these alternatively classified chapters, we find that this could be attributed to overlapping vocabulary between these subject categories which indicates that these disciplines may co-occur together more frequently.

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry
Statistics	573	4	0	39	2	26	2	0	1	0	0	9	0	0	4	28	0	0	4	0	1	0	6	0	0	0	73	2
Computer science	46	744	27	0	14	40	0	108	101	23	0	40	7	2	1	0	0	0	0	0	0	0	130	3	0	0	1	0
Electrical engineering	0	4	748	0	0	0	0	63	0	0	20	1	16	0	0	0	14	0	5	0	0	0	0	0	0	0	0	0
Civil engineering	4	0	2	1742	0	6	4	0	0	0	2	11	0	0	0	17	59	0	1	0	0	0	2	0	0	0	0	0
Mathematics	22	0	0	0	256	0	0	0	0	5	6	4	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
Public administration	88	2	0	9	0	1083	0	0	10	0	0	62	6	0	0	41	0	8	0	0	4	0	12	0	0	0	95	0
Ecology	1	0	0	0	0	0	1074	0	0	0	0	0	0	22	0	309	0	0	0	0	1	0	0	85	0	0	9	0
Computer Engineering	6	103	112	7	0	38	0	85	0	0	4	18	0	4	0	0	0	0	6	0	0	0	12	7	0	0	0	0
Adult education	0	0	0	0	0	135	0	0	909	86	0	6	0	25	169	0	0	30	0	26	41	1	2	0	0	9	59	0
Secondary education	0	0	0	0	0	1	0	0	184	840	0	0	0	0	46	2	0	4	0	20	18	9	0	0	0	401	0	0
Mechanical engineering	2	1	7	106	10	0	0	1	0	0	1271	12	56	0	0	13	0	14	0	0	0	0	0	236	0	0	0	0
Industrial engineering	89	1	8	45	0	54	0	2	0	0	15	364	9	0	0	2	1	0	1	0	0	0	5	1	0	0	14	0
Aerospace engineering	8	0	1	5	2	23	0	0	0	0	224	12	389	0	0	2	9	0	0	0	6	0	1	0	0	0	12	0
Molecular biology	2	0	0	0	0	0	1	0	0	0	0	0	0	1202	0	4	0	0	2	0	0	0	4	52	0	0	0	0

Figure 8.20: Part 1: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the LSTM model trained on Full-text (All data)

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry
Educational psychology	0	0	0	0	0	35	0	0	290	155	0	0	0	0	431	0	0	0	7	161	13	2	0	0	0	98	53	0
Environmental science	0	0	0	33	5	27	153	0	2	21	0	2	0	1	5	995	41	0	0	0	0	15	0	14	6	0	0	0
Chemical engineering	5	0	0	71	0	0	0	4	0	0	9	27	0	30	25	564	0	14	0	0	0	0	19	73	0	0	0	0
Higher education	13	0	0	0	2	104	0	0	418	374	0	0	0	5	162	19	0	270	0	9	63	6	30	0	0	60	17	0
Materials science	7	0	17	3	6	0	2	0	0	0	43	0	14	0	0	22	29	0	1009	0	0	0	1	21	74	0	0	0
Educational leadership	24	0	0	0	0	54	0	0	217	511	0	0	0	0	48	0	0	205	0	380	12	8	0	0	0	81	88	0
Special education	0	0	0	0	0	7	0	0	104	123	0	0	0	10	3	0	7	0	95	995	32	0	0	0	0	65	0	0
Teacher education	0	0	0	0	0	0	0	0	130	462	0	0	0	0	37	0	0	1	0	286	45	215	0	0	0	227	0	0
Marketing	9	0	0	0	0	199	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	648	0	0	0	10	0
Biomedical engineering	12	26	119	0	0	0	0	8	2	0	102	20	0	220	0	18	0	13	0	0	0	0	987	1	0	0	0	0
Organic chemistry	0	0	0	6	0	0	3	0	0	4	0	0	0	30	1	11	12	0	3	0	0	0	0	858	0	0	0	0
Elementary education	37	0	0	0	0	1	0	0	26	358	0	0	0	0	14	0	0	0	123	35	8	0	0	0	0	483	0	0
Occupational psychology	1	0	0	0	0	58	0	0	121	7	0	3	0	0	60	1	0	9	0	8	2	0	7	0	0	0	1193	0
Forestry	3	0	0	4	0	21	181	0	0	0	0	0	0	0	0	73	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8.21: Part 2: Interdisciplinarity of chapters of ETDs represented by co-occurrence of subject categories based on predictions made by the LSTM model trained on Full-text (All data)

Figures 8.21 and 8.21 indicate the number of times subject categories were commonly predicted to co-occur together. There were a total of 32607 chapters as part of this test set. These results have been presented for our best performing LSTM model trained using Full-text (All data).

Figure 8.22 represents the number of STEM and non-STEM subject categories that were assigned to each of the individual subject categories. The values on the X-axis represent the

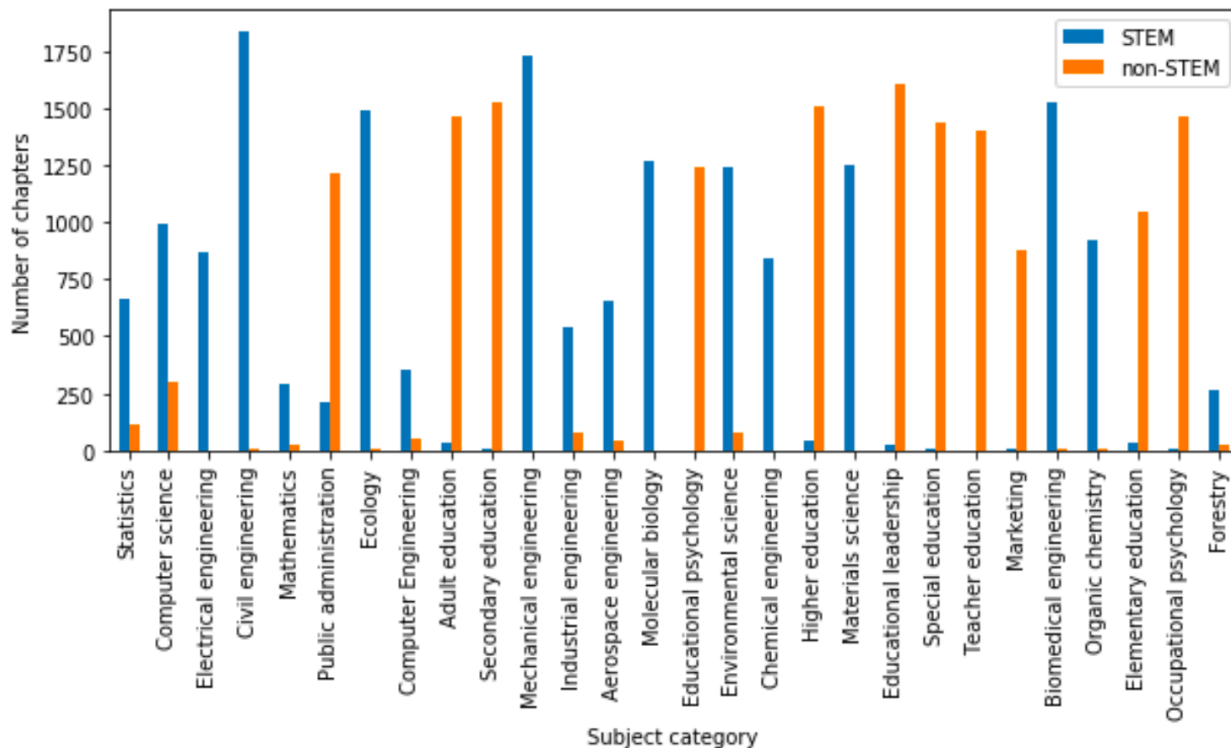


Figure 8.22: STEM and non-STEM categories predicted for chapters of ETDs by the LSTM model trained using Full-text (All data)

ground truth subject categories for each of the chapters of the ETDs considered. As can be seen from this figure, categories such as ‘Civil engineering’, and ‘Chemical engineering’ which are STEM categories contain chapters that also predominantly belong to STEM categories.

Chapter 9

Future Work

This research was performed on a fairly large corpus. However, we find that our deep learning models were not able to perform as well as we expected. Thus, we believe that using the methods developed as part of this research, we should train classifiers on a larger dataset. This would enable the deep learning models to learn better and would in turn improve the performance of the models.

Since we do not have expert annotations at the chapter-level, we believe that obtaining such annotations would enhance the performance of the models and would help us to better evaluate the multi-disciplinary labels generated by our automatic classification methods.

As described in Section 6.1.2, `gensim` provides us with top keywords extracted using the TextRank algorithm. We believe that the use of these keywords combined might give better performance with our sequential deep learning models.

Additionally, while the current dataset includes a mix of ETDs from different universities, it would be interesting to evaluate the classifiers built using this corpus on other datasets such as a dataset that is comprised of data from Historically Black Colleges and Universities (HBCUs) and Hispanic-Serving Institutions (HSIs).

Chapter 10

Conclusions

This research aims to generate chapter-level labels for chapters from Electronic Theses and Dissertations (ETDs). We trained over 2,000 different classifiers with varied embeddings and summarization techniques. Our study focuses on developing a methodology that is best suited to train classifiers at the full-text level and generate chapter-level labels through transfer learning. We find that our hypothesis that posits custom word and document embeddings trained on ETDs would outperform the pre-trained embeddings trained on the Wikipedia and Common Crawl datasets was confirmed. We also find that our hypothesis on the use of extractive summaries to improve the performance of our classifiers was partially confirmed by the use of the summaries generated using Luhn's Algorithm on our machine learning classifiers. This, however, was not the case with our deep learning classifiers. Additionally, the F1-scores achieved by our machine learning models were greater than our deep learning models which was contrary to our hypothesis. We believe that the performance of the deep learning classifiers could be improved by increasing the size of the dataset used to train our models. The small size of the dataset is a limitation of this study. We were able to successfully generate labels at the chapter-level and found that these labels do differ from those of the full-text. This indicates the merits of this research and its ability to help researchers identify the inter-disciplinary nature at the chapter-level of an ETD. However, the study was not compared against expert annotations at the chapter-level which would have provided a more robust analysis of the performance of our classifiers.

Bibliography

- [1] ABBYY. 2020. ABBYY Cloud OCR SDK. Retrieved May 25, 2020 from <https://www.ocrsdk.com/>
- [2] ACM. 2020. ACM Computing Classification System. Retrieved May 23, 2020 from <https://dl.acm.org/ccs>
- [3] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. DocBERT: BERT for document classification. *arXiv preprint arXiv:1904.08398* (2019).
- [4] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Rethinking complex neural network architectures for document classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4046–4051.
- [5] Isaac Alpizar-Chacon and Sergey Sosnovsky. 2019. Expanding the Web of Knowledge: one Textbook at a Time. In *Proceedings of the 30th on Hypertext and Social Media (HT '19)*. ACM, New York, NY, USA.
- [6] John Aromando, Bipasha Banerjee, William A Ingram, Palakh Jude, and Sampanna Kahu. 2020. Classification and extraction of information from ETD documents. (2020).
- [7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

- [8] Jason Brownlee. 2019. A Gentle Introduction to the Bag-of-Words Model. *Machine Learning Mastery* (Aug 2019). Retrieved May 25, 2020 from <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>
- [9] Jason Brownlee. 2020. Why One-Hot Encode Data in Machine Learning? *Machine Learning Mastery* (Apr 2020). Retrieved May 25, 2020 from <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>
- [10] Timo Böhm. 2018. The Three Main Branches of Word Embeddings. *Medium* (Dec 2018). Retrieved May 25, 2020 from <https://towardsdatascience.com/the-three-main-branches-of-word-embeddings-7b90fa36dfb9>
- [11] Chen and Yinlin. 2017. *A High-quality Digital Library Supporting Computing Education: The Ensemble Approach*. Ph.D. Dissertation. Virginia Tech, Blacksburg, VA, USA. Advisor(s) Fox, Edward A. <http://hdl.handle.net/10919/78750>
- [12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
- [13] David Cournapeau. 2019. *Scikit-learn: Machine Learning in Python*. Retrieved December 01, 2019 from <https://scikit-learn.org/stable/>
- [14] Google Developers. 2020. Machine Learning Glossary. Retrieved May 10, 2020 from <https://developers.google.com/machine-learning/glossary#1>
- [15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

- [16] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research* 22 (2004), 457–479.
- [17] Edward A Fox, Gail McMillan, and Venkat Srinivasan. 2012. Electronic theses and dissertations: progress, issues, and prospects. In *Putting Knowledge to Work & Letting Information Play*. Brill Sense, 95–110.
- [18] K. Fukunaga and P. M. Narendra. 1975. A Branch and Bound Algorithm for Computing k-Nearest Neighbors. *IEEE Trans. Comput.* C-24, 7 (July 1975), 750–753. <https://doi.org/10.1109/T-C.1975.224297>
- [19] Steve R Gunn. 1998. *Support vector machines for classification and regression*. Technical Report MP-TR-98-05. University of Southampton. Retrieved October 9, 2019 from <http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf>
- [20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- [22] Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* 12, 1 (1970), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>
- [23] Thorsten Joachims. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *Machine Learning: ECML-98*, Claire Nédellec

- and Céline Rouveirol (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 137–142. <https://doi.org/10.1007/BFb0026683>
- [24] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).
- [25] Ashraf M. Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. 2005. Multinomial Naive Bayes for Text Categorization Revisited. In *AI 2004: Advances in Artificial Intelligence*, Geoffrey I. Webb and Xinghuo Yu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 488–499. https://doi.org/10.1007/978-3-540-30549-1_43
- [26] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.
- [27] Steven Kelly Letkowski. 2009. *Beech bark disease: The relationship between scale and Neofractria lesion densities in an aftermath forest*. State University of New York College of Environmental Science and Forestry.
- [28] Andy Liaw and Matthew Wiener. 2002. Classification and Regression by randomForest. *R News* 2, 3 (2002), 18–22. Retrieved October 9, 2019 from https://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf
- [29] Ruikai Liu and Jorj McKie. 2020. PyMuPDF: Python bindings for the PDF rendering library MuPDF. Retrieved May 25, 2020 from <https://github.com/pymupdf/PyMuPDF>
- [30] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Research and Advanced Technology for Digital Libraries*, Maristella Agosti, José Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 473–474. https://doi.org/10.1007/978-3-642-04346-8_62

- [31] Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2, 2 (1958), 159–165.
- [32] Dean Malmgren. 2014. Textract 1.6.1 documentation. Retrieved May 25, 2020 from <https://textract.readthedocs.io/en/stable/>
- [33] Chris Mattmann. 2020. Tika-Python. Retrieved May 25, 2020 from <https://github.com/chris mattmann/tika-python> original-date: 2014-06-26T15:04:49Z.
- [34] Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, Vol. 752. 41–48. Retrieved October 9, 2019 from <https://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-007.pdf>
- [35] Chris McCormick. 2016. Word2Vec Tutorial - The Skip-Gram Model. *Chris McCormick Blog* (Apr 2016). Retrieved May 25, 2020 from <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>
- [36] Sevim McCutcheon. 2011. Basic, Fuller, Fullest: Treatment Options for Electronic Theses and Dissertations. *Library Collections, Acquisitions, and Technical Services* 35, 2 (Jan. 2011), 64–68. <https://doi.org/10.1016/j.lcats.2011.03.019>
- [37] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.
- [38] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
- [39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013). arXiv:1310.4546 <http://arxiv.org/abs/1310.4546>

- [40] Miso-Belica. 2020. Sumy. *GitHub* (Mar 2020). Retrieved April 22, 2020 from <https://github.com/miso-belica/sumy>
- [41] Michael Nguyen. 2019. Illustrated Guide to LSTM's and GRU's: A step by step explanation. Retrieved April 5, 2020 from <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- [42] Kartik Nooney. 2019. Deep dive into multi-label classification..! (With detailed Case Study). *Medium* (Feb. 2019). <https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff>
- [43] NSF. 2014. NSF Approved STEM Fields. Retrieved March 27, 2020 from <https://www.btaa.org/docs/default-source/diversity/nsf-approved-fields-of-study.pdf>
- [44] University of Cincinnati. 2020. STEM fields and CIP Codes. Retrieved March 27, 2020 from <https://www.uc.edu/content/dam/uc/international/docs/services/content/OPTSTEMfields.pdf>
- [45] The Library of Congress. 2019. *LC Linked Data Service: Authorities and Vocabularies (Library of Congress)*. Retrieved December 11, 2019 from <http://id.loc.gov/authorities/subjects.html>
- [46] Christopher Olah. 2015. Understanding LSTM Networks. Retrieved April 5, 2020 from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [47] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The pagerank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.
- [48] Jason Alan Palmer. 2020. Pdftotext. Retrieved May 25, 2020 from <https://pypi.org/project/pdftotext/>

- [49] Raghavendra Pappagari, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical Transformers for Long Document Classification. *arXiv preprint arXiv:1910.10781* (2019).
- [50] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. Retrieved December 11, 2019 from <http://www.aclweb.org/anthology/D14-1162>
- [51] Inc Phaseit and Mathieu Fenniak. 2016. PyPDF2 1.26.0 documentation. Retrieved May 25, 2020 from <https://pythonhosted.org/PyPDF2/>
- [52] NLTK Project. 2020. Natural Language Toolkit. Retrieved May 25, 2020 from <https://www.nltk.org/>
- [53] ProQuest. 2018. Subject Categories 2018–2019 Academic Year. <https://media2.proquest.com/documents/subject-categories-academic.pdf>. (Mar 2018).
- [54] RaRe-Technologies. 2020. Gensim. Retrieved April 22, 2020 from <https://github.com/RaRe-Technologies/gensim>
- [55] Sanjana Reddy. 2019. GloVe and fastText - Two Popular Word Vector Models in NLP. Retrieved May 25, 2020 from <https://cai.tools.sap/blog/glove-and-fasttext-two-popular-word-vector-models-in-nlp/>
- [56] W Ryan Richardson, Venkat Srinivasan, and Edward A Fox. 2008. Knowledge discovery in digital libraries of electronic theses and dissertations: an NDLTD case study. *International Journal on Digital Libraries* 9, 2 (2008), 163–171.
- [57] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal*

- representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.
- [58] Dipanjan (DJ) Sarkar. 2019. A hands-on intuitive approach to DL Methods for Text Data - Word2Vec, GloVe and FastText. *Medium* (March 2019).
<https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa>.
- [59] Yusuke Shinyama. 2019. PDFMiner: PDF parser and analyzer. Retrieved May 25, 2020 from <http://github.com/euske/pdfminer>
- [60] Gidi Shperber. 2019. A gentle introduction to Doc2Vec. Retrieved April 15, 2020 from <https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e>
- [61] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21. Retrieved October 9, 2019 from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.115.8343>
- [62] Venkat Srinivasan and Edward Fox. 2016. Progress towards automated ETD cataloging. In *19th International Symposium on Electronic Theses and Dissertations (ETD 2016): "Data and Dissertations"*.
- [63] Venkat Srinivasan and Edward A Fox. 2009. Topical categorization of large collections of electronic theses and dissertations. In *Proceedings of the 12th International Symposium on Electronic Theses and Dissertations*.
- [64] Nishan Subedi. 2018. FastText: Under the Hood. Retrieved April 15, 2020 from <https://towardsdatascience.com/fasttext-under-the-hood-11efc57b2b3>

- [65] RaRe Technologies. 2017. Text Summarization in Python: Extractive vs. Abstractive techniques revisited. <https://rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/>. *Pragmatic Machine Learning* (Apr 2017).
- [66] Madhav Thaker. 2019. Comparing Text Summarization Techniques. *Medium* (Apr 2019). Retrieved April 15, 2020 from <https://towardsdatascience.com/comparing-text-summarization-techniques-d1e2e465584e>
- [67] Theodoros Theodorou, Iosif Mporas, and Nikos Fakotakis. 2014. An overview of automatic audio segmentation. *International Journal of Information Technology and Computer Science (IJITCS)* 6, 11 (2014), 1.
- [68] Gregory L Thompson. 2011. *The Social Security retirement decision: Maximizing expected discounted worth*. Purdue University.
- [69] Balakrishnan Varadarajan et al. 2011. *Learning and inference algorithms for dynamical system models of dextrous motion*. Johns Hopkins University.
- [70] Lulu Wan, George Papageorgiou, Michael Seddon, and Mirko Bernardoni. 2019. Long-length Legal Document Classification. *arXiv preprint arXiv:1912.06905* (2019).
- [71] Bernard Widrow and Michael A Lehr. 1990. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proc. IEEE* 78, 9 (Sep. 1990), 1415–1442. <https://doi.org/10.1109/5.58323>
- [72] Wikipedia. 2020. Latent semantic analysis. Retrieved April 22, 2020 from https://en.wikipedia.org/wiki/Latent_semantic_analysis
- [73] Wikipedia. 2020. Multi-label classification. Retrieved May 10, 2020 from https://en.wikipedia.org/wiki/Multi-label_classification

- [74] J. Wu, B. Kandimalla, S. Rohatgi, A. Sefid, J. Mao, and C. L. Giles. 2018. CiteSeerX-2018: A Cleansed Multidisciplinary Scholarly Big Dataset. In *2018 IEEE International Conference on Big Data (Big Data)*. 5465–5467. <https://doi.org/10.1109/BigData.2018.8622114>
- [75] Jian Wu, Kyle Mark Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Suppawong Tuarob, Alexander G. Ororbia, Douglas Jordan, Prasenjit Mitra, and C. Lee Giles. 2015. CiteSeerX: AI in a Digital Library Search Engine. *AI Magazine* 36, 3 (Sep. 2015), 35–48. <https://doi.org/10.1609/aimag.v36i3.2601>

Appendices

Appendix A

Data Description

In this appendix, we provide auxiliary information for Chapter [4](#), ‘Data’.

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer Engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing engineering	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry		
Statistics	0	37	9	8	29	1	7	5	1	1	3	11	3	3	3	3	1	4	0	0	0	4	0	0	0	0	0	0	0	0
Computer science	37	0	66	6	15	0	6	175	0	3	15	6	13	5	2	4	3	1	3	0	0	0	4	0	0	1	0	0	0	0
Electrical engineering	9	66	0	3	4	0	1	98	0	0	24	3	27	0	0	4	9	1	36	0	0	0	0	71	1	0	0	0	0	0
Civil engineering	8	6	3	0	2	0	1	2	0	0	31	7	4	0	0	9	8	2	24	0	0	0	0	1	1	0	0	0	0	0
Mathematics	29	15	4	2	0	0	1	0	1	1	3	1	2	0	1	0	0	1	3	1	1	1	0	1	0	2	0	0	0	0
Public administration	1	0	0	0	0	0	0	0	4	1	0	0	0	0	0	4	0	9	0	10	0	0	1	0	0	0	0	13	1	0
Ecology	7	6	1	1	1	0	0	0	1	2	0	0	1	12	1	92	0	1	0	0	0	0	0	0	0	1	0	0	51	0
Computer Engineering	5	175	98	2	0	0	0	0	0	0	9	2	4	0	0	1	0	0	1	0	0	0	0	5	0	0	0	0	0	0
Adult education	1	0	0	0	1	4	1	0	0	8	0	0	0	0	22	0	0	0	0	61	8	35	3	0	0	6	5	0	0	0
Secondary education	1	3	0	0	1	1	2	0	0	0	0	0	0	0	48	0	0	97	0	0	0	0	0	0	0	35	35	0	0	0
Mechanical engineering	3	15	24	31	3	0	0	9	0	0	0	0	0	0	0	1	23	0	81	0	0	0	0	44	1	0	0	12	6	0
Industrial engineering	11	6	3	7	1	0	0	2	0	0	14	0	5	0	0	0	1	2	5	0	0	0	1	8	0	0	0	3	0	0
Aerospace engineering	3	13	27	4	2	0	1	4	0	0	124	5	0	0	0	0	6	0	21	0	0	0	1	1	1	0	3	0	0	0
Molecular Biology	3	5	0	0	0	0	12	0	0	0	0	0	0	0	0	4	9	0	1	0	0	0	0	16	3	0	0	0	0	0

Figure A.1: Part 1: Co-occurrence of the 28 subject categories

	Statistics	Computer science	Electrical engineering	Civil engineering	Mathematics	Public administration	Ecology	Computer engineering	Adult education	Secondary education	Mechanical engineering	Industrial engineering	Aerospace engineering	Molecular biology	Educational psychology	Environmental science	Chemical engineering	Higher education	Materials science	Educational leadership	Special education	Teacher education	Marketing	Biomedical engineering	Organic chemistry	Elementary education	Occupational psychology	Forestry
Educational psychology	3	2	0	0	1	0	1	0	22	48	0	0	0	0	0	0	0	136	0	95	50	34	1	0	0	66	16	0
Environmental science	3	4	4	9	0	4	92	1	0	0	1	0	0	4	0	0	0	2	2	0	1	1	0	1	3	0	0	10
Chemical engineering	1	3	9	8	0	0	0	0	0	0	23	1	6	9	0	0	0	0	53	0	0	0	0	18	6	0	0	0
Higher education	4	1	1	2	1	9	1	0	97	22	0	2	0	0	136	2	0	0	0	100	26	33	9	0	0	3	14	0
Materials science	0	3	36	24	1	0	0	1	0	0	81	5	21	1	0	2	53	0	0	0	0	0	0	22	8	0	0	1
Educational leadership	0	0	0	0	3	10	0	0	61	122	0	0	0	0	95	0	0	100	0	0	59	112	1	0	0	102	39	0
Special education	0	0	0	0	1	0	0	0	8	35	0	0	0	0	50	1	0	26	0	0	0	46	0	0	0	48	4	0
Teacher education	0	1	0	0	1	0	0	0	35	35	0	0	0	0	34	1	0	33	0	112	46	0	0	0	55	9	0	
Marketing	4	5	0	0	0	1	0	0	3	1	0	1	1	0	1	0	0	9	0	1	0	0	0	0	0	2	1	0
Biomedical engineering	0	12	71	1	1	0	0	5	0	0	44	8	1	16	0	1	18	0	22	0	0	0	0	1	0	0	0	0
Organic chemistry	0	0	1	1	0	0	0	0	0	0	1	0	1	3	0	3	6	0	8	0	0	0	0	1	0	0	0	0
Elementary education	1	2	0	0	2	0	1	0	6	12	0	0	0	0	66	0	0	3	0	102	48	55	0	0	0	0	1	0
Occupational psychology	0	3	0	0	0	13	0	0	5	6	0	3	3	0	16	0	0	14	0	39	4	9	2	0	0	0	0	0
Forestry	2	0	0	0	0	1	51	0	0	0	1	0	0	0	0	10	0	0	1	0	0	1	0	0	0	0	0	0

Figure A.2: Part 2: Co-occurrence of the 28 subject categories

Appendix B

Classifier Performance

In this appendix, we present additional results of our machine learning and deep learning classifiers.

B.1 Machine Learning Classifiers

Tables [B.1](#), [B.2](#), [B.3](#), [B.4](#), [B.5](#), [B.6](#) represent the scores obtained for the RF and SVM models with different embeddings on the Full-text (half data) set.

Table B.1: Performance of top 10 RF models with embeddings trained using the PSU + Illinois dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	38.431	56.676	41.047	46.629
25	3	200	0	200	100	38.174	56.271	41.316	46.575
50	3	200	1	100	25	37.874	56.523	40.985	46.57
50	3	200	1	100	50	38.517	55.628	41.242	46.439
25	3	100	1	100	25	38.303	55.479	40.979	46.381
100	3	100	1	200	25	38.903	56.311	40.687	46.323
50	3	200	0	100	50	38.003	55.747	40.813	46.309
50	5	200	1	100	100	37.446	56.204	40.787	46.309
25	3	200	0	100	100	38.903	55.264	40.925	46.235
100	5	100	1	100	50	37.96	55.717	40.765	46.187

Table B.2: Performance of top 10 SVM models with embeddings trained using the PSU + Illinois dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	200	0	200	25	53.256	75.098	59.172	65.508
50	3	200	0	100	25	52.569	75.436	58.511	65.143
50	3	200	0	200	25	52.699	75.064	58.714	65.131
100	3	200	0	200	25	52.998	74.548	58.815	65.053
25	5	200	0	200	25	52.656	74.812	58.718	65.049
100	5	200	0	200	25	52.742	74.593	58.849	65.047
25	3	200	0	200	25	52.312	74.888	58.435	64.856
25	3	200	0	100	25	52.27	74.917	58.144	64.763
50	5	200	0	100	25	52.356	74.687	58.141	64.652
50	5	100	0	200	25	51.842	74.599	58.109	64.586

Table B.3: Performance of top 10 RF models with embeddings trained using the VT dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	100	50	37.789	56.97	40.961	46.711
25	5	100	0	100	100	38.603	55.801	41.315	46.704
25	3	100	0	100	50	37.403	56.145	40.935	46.475
50	3	200	1	100	100	38.56	55.713	40.948	46.422
25	3	200	1	200	50	38.131	56.023	41.02	46.367
25	3	100	1	100	25	38.431	56.03	40.752	46.178
100	5	100	0	100	25	37.146	55.445	40.642	45.963
25	3	200	0	100	50	38.389	55.349	40.339	45.946
50	3	100	1	200	50	37.532	56.469	40.197	45.895
25	5	200	1	100	25	38.089	55.364	40.582	45.884

Table B.4: Performance of top 10 SVM models with embeddings trained using the VT dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	200	25	52.356	74.935	58.547	64.938
25	5	200	0	200	25	52.141	74.717	58.382	64.734
50	3	200	0	200	25	52.141	74.689	58.169	64.515
50	5	200	0	200	25	52.099	74.416	57.999	64.419
100	5	200	0	200	25	51.67	74.622	58.016	64.380
100	3	200	0	200	25	51.798	74.202	58.018	64.220
50	5	200	0	200	50	51.327	74.006	57.484	63.847
50	3	200	0	100	25	50.985	74.126	57.312	63.817
50	3	100	0	200	25	50.556	73.932	57.232	63.746
25	3	100	0	200	25	51.028	73.515	57.374	63.715

Table B.5: Performance of top 10 RF models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	100	25	39.202	56.446	41.46	47.004
100	3	100	1	100	100	39.545	56.543	41.394	46.831
100	3	100	1	100	50	38.174	56.875	41.066	46.623
50	3	100	1	100	100	38.56	56.235	40.996	46.559
25	5	100	1	100	100	38.389	56.088	41.061	46.531
25	3	100	1	100	100	39.160	56.316	41.214	46.526
50	5	200	1	100	25	38.346	56.18	41.003	46.469
50	3	100	0	100	50	38.217	56.233	40.794	46.346
25	5	200	1	100	25	38.389	56.032	40.969	46.333
100	3	100	0	100	100	37.917	56.155	40.741	46.306

Table B.6: Performance of top 10 SVM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	200	25	52.656	75.083	58.815	65.146
25	3	200	0	200	25	52.27	75.122	58.692	65.069
25	5	200	0	200	25	52.312	74.746	58.513	64.809
50	5	200	0	200	25	52.27	74.754	58.499	64.769
25	3	200	0	100	25	51.498	74.538	57.962	64.503
50	5	200	0	100	25	51.498	74.668	57.896	64.444
50	3	200	0	100	25	51.242	74.698	57.721	64.342
25	5	200	0	100	25	51.156	74.818	57.643	64.318
25	3	100	0	200	25	51.113	74.560	57.847	64.277
25	5	100	0	200	25	51.156	74.454	57.806	64.270

Tables B.7, B.8, B.9, B.10, B.11, B.12 represent the scores obtained for the RF and SVM models with different embeddings on the Full-text (All data).

Table B.7: Performance of top 10 RF models with embeddings trained using the PSU + Illinois dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	100	0	100	25	43.059	59.641	46.261	51.342
25	3	100	0	100	50	43.401	58.741	45.451	50.653
25	5	100	0	100	100	43.573	58.878	45.745	50.639
25	3	200	1	100	25	43.916	58.311	45.757	50.442
50	3	100	0	100	100	43.916	59.12	45.333	50.434
100	5	100	0	100	25	43.958	58.414	45.479	50.334
50	5	100	0	100	100	43.701	58.614	45.324	50.245
25	5	100	1	100	50	43.873	58.8	45.143	50.205
100	3	100	1	100	100	43.016	58.625	45.114	50.165
50	5	100	1	100	25	43.744	58.353	45.273	50.134

Table B.8: Performance of top 10 SVM models with embeddings trained using the PSU + Illinois dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	200	25	58.397	77.17	63.083	68.899
100	3	200	0	200	25	58.526	77.045	63.161	68.869
50	5	200	0	200	25	58.568	77.078	63.068	68.862
25	5	200	0	200	25	57.669	77.077	62.788	68.612
25	3	100	0	200	25	56.983	77.32	62.311	68.489
25	3	200	0	200	25	57.669	76.98	62.547	68.439
100	5	200	0	200	25	57.882	76.511	62.552	68.277
50	3	100	0	200	25	56.983	76.402	61.966	67.975
100	5	200	0	200	50	57.196	75.928	61.919	67.69
50	3	200	0	200	50	57.111	76.160	61.745	67.636

Table B.9: Performance of top 10 RF models with embeddings trained using the VT dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	3	200	0	100	100	42.93	59.514	45.663	50.766
100	5	100	1	100	100	43.830	58.723	45.516	50.570
50	3	100	1	100	100	42.716	59.446	45.224	50.444
25	3	200	1	100	25	43.486	58.098	45.463	50.346
25	3	100	0	100	25	43.401	57.998	45.182	50.075
25	3	100	0	100	100	43.958	58.013	45.109	50.047
25	5	100	1	100	50	43.144	58.811	44.822	49.975
100	3	200	0	100	25	42.502	59.074	44.567	49.873
100	3	200	1	200	50	43.101	57.811	44.952	49.793
25	3	200	1	100	100	43.187	57.533	44.832	49.716

Table B.10: Performance of top 10 SVM models with embeddings trained using the VT dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	25	57.926	76.983	62.612	68.530
50	3	200	0	200	25	57.669	77.098	62.443	68.43
100	5	200	0	200	25	57.926	76.259	62.0	68.118
25	3	200	0	200	25	57.669	76.742	62.197	68.091
50	5	200	0	200	25	57.711	76.451	62.292	68.088
100	3	200	0	200	25	57.196	75.738	61.848	67.549
50	5	100	0	200	25	56.682	76.178	61.509	67.473
100	5	200	0	100	25	55.998	76.214	61.288	67.339
25	5	100	0	200	25	56.383	76.153	61.28	67.277
50	5	200	0	100	25	56.340	76.032	61.259	67.273

Table B.11: Performance of top 10 RF models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	100	0	100	50	43.701	59.174	45.238	50.416
25	3	100	1	100	50	42.844	58.885	44.989	50.163
25	3	100	1	100	25	43.358	58.118	44.868	49.775
25	5	100	0	100	25	42.802	57.875	44.735	49.693
50	5	100	0	100	50	43.830	57.923	44.884	49.683
100	3	100	1	100	25	43.358	58.203	44.595	49.666
25	5	200	1	100	25	43.358	58.365	44.775	49.644
50	3	100	0	100	25	42.759	58.113	44.647	49.62
25	3	200	0	100	25	43.187	57.870	44.811	49.61
25	3	100	0	100	50	42.502	58.016	44.725	49.589

Table B.12: Performance of top 10 SVM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	200	25	57.455	76.885	62.358	68.325
50	5	200	0	200	25	57.196	76.434	62.108	67.932
50	3	200	0	200	25	56.769	76.127	61.638	67.481
50	3	200	0	100	25	56.255	75.802	61.206	67.261
25	5	200	0	200	25	56.855	75.62	61.453	67.222
25	5	200	0	100	25	55.998	75.654	61.004	67.123
100	5	100	0	200	25	55.869	75.818	60.989	67.006
50	5	200	0	200	50	56.426	75.192	61.21	66.998
25	5	100	0	200	25	55.698	75.619	60.885	66.923
50	3	200	0	200	50	56.255	75.131	61.065	66.862

Tables B.13, B.14, B.17, B.18, B.15, B.16 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings on the chapter subset data (full-text data).

Table B.13: Performance of top 10 RF models with embeddings trained using the PSU + Illinois dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	100	50	46.341	57.654	46.544	50.785
25	5	200	1	100	50	46.146	57.555	46.198	50.583
25	3	200	1	100	50	45.951	58.181	46.166	50.564
100	3	100	0	100	50	46.731	57.172	46.248	50.488
50	3	200	1	100	100	47.317	57.182	46.122	50.449
25	3	200	1	200	100	46.536	57.525	46.115	50.425
50	5	100	0	100	25	46.341	57.282	45.888	50.341
50	3	100	1	200	100	45.658	58.082	45.619	50.293
50	3	100	1	100	50	44.585	56.556	46.089	50.144
50	3	200	0	100	50	45.268	57.721	45.476	49.979

Table B.14: Performance of top 10 SVM models with embeddings trained using the PSU + Illinois dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	25	63.219	74.263	64.904	68.864
25	5	200	0	100	25	63.024	74.405	64.720	68.762
100	3	200	0	200	25	63.121	74.278	64.513	68.609
25	3	200	0	200	25	62.341	74.306	64.347	68.564
50	3	200	0	200	50	62.048	75.015	63.976	68.557
50	5	200	0	200	25	62.634	74.158	64.347	68.494
100	5	200	0	200	25	62.829	74.043	64.420	68.491
25	5	100	0	100	25	62.243	74.245	64.102	68.318
100	5	200	0	100	25	62.634	74.071	64.071	68.314
100	5	100	0	200	25	62.048	74.386	63.744	68.25

Table B.15: Performance of top 10 RF models with embeddings trained using the VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	5	100	0	100	100	46.536	58.013	46.705	50.809
25	5	100	1	100	50	45.658	57.67	46.248	50.482
25	5	200	1	100	25	45.073	57.413	45.655	50.109
50	5	100	0	100	100	45.365	56.083	46.026	49.952
25	3	100	1	200	50	46.341	57.425	45.245	49.907
100	3	200	0	200	25	45.073	57.178	45.446	49.844
100	5	200	0	200	25	45.853	56.95	45.379	49.573
25	3	100	0	200	100	45.268	56.954	44.939	49.538
100	5	100	0	100	25	44.682	56.757	45.07	49.451
25	3	100	1	100	25	44.975	57.111	44.881	49.442

Table B.16: Performance of top 10 SVM models with embeddings trained using the VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	25	62.536	74.705	64.115	68.51
25	3	200	0	200	25	62.731	74.256	64.023	68.34
100	3	200	0	200	25	62.439	73.978	64.168	68.301
100	5	200	0	200	25	62.731	73.787	64.12	68.198
50	5	200	0	200	25	62.536	73.947	63.907	68.152
25	5	200	0	200	50	62.341	73.76	63.971	68.022
50	3	200	0	200	25	62.048	73.8	63.759	68.014
25	3	200	0	200	50	62.146	74.087	63.798	68.009
100	5	200	0	100	25	62.536	73.966	63.566	67.946
100	3	200	0	200	50	62.048	73.76	63.745	67.869

Table B.17: Performance of top 10 RF models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	0	100	50	46.634	56.744	46.599	50.419
50	3	200	0	100	25	45.951	56.916	46.23	50.375
50	5	100	1	100	100	45.268	57.303	46.089	50.222
50	5	200	0	200	50	46.048	56.760	45.702	49.992
25	3	200	0	100	100	46.341	56.247	46.058	49.946
25	3	100	1	100	50	45.268	57.312	45.304	49.899
50	5	200	0	100	25	45.56	57.574	45.484	49.875
100	3	100	1	100	100	46.341	56.703	45.771	49.822
25	5	200	0	200	25	45.658	56.811	45.373	49.81
50	3	100	1	100	100	46.439	57.132	45.53	49.762

Table B.18: Performance of top 10 SVM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	200	50	63.512	75.999	65.042	69.512
50	5	200	0	200	25	63.317	75.717	64.521	69.08
50	5	200	0	200	50	63.121	75.095	64.554	68.875
25	5	200	0	200	50	63.024	74.952	64.524	68.806
25	5	200	0	200	25	63.219	75.416	64.261	68.797
25	3	200	0	200	50	63.121	75.063	64.469	68.764
25	3	200	0	200	25	63.512	75.093	64.285	68.707
25	5	100	0	200	50	62.536	74.765	64.2	68.61
50	3	200	0	200	25	62.829	75.104	64.163	68.604
25	3	100	0	200	50	62.048	74.995	63.958	68.518

Tables B.19, B.20, B.23, B.24, B.21, B.22 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `gensim`'s TextRank Generated Summary with ratio of 0.2.

Table B.19: Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using `gensim`'s TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	1	100	100	46.634	56.631	46.452	50.295
25	3	100	1	100	50	44.096	56.513	44.389	48.731
25	3	200	1	100	50	44.585	55.37	43.46	47.667
100	3	100	0	100	50	43.804	54.569	43.292	47.516
25	5	200	1	100	50	42.925	54.849	42.961	47.291

Table B.20: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using `gensim`'s TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	100	25	61.951	74.783	64.026	68.577
50	3	200	0	200	50	62.243	74.505	64.115	68.565
25	5	200	0	200	25	61.756	74.127	63.744	68.133
100	3	200	0	200	25	62.048	74.036	63.837	68.074
25	3	200	0	200	25	61.756	73.77	63.62	67.9

Table B.21: Performance of top 5 RF models with embeddings trained using the VT dataset using `gensim`'s TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	1	100	25	45.365	55.553	44.908	49.0
50	5	100	0	100	100	43.414	56.032	44.174	48.509
25	3	100	1	200	50	44.39	54.734	43.453	47.545
100	5	100	0	100	100	43.024	53.088	42.903	46.650
25	5	100	1	100	50	43.024	53.282	42.227	46.364

Table B.22: Performance of top 5 SVM models with embeddings trained using the VT dataset using `gensim`'s TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	5	200	0	200	25	63.219	75.029	64.69	68.952
100	3	200	0	200	25	62.536	74.388	63.907	68.194
25	5	200	0	200	25	61.951	73.904	63.395	67.757
50	5	200	0	200	25	61.951	73.789	63.341	67.662
25	3	200	0	200	25	61.756	73.667	63.148	67.506

Table B.23: Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using gensim's TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	100	25	44.975	56.431	45.327	49.431
25	3	100	0	100	50	44.292	56.088	44.016	48.52
50	5	200	0	200	50	42.829	55.354	43.775	48.329
50	5	100	1	100	100	40.0	54.101	44.052	47.937
25	3	200	0	100	100	43.902	53.937	42.962	47.003

Table B.24: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using gensim's TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	50	62.634	74.756	64.236	68.596
50	5	200	0	200	25	62.146	74.9	63.79	68.344
50	5	200	0	200	50	62.048	74.639	63.869	68.315
50	3	200	0	200	50	62.341	74.446	63.995	68.309
25	5	200	0	200	25	61.853	74.477	63.528	68.053

Tables B.25, B.26, B.29, B.30, B.27, B.28 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `gensim`'s TextRank Generated Summary with 100 words.

Table B.25: Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using `gensim`'s TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	3	100	0	100	50	45.756	57.342	45.612	49.669
25	3	100	1	100	50	45.268	55.69	44.271	48.646
50	3	200	1	100	100	44.78	55.123	43.888	48.108
25	3	200	1	100	50	43.707	53.909	42.473	46.664
25	5	200	1	100	50	42.634	53.166	42.376	46.438

Table B.26: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using `gensim`'s TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	3	200	0	200	25	63.512	73.622	64.335	68.319
25	5	200	0	200	25	62.243	72.746	63.471	67.477
25	3	200	0	200	25	62.341	72.765	63.370	67.396
25	5	200	0	100	25	60.39	72.883	62.214	66.732
50	3	200	0	200	50	61.17	71.447	62.661	66.463

Table B.27: Performance of top 5 RF models with embeddings trained using the VT dataset using `gensim`'s TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	0	100	100	44.096	55.823	44.416	48.486
25	5	200	1	100	25	43.804	53.890	43.297	47.372
100	5	100	0	100	100	43.024	55.041	43.068	47.333
25	3	100	1	200	50	42.925	54.195	43.194	47.226
25	5	100	1	100	50	43.219	54.052	43.107	47.019

Table B.28: Performance of top 5 SVM models with embeddings trained using the VT dataset using `gensim`'s TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	200	0	200	25	63.414	74.529	64.771	68.964
100	3	200	0	200	25	63.609	74.003	64.869	68.827
25	5	200	0	200	25	63.317	74.41	64.579	68.796
25	3	200	0	200	25	64.195	74.167	64.574	68.67
100	5	200	0	200	25	63.707	73.823	64.608	68.544

Table B.29: Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using gensim's TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	100	100	43.414	54.973	43.971	48.101
50	5	200	0	200	50	44.096	54.308	43.830	47.965
25	3	100	0	100	50	43.512	54.825	43.349	47.624
50	3	200	0	100	25	40.0	52.809	43.025	46.546
50	5	100	1	100	100	41.853	52.271	41.893	45.751

Table B.30: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using gensim's TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	200	0	200	25	62.048	73.246	63.402	67.565
25	5	200	0	200	25	61.756	72.933	63.099	67.286
50	5	200	0	200	50	61.56	72.122	62.876	66.794
50	3	200	0	200	50	61.463	72.04	62.649	66.595
25	5	200	0	200	50	61.463	71.621	62.586	66.405

Tables B.31, B.32, B.35, B.36, B.33, B.34 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using sumy's LexRank Generated Summary.

Table B.31: Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	1	100	50	45.463	56.411	45.538	49.347
25	3	100	1	100	50	44.096	54.43	43.159	47.398
100	3	100	0	100	50	43.804	54.933	42.95	47.311
25	5	200	1	100	50	43.219	54.149	42.644	46.922
50	3	200	1	100	100	42.439	52.977	42.798	46.682

Table B.32: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	200	25	61.951	73.714	63.512	67.785
25	5	200	0	200	25	61.56	73.351	63.395	67.646
100	3	200	0	200	25	62.243	73.345	63.293	67.534
25	5	200	0	100	25	59.707	71.526	61.872	65.973
50	3	200	0	200	50	59.609	70.875	61.702	65.593

Table B.33: Performance of top 5 RF models with embeddings trained using the VT dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	0	100	100	43.608	56.169	43.618	48.303
25	5	200	1	100	25	43.219	54.535	43.368	47.609
25	5	100	1	100	50	42.634	54.336	42.974	47.296
25	3	100	1	200	50	42.341	53.725	42.403	46.741
100	5	100	0	100	100	43.120	52.653	42.225	46.184

Table B.34: Performance of top 5 SVM models with embeddings trained using the VT dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	5	200	0	200	25	62.634	74.283	63.876	68.119
100	3	200	0	200	25	62.439	73.897	63.729	68.002
50	5	200	0	200	25	62.048	73.772	63.527	67.743
25	5	200	0	200	25	62.048	73.407	63.55	67.669
25	3	200	0	200	25	61.756	73.714	63.193	67.616

Table B.35: Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	0	100	50	45.658	56.422	44.416	48.675
50	5	100	1	100	100	44.292	54.642	44.035	47.876
50	3	200	0	100	25	44.487	52.928	43.943	47.504
50	5	200	0	200	50	42.439	52.837	41.797	45.882
25	3	200	0	100	100	42.341	52.427	41.337	45.446

Table B.36: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	25	62.536	73.889	63.597	67.729
50	3	200	0	200	50	62.048	73.443	63.324	67.459
50	5	200	0	200	25	61.756	73.708	62.905	67.207
25	5	200	0	200	50	60.878	72.606	62.200	66.486
50	5	200	0	200	50	60.975	72.359	62.148	66.346

Tables B.37, B.38, B.41, B.42, B.39, B.40 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using Luhn's Algorithm.

Table B.37: Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	100	50	44.878	55.162	44.638	48.482
50	3	200	1	100	100	44.682	53.601	44.601	48.071
25	3	200	1	100	50	45.073	54.693	43.846	48.028
100	3	100	0	100	50	44.195	55.400	43.765	47.994
25	5	200	1	100	50	43.317	53.077	43.046	46.836

Table B.38: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	3	200	0	200	25	62.829	73.005	60.0	67.845
25	5	200	0	200	25	62.439	72.133	63.368	67.133
50	3	200	0	200	50	60.975	72.397	62.778	66.804
25	3	200	0	200	25	61.56	71.6	62.683	66.539
25	5	200	0	100	25	60.585	71.769	61.925	66.035

Table B.39: Performance of top 5 RF models with embeddings trained using the VT dataset using sumy's Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	100	1	100	50	45.17	57.328	45.298	49.984
25	5	200	1	100	25	46.048	57.107	45.768	49.868
25	3	100	1	200	50	44.195	55.403	44.307	48.673
100	5	100	0	100	100	42.829	51.928	42.079	45.940
50	5	100	0	100	100	42.536	53.458	41.404	45.795

Table B.40: Performance of top 5 SVM models with embeddings trained using the VT dataset using sumy's Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	200	25	64.39	75.952	65.694	70.052
100	5	200	0	200	25	64.78	75.215	65.595	69.64
25	5	200	0	200	25	63.609	75.182	65.401	69.553
50	5	200	0	200	25	64.195	75.163	65.198	69.433
100	3	200	0	200	25	63.902	75.075	65.086	69.291

Table B.41: Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	100	25	44.975	55.596	44.991	49.145
50	5	100	1	100	100	44.096	54.7	43.407	47.598
25	3	200	0	100	100	43.317	54.067	42.568	46.967
50	5	200	0	200	50	40.0	53.339	42.845	46.963
25	3	100	0	100	50	41.853	52.949	42.16	46.04

Table B.42: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	200	0	200	25	63.024	74.329	64.432	68.518
25	5	200	0	200	25	62.731	74.017	64.199	68.328
25	5	200	0	200	50	63.024	73.306	64.031	67.876
50	3	200	0	200	50	62.439	73.489	63.944	67.864
50	5	200	0	200	50	62.146	73.107	63.636	67.545

Tables B.43, B.44, B.47, B.48, B.45, B.46 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using LSA.

Table B.43: Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	3	100	0	100	50	44.487	55.249	44.333	48.294
25	3	200	1	100	50	44.682	54.775	44.379	48.217
25	3	100	1	100	50	42.634	54.705	43.11	47.457
50	3	200	1	100	100	42.439	54.789	43.032	47.205
25	5	200	1	100	50	42.341	53.827	42.436	46.465

Table B.44: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	25	62.341	73.964	64.563	68.561
100	3	200	0	200	25	62.926	73.178	64.347	68.094
25	3	200	0	200	25	61.853	73.367	63.903	67.847
50	3	200	0	200	50	60.195	72.170	62.532	66.605
25	5	200	0	100	25	60.487	72.396	61.895	66.269

Table B.45: Performance of top 5 RF models with embeddings trained using the VT dataset using sumy's Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	1	100	25	43.317	54.983	44.201	48.405
25	5	100	1	100	50	44.195	54.458	43.438	47.532
100	5	100	0	100	100	42.731	52.658	42.553	46.583
25	3	100	1	200	50	43.219	53.017	42.455	46.461
50	5	100	0	100	100	42.341	53.413	41.922	46.245

Table B.46: Performance of top 5 SVM models with embeddings trained using the VT dataset using sumy's Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	200	25	63.902	75.735	65.751	70.008
50	5	200	0	200	25	64.195	75.175	65.83	69.817
100	3	200	0	200	25	60.0	75.164	65.378	69.482
100	5	200	0	200	25	63.804	74.936	65.292	69.338
25	5	200	0	200	25	62.341	74.705	64.474	68.818

Table B.47: Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	100	25	40.0	55.850	44.327	48.684
25	3	200	0	100	100	44.682	55.193	44.315	48.409
50	5	200	0	200	50	44.292	54.581	43.903	47.910
25	3	100	0	100	50	45.17	53.044	43.858	47.413
50	5	100	1	100	100	42.341	53.961	42.693	46.883

Table B.48: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	200	50	60.0	74.732	65.051	69.065
25	5	200	0	200	50	63.707	74.499	64.995	68.974
50	5	200	0	200	25	63.804	74.526	64.969	68.94
50	5	200	0	200	50	63.121	74.198	64.675	68.62
25	5	200	0	200	25	62.634	74.047	64.095	68.234

Tables B.49, B.50, B.53, B.54, B.51, B.52 represent the scores obtained on the chapter subset dataset for the RF and SVM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using LSA with stopwords.

Table B.49: Performance of top 5 RF models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	1	100	50	45.17	55.254	44.471	48.618
50	3	200	1	100	100	44.292	55.277	44.59	48.545
25	3	100	1	100	50	43.317	54.562	43.113	47.479
25	5	200	1	100	50	43.024	53.276	42.372	46.553
100	3	100	0	100	50	41.658	51.49	40.786	44.71

Table B.50: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	0	200	25	61.853	73.036	63.379	67.524
25	5	200	0	200	25	61.951	72.277	63.321	67.179
25	5	200	0	100	25	60.682	72.586	62.308	66.663
50	3	200	0	200	50	60.39	72.03	61.953	66.215
100	3	200	0	200	25	60.682	71.31	62.024	65.932

Table B.51: Performance of top 5 RF models with embeddings trained using the VT dataset using sumy's Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	100	1	100	50	40.0	55.440	44.352	48.372
50	5	100	0	100	100	44.195	54.932	44.032	48.001
25	5	200	1	100	25	44.096	53.871	43.524	47.426
100	5	100	0	100	100	41.951	52.763	42.044	45.971
25	3	100	1	200	50	43.414	51.913	41.427	45.189

Table B.52: Performance of top 5 SVM models with embeddings trained using the VT dataset using sumy's Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	3	200	0	200	25	62.926	73.402	63.934	67.978
100	5	200	0	200	25	62.731	73.334	63.91	67.894
50	5	200	0	200	25	61.951	73.267	63.341	67.551
25	3	200	0	200	25	62.243	73.093	63.464	67.545
25	5	200	0	200	25	61.56	72.566	62.982	67.054

Table B.53: Performance of top 5 RF models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	0	100	50	44.878	55.989	44.903	48.962
50	3	200	0	100	25	44.78	54.166	43.801	47.641
50	5	100	1	100	100	43.608	54.315	43.234	47.345
50	5	200	0	200	50	43.120	53.418	42.751	46.872
25	3	200	0	100	100	43.414	53.675	42.620	46.64

Table B.54: Performance of top 5 SVM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	0	200	50	61.56	73.409	63.464	67.642
50	5	200	0	200	25	61.56	72.811	63.23	67.332
25	5	200	0	200	25	61.463	72.89	63.076	67.318
25	5	200	0	200	50	60.975	73.070	62.83	67.140
50	5	200	0	200	50	61.17	73.232	62.82	67.133

B.2 Deep Learning Classifiers

Tables B.55, B.56, and B.57 represent the scores obtained on the Full-text (Half data) for the LSTM models with different embeddings. Each of these models was trained with a statesize of 1024, a dropout of 0.2, batch size of 512, and 15 epochs.

Table B.55: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	30.548	80.399	33.068	41.609
50	3	200	1	100	50	27.977	78.742	29.815	38.601
50	3	200	1	100	25	27.892	86.816	28.811	36.563
25	3	100	1	100	25	27.162	83.088	27.592	35.276
50	3	200	0	100	25	15.895	93.93	15.895	26.244
25	3	200	0	200	100	17.009	84.714	17.009	25.028
50	3	200	0	200	25	13.753	79.375	14.261	22.837
100	3	200	0	200	25	10.711	88.34	12.489	18.616
25	5	200	0	200	25	8.483	99.089	8.483	15.619
50	5	200	0	200	25	8.654	97.78	8.654	15.511

Table B.56: Performance of top 10 LSTM models with embeddings trained using the VT dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	1	200	50	38.431	77.117	40.296	48.759
50	3	200	1	100	100	36.332	80.334	38.126	46.495
25	3	100	1	100	50	32.903	73.382	36.336	45.822
100	5	200	0	200	25	14.91	79.304	14.946	23.73
50	3	200	0	200	25	15.338	83.845	15.338	22.958
50	5	200	0	200	25	13.496	87.758	13.496	21.373
25	5	200	0	200	25	12.982	90.002	13.174	21.198
25	5	100	0	100	100	11.568	87.31	11.568	19.052
25	3	200	0	200	25	10.453	79.55	11.916	16.603
25	3	100	0	100	50	8.74	70.704	8.74	14.89

Table B.57: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (Half data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	100	1	100	100	34.79	67.698	37.03	44.111
50	3	100	1	100	100	29.604	77.983	32.537	40.764
100	3	100	1	100	100	30.334	72.959	32.251	39.201
100	3	100	1	100	50	25.491	73.894	28.328	36.964
25	3	200	0	200	25	15.038	73.806	16.281	24.659
25	3	200	0	100	25	13.11	85.078	14.124	22.404
50	3	200	0	200	25	13.024	90.58	13.024	21.359
50	5	200	0	200	25	12.982	79.651	12.975	20.241
25	5	200	0	200	25	11.525	89.13	11.525	18.573
25	3	200	0	100	25	6.598	86.105	6.598	12.125

Tables B.58, B.59, and B.60 represent the scores obtained on the Full-text (All data) for the LSTM models with different embeddings. Each of these models was trained with a statesize of 1024, a dropout of 0.2, batch size of 512, and 15 epochs.

Table B.58: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	1	100	25	52.056	76.842	54.196	60.699
25	3	100	0	200	25	41.516	78.09	43.043	51.405
25	5	200	0	200	25	40.959	78.005	42.03	49.809
50	3	200	0	200	25	39.717	75.088	41.543	49.306
50	5	200	0	200	25	38.474	82.326	39.693	47.978
50	3	100	0	100	100	37.36	70.606	38.174	45.661
100	3	200	0	200	25	32.733	72.461	32.867	42.015
25	3	100	0	100	50	31.747	77.128	32.269	41.409
50	3	100	0	100	25	28.534	79.58	29.921	39.050
25	5	100	0	100	100	28.191	72.690	28.406	35.175

Table B.59: Performance of top 10 LSTM models with embeddings trained using the VT dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
100	5	100	1	100	100	50.257	72.348	52.758	58.583
25	3	200	1	100	25	49.614	72.539	51.166	57.450
50	3	100	1	100	100	48.886	73.509	50.117	56.772
25	5	200	0	200	25	43.701	77.389	44.743	52.26
25	3	200	0	200	25	41.302	79.015	42.809	50.751
50	5	200	0	200	25	40.488	77.827	41.829	49.927
100	5	200	0	200	25	39.160	74.422	40.636	48.906
100	3	200	0	100	100	36.418	82.599	37.129	45.42
25	3	100	0	100	25	36.975	74.972	37.324	44.734
50	3	200	0	200	25	35.132	75.626	36.556	42.532

Table B.60: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Full-text (All data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	100	50	48.071	74.37	50.516	56.664
25	3	100	1	100	25	47.043	76.09	48.805	55.940
50	5	200	0	200	25	38.217	80.191	39.900	47.967
50	3	100	0	100	50	37.574	81.529	38.711	47.471
25	3	200	0	200	25	37.146	74.827	38.477	45.614
50	5	100	0	100	50	37.746	74.173	37.879	45.062
25	5	200	0	200	25	37.103	77.065	37.973	44.929
50	3	200	0	200	25	36.803	76.292	37.137	44.291
50	3	200	0	100	25	29.220	87.602	29.220	39.827
25	5	100	0	100	25	24.507	77.269	24.507	33.072

Tables B.61, B.62, and B.63 represent the scores obtained on the chapter subset data (full-text data) for the LSTM models with different embeddings.

Table B.61: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	1	100	100	60.0	74.14	60.0	64.407
25	5	200	1	100	50	57.56	77.753	57.56	63.556
25	3	200	1	100	50	57.17	76.049	57.17	62.541
25	3	100	1	100	50	54.340	75.457	54.340	59.646
25	5	200	0	100	25	38.829	82.308	38.829	46.907
100	3	200	0	200	25	34.731	83.557	34.731	43.203
25	3	200	0	200	25	35.219	89.057	35.219	43.081
100	3	100	0	100	50	28.975	86.006	28.975	37.515
25	5	200	0	200	25	28.877	93.544	28.877	37.367

Table B.62: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	61.073	73.088	61.111	64.821
25	5	200	1	100	25	58.243	74.559	58.243	62.995
25	5	100	1	100	50	55.120	74.046	55.544	61.159
25	5	200	0	200	25	50.439	78.903	50.439	56.964
25	3	200	0	200	25	50.439	79.178	50.439	56.632
100	3	200	0	200	25	47.121	78.349	47.121	54.384
100	5	200	0	200	25	46.243	75.691	46.243	52.477
50	5	200	0	200	25	42.731	78.393	42.731	51.053
100	5	100	0	100	100	38.634	80.886	38.634	48.344
50	5	100	0	100	100	34.634	81.119	34.634	42.63

Table B.63: Performance of top 10 LSTM models with embeddings trained using the PSU + Illinois + VT dataset on the Chapter subset data (full-text data)

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	56.389	71.868	56.852	61.607
50	3	200	0	200	50	48.487	77.884	48.487	55.091
50	5	200	0	200	25	45.853	85.835	45.853	54.738
25	5	200	0	200	25	46.146	83.202	46.146	54.386
25	3	200	0	100	100	47.024	75.511	47.024	53.542
50	3	200	0	100	25	43.317	79.581	43.317	51.766
50	5	200	0	200	50	47.317	74.19	47.317	51.336
25	3	100	0	100	50	42.634	80.342	42.634	50.926
25	5	200	0	200	50	43.024	83.384	43.024	50.881
50	5	200	0	200	50	40.097	80.654	40.097	47.504

Tables B.64, B.65, and B.66 represent the scores obtained on the chapter subset dataset for the LSTM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `gensim`'s TextRank Generated Summary with ratio of 0.2.

Table B.64: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using `gensim`'s TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	1	100	50	58.926	72.63	59.006	63.273
50	3	200	1	100	100	56.096	74.694	56.183	61.149
25	3	100	1	100	50	52.486	73.303	52.626	58.628
25	3	200	1	100	50	52.681	74.988	52.681	58.291
25	5	200	0	200	25	43.902	85.814	43.902	52.995

Table B.65: Performance of top 5 LSTM models with embeddings trained using the VT dataset using `gensim`'s TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	61.463	72.043	61.612	64.755
25	5	200	1	100	25	59.317	71.197	59.592	62.975
25	5	100	1	100	50	58.243	72.179	58.373	62.461
50	5	200	0	200	25	52.681	77.281	52.681	58.424
100	5	200	0	200	25	49.852	79.403	49.852	56.518

Table B.66: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using gensim's TextRank Generated Summary with ratio of 0.2

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	57.755	70.126	57.755	61.382
50	5	200	0	200	50	50.634	79.339	50.634	57.943
25	5	200	0	200	50	52.195	74.713	52.195	57.748
50	5	200	0	200	50	48.878	82.119	48.878	56.248
25	5	200	0	200	25	47.317	83.585	47.317	56.211

Tables B.67, B.68, and B.69 represent the scores obtained on the chapter subset dataset for the LSTM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `gensim`'s TextRank Generated Summary with 100 words.

Table B.67: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using `gensim`'s TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	200	1	100	50	57.072	74.060	57.072	61.815
50	3	200	1	100	100	51.512	75.067	51.512	57.865
25	5	200	1	100	50	49.073	72.334	49.073	54.473
25	3	100	1	100	50	47.219	72.649	47.219	53.703
100	3	200	0	200	25	43.317	76.267	43.317	50.67

Table B.68: Performance of top 5 LSTM models with embeddings trained using the VT dataset using `gensim`'s TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	57.365	70.769	57.365	61.312
25	5	200	1	100	25	51.609	75.468	51.609	58.109
25	5	100	1	100	50	51.317	75.985	51.317	57.708
50	5	200	0	200	25	47.024	79.562	47.024	54.168
25	3	200	0	200	25	45.073	81.57	45.073	52.534

Table B.69: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using gensim's TextRank Generated Summary with 100 words

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	200	0	200	50	51.317	77.275	51.317	57.942
50	5	100	1	100	100	51.121	71.857	51.121	56.426
50	5	200	0	200	25	50.634	73.887	50.634	56.049
25	5	200	0	200	50	46.341	75.823	46.341	53.181
25	5	200	0	200	25	45.073	78.68	45.073	52.795

Tables B.70, B.71, and B.72 represent the scores obtained on the chapter subset dataset for the LSTM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s LexRank Generated Summary.

Table B.70: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	1	100	100	53.657	73.756	53.429	58.278
25	3	200	1	100	50	51.219	73.66	51.219	57.523
25	5	200	1	100	50	47.804	78.959	47.804	54.301
25	3	100	1	100	50	44.195	81.276	44.195	51.516
100	3	200	0	200	25	42.341	82.725	42.341	51.141

Table B.71: Performance of top 5 LSTM models with embeddings trained using the VT dataset using `sumy`'s LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	54.047	72.848	54.227	58.994
25	5	200	1	100	25	52.975	70.586	52.975	57.469
25	5	100	1	100	50	48.195	76.500	48.195	55.157
100	3	200	0	200	25	47.804	73.993	47.804	54.716
25	3	200	0	200	25	46.536	82.256	46.536	54.273

Table B.72: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's LexRank Generated Summary

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	55.706	68.354	55.706	59.343
50	5	200	0	200	50	47.414	80.098	47.414	54.922
50	5	200	0	200	25	44.78	84.37	44.78	53.569
25	5	200	0	200	25	44.39	80.707	44.39	53.059
25	5	200	0	200	50	46.536	79.63	46.536	52.886

Tables B.73, B.74, and B.75 represent the scores obtained on the chapter subset dataset for the LSTM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using Luhn's Algorithm.

Table B.73: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	1	100	100	53.657	76.957	53.657	59.465
25	5	200	1	100	50	53.364	74.914	53.364	58.218
25	3	100	1	100	50	51.121	75.787	51.121	58.001
25	3	200	1	100	50	48.975	82.529	48.975	57.255
100	3	200	0	200	25	40.0	80.22	40.0	52.217

Table B.74: Performance of top 5 LSTM models with embeddings trained using the VT dataset using `sumy`'s Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	58.731	74.297	58.754	63.79
25	5	100	1	100	50	52.195	76.416	52.190	58.469
100	3	200	0	200	25	47.024	83.152	47.024	56.167
25	5	200	1	100	25	48.681	77.357	48.681	54.801
25	3	200	0	200	25	45.463	77.759	45.463	52.955

Table B.75: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using Luhn's Algorithm

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	55.803	72.104	55.793	60.057
50	5	200	0	200	25	49.463	79.995	49.463	56.882
50	3	200	0	200	50	48.78	75.500	48.78	54.996
25	5	200	0	200	25	46.536	82.201	46.536	54.183
25	3	200	0	100	100	44.096	82.551	44.096	53.400

Tables B.76, B.77, and B.78 represent the scores obtained on the chapter subset dataset for the LSTM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using LSA.

Table B.76: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	1	100	100	55.803	74.469	55.847	61.023
25	3	200	1	100	50	53.657	76.089	53.703	59.427
25	5	200	1	100	50	51.803	71.637	51.803	57.46
25	3	100	1	100	50	49.658	75.357	49.658	55.583
25	3	200	0	200	25	43.902	76.492	43.902	50.554

Table B.77: Performance of top 5 LSTM models with embeddings trained using the VT dataset using `sumy`'s Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	56.291	73.272	56.335	61.146
25	5	100	1	100	50	53.657	72.570	53.605	58.913
25	5	200	1	100	25	52.486	74.776	52.486	58.635
25	3	200	0	200	25	48.975	81.101	48.975	55.452
100	5	200	0	200	25	46.926	82.758	46.926	54.413

Table B.78: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	5	200	0	200	25	51.803	74.175	51.803	57.838
50	3	200	0	200	50	50.341	80.231	50.341	56.842
50	5	100	1	100	100	50.731	72.414	50.731	56.447
50	5	200	0	200	50	48.195	78.393	48.195	55.300
50	5	200	0	200	50	46.146	80.059	46.146	53.352

Tables B.79, B.80, and B.81 represent the scores obtained on the chapter subset dataset for the LSTM models with different embeddings. For this set of experiments, we use the combined summaries of the chapters of the ETDs present in this subset using `sumy`'s Generated Summary using LSA with stopwords.

Table B.79: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois dataset using `sumy`'s Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	3	200	1	100	100	56.486	73.496	56.572	61.109
25	3	200	1	100	50	52.585	75.182	52.585	58.58
25	5	200	1	100	50	51.219	77.393	51.456	57.617
25	3	100	1	100	50	50.731	74.79	50.778	56.782
100	3	200	0	200	25	38.439	83.341	38.439	45.53

Table B.80: Performance of top 5 LSTM models with embeddings trained using the VT dataset using `sumy`'s Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
25	3	100	1	200	50	57.56	71.942	57.889	61.82
25	5	200	1	100	25	53.073	74.833	53.073	59.473
50	5	200	0	200	25	51.512	79.837	51.512	59.345
25	5	100	1	100	50	54.146	72.448	54.146	59.084
100	5	200	0	200	25	48.487	79.843	48.487	56.011

Table B.81: Performance of top 5 LSTM models with embeddings trained using the PSU + Illinois + VT dataset using sumy's Generated Summary using LSA with stopwords

fastText			Doc2Vec			Accuracy	Precision	Recall	F1-score
Epochs	Word N Grams	Dim	DM	Dim	Epochs				
50	5	100	1	100	100	55.218	72.502	55.218	60.488
50	5	200	0	200	25	49.17	77.669	49.17	56.610
50	5	200	0	200	50	47.804	77.445	47.804	55.396
25	5	200	0	200	50	48.292	79.42	48.292	55.198
25	3	200	0	100	100	47.512	77.503	47.512	55.101