CSCI 6350 Special Topics in AI, NLP Project: Topic Modeling for a Digital Book Corpus

Matthew I. Swindall

Abstract

In various experiments, we explore the application of various topic modeling techniques and named entity recognition to a corpus of 925 educational books. The goal is to extract important terms, topics, and named entities from the book corpus using methods such as n-gram models, Latent Dirichlet Allocation (LDA), a spaCy-based language model, and the transformer-based BERTopic model. The text data are pre-processed to remove formatting artifacts such as page numbers and headers. The extracted topics and named entities from each model were then combined and ranked by their confidence/frequency values to produce a robust analysis of the content within each book. Additionally, a small-scale experiment was performed using a hierarchical text summarization approach to provide concise summaries of the book contents. The results show that the topic modeling and named entity extraction techniques were generally effective, but challenges remain in fully removing formatting artifacts and optimizing the combination of model outputs. The hierarchical summarization approach also produced promising results, warranting further exploration.

1 Introduction

Topic modelling is a well know Natural Language Processing (NLP) technique for text analysis that seeks to identify important words and phrases in a document or corpus [6]. The end goal is to gain an understanding of the various texts contained in a corpus and explore similarities and dissimilarities between the component texts. A similar NLP methodology is Named Entity Recognition (NER) which identifies proper nouns within a text, i.e. people, geopolitical entities such as cities or nations, organizations, and works of art such as book or film titles. Many tools exist for Topic Modeling and NER that are state-of-the-art [7], but no system is perfect. It is typical to see multiple methods used on the same corpus and those results compared and combined to produce more reliable modeling of the topics contained within the documents.

In this article we discuss employing multiple topic modelling techniques and an NER model to a book corpus. Methodologies including n-gram models; Latent Dirichlet Allocation, a Convolutional Neural Network (CNN) based language model called en_core_web_sm which is available in the spaCy Python library (`https://spacy.io/`), and the transformer based models available in BERTopic. The goal will be to extract important terms and topics from the corpus using each technique, then compare and com-

bine these results to produce a robust analysis of the topics contained within each book in the corpus. All code, data files, and the results data are available online at `https://github.com/mis2n/nlp_code`

2 Background & Related Work

2.1 Data Source

The data for this project consists of text extracted from 925 PDF files containing various books related to education. These books have aged out of copyright, and were set to be destroyed to make room for new books in the Walker Library at Middle Tennessee State University The books were optically scanned for preservation and saved as PDF files. To add text to the images contained in each PDF, the Optical Character Recognition (OCR) engine Tesseract (`https://github.com/tesseract-ocr/tesseract`) was used to identify text in each book file. The text produced by Tesseract was then added to the PDF files as a text overlay, and saved as separate text files for NLP processing. Fifteen of these books produced unfavorably during the OCR process due to their atypical contents, such as picture books or books containing sheet music. Tesseract produced little-to-no useful text from these books, leaving a total of 925 books for these experiments.

2.2 Literature Review

A fair amount of literature exists relating to topic modeling. Much focus is given to the general length of documents [1] [2]. Some approaches are better suited to shorter texts, while other work well for longer texts. For a corpus of books, it may be reasonable to explore both paths. Typical approaches found in literature include n-gram models [3], Latent Dirichlet Allocation (LDA) [4], non-negative matrix factorization (NMF) [5] [1], as well as newer modeling techniques such as Top2Vec [5] and BERTopic [5] which is a transformer based model.

3 Research Method

3.1 Pre-Processing

As with any text corpus, the data must be cleaned. Topic modeling can be challenging when artifacts such as contents pages, indexes, section headers, and page numbers exist. Such artifacts need to be removed from each document, otherwise topic modeling can misunderstand frequent word instances as being of significance when they are not. For example, in a book where page numbers are written as, "Page 3", or "page 7", the frequent appearance of the word, "page" may lead the model to assume it is a word of importance. In contrast, frequent references to specific pages within the text, may reveal portions of the text of great importance. To remove unwanted artifacts, Regular Expressions are employed utilizing the Python Re library. An additional challenge in this case comes from the OCR model used to extract text from the book images. Though Tesseract is an excellent OCR model, it is imperfect and makes many errors. Thus, a spell-checking model was employed to correct spelling errors in the Tesseract produced text using the Python library TextBlob (`https://textblob.readthedocs.io/en/dev/`). The data are then split into sentences and then into data structures compatible with each NLP library used for each task. Some of the libraries require specific pre-processing steps, such as the remove of stop-word or converting all characters to lower-case. Other libraries, specifically BERTopic, require the text in nearly unedited form as it learns con-

text from details such as character-cases, pronouns and articles.

3.2 Computational Environment

The computations were carried out on a server running 64-bit Debian GNU/Linux version 11 with X86_64 architecture. The server consists of a 24 Intel CPUs, each operation at 2.6 MHz with a single thread per processor. The hardware did not include a GPU.

3.3 Python Modules

Several Python modules were written for this project. These modules handle specific functions during the topic and NER extraction process. These modules are as follows:

- **parse.py:** This module performs all preprocessing operations including spelling correction, removal of stop-words, and removal of artifacts including page number and contents pages.

- **ngrams.py:** This module extracts topics from the processed text using n-grams models where $n = 1, 2, 3$, and returns the topics along with model confidence values.

- **ldatops.py:** This module extracts topics from the processed text using an LDA model in the Gensim library, and returns the topics along with model confidence values.

- **run_bert.py:** This module extracts topics from the processed text using the transfomer based BERTopic library, and returns the topics along with model confidence values.

- **run_spacy.py:** This module extracts named entities from the processed text using the spaCy library, and returns a list of the entities extracted.

- **threader.py:** This module is optional code for parallelizing the process. The code gathers a list of all files in a source directory (defined in a configuration file), and creates separate instances of topics.py to perform simultaneous processing of multiple files. The number of thread used (number of simultaneously processed books) can also be modified in the configuration file.

- **topics.py:** This is the main processing code for the NLP portion of the experiments. This program calls the other modules, and performs all related operations. The topics produced by each module are then compared. Topics produced by multiple modules are combined and their confidence values averaged. Named entities are compared to the associated text document to calculate a document frequency value, which is then normalized to create a makeshift confidence value. All topics and entities are then sorted by confidence value, and the top 25 topics and entities are then saved to data files for each book.

- **topic_compile.py:** This module compiles all output data files into a single CSV file.

3.4 NLP Libraries

All software was written and executed using Python 3.11.5 in an Anaconda 23.11.0 environment. Several libraries were require to perform the necessary NLP tasks including:

- **PyTesseract:** This OCR engine allows for text to be extracted from images. In

this use-case, the images are individual book pages.

- **PyPDF2:** This library, for the manipulation of PDF objects, allows PDF files to be split into individual pages, and then recombined post-processing.
- **pdf2image:** This library allows for pdf pages to be converted into images.
- **Pillow:** The Pillow library (PIL) is used for image object manipulation.
- **Regular Expressions (re):** This library allows for the use of regular expression in Python programs.
- **Pandas:** This library for dataframes was primarily used to combine and sort data produced by the various NLP tasks, as well as for producing the output data file (CSV).
- **TextBlob:** This correct() method in this library is a spellchecking model that was used here to correct spelling errors produced by Tesseract.
- **NLTK:** One of the most well-known NLP libraries, NLTK was primarily utilized to poduce N-Gram models, as well as for their stop-word corpus
- **spaCy:** Another well-known NLP library, spaCy was used here for it's NER model, en_core_web_sm. Unfortunately this model does not produce probabilities for each prediction. As such, term frequencies were used instead.
- **Gensim:** Yet another popular NLP library, Gensim was utilized for its corpora tokenizer and Latent Dirichlet Association model, LdaMulticore().
- **BERTopic:** BERTopic is a BERT (transformer) inspired library for topic modelling.
- **Transformers:** The Transformers library, by HuggingFace, enables the use of a vast array of transformer based models. This model was used for some early-stage experimentation using the text summarization pipeline.

3.5 Data Processing

Data are processed in three main phases: OCR and text extraction, NLP processing, Data aggregation. The process begins with PDF files containing images of all pages in an individual book, and ends with a CSV file containing book details, extracted topics and keywords, as well as extracted named entities. Additionally, a small selection of books have been processed using a hierarchical summarization scheme. Each stage of processing was completed using a naive parallelization approach, where books were processed 4 at a time using Python's *multiprocessing* library. Processing time was not meticulously timed, but OCR and text extraction took approximately one week, NLP Processing to roughly 5 days. The data aggregation step was trivial in comparison and only took a matter of seconds.

3.5.1 OCR and Text Extraction

During this processing phase, each book is split into individual pages. Each page is then processed using PyTesseract. The produced text is then recombined, as a text overlay, with the page image and converted to a single-page PDF. These pages are then recombined to produce a PDF with searchable text. Once this process is complete, the files are again processed page-by-page to extract the text which is then saved to a .txt file for further processing.

3.5.2 NLP Processing

The text files produced in the previous phase contain all text produced during the OCR process, as well as pager markers. These markers, while not used for these experiments, allow for page-level analysis in the future. Each text file is pre-processed in the following order:

1. Text data are loaded from the file.
2. The full text is spellchecked and corrected using the TextBlob library.
3. Page markers, headers, page numbers, contents pages, preface pages, and index pages are removed. The remaining text is combined into a single string. For clarity, we will refer to this text as data structure 1 (DS1).The data structure is maintained in this state for use with BERTopic at a later stage.
4. A copy of DS1 is created and regular expressions are then used to remove stop-words, symbols, and most punctuation. All characters are then converted to lower-case.
5. The text is then split into a list of sentences for NLP processing. We will refer to this as data structure 2 (DS2).
6. DS2 is then used to produce N-Gram models (NLTK) for n=1,2,3, an LDA model (Gensim), and and NER model (spaCy). The N-Gram and LDA models are then used to produce common terms and topics from individual books along with model confidence values. As mentioned in Section 3.4, the spaCy NER model does not produce confidence or probabilities for it's predictions, as such, term frequencies were produced by calculating the number of instances for each term and dividing by the total number of entities extracted.
7. DS1 is then inferenced using BERTopic which produces a list of topics found in each book, along with model confidence values.

3.5.3 Data Aggregation

The terms and probabilities produced from each model, except for the NER model, are then compared. Terms produced by multiple models are aggregated into a single term with their model confidence values averaged. This allows for the inclusion of all produced terms and ensure that terms produced by a single mode, but with very high accuracy, are considered. Terms are then sorted by confidence value, and the 25 terms with the highest confidence are selected as the output. As the NER does not produce confidence values, the raw frequencies calculated for this model should not be combined with confidences from the other models. Thus, these terms and frequencies were kept separate. As with the other models, the NER produced terms were sorted, by term frequency, and the top 25 terms were selected as the output.

The resulting data are then saved to a data file for each book in the corpus. Each data file contains:

- The book's file name.
- The book title.
- The top 25 terms and associated confidence values produced from the N-Gram, LDA, and BERTopic models.
- The top 25 terms and associated term frequencies produced from the NER model.

The data files are then aggregated, using Pandas, into a single CSV file that contains all the produced information on each book in the corpus.

3.5.4 Hierarchical Book Summarization

An additional experiment was performed on a very small number of books: hierarchical book summarization. For this process, the pipelines feature of the Transformers library by HuggingFace was used. In specific, the summarization pipeline, which takes in a string of up to 1024 characters, and produces a summarization of specified length. In my experiments I chose an output length of 128 characters. This results in a summary that is 1/8 th the size of the input text. These experiments were performed using the default summarization model, distilbart-cnn-12-6, available at `https://huggingface.co/sshleifer/distilbart-cnn-12-6`.

To perform hierarchical summarization, we begin by splitting the pre-processed text into chunks of 1024 characters. Each chunk is then summarized using the pipeline. The produced summaries are then concatenated into a single string. This new string is then, again, split into 1024 character chunks, and processed by the summarization pipeline. This process is continued until the aggregate summary is 15 sentences or less.

4 Results & Analysis

As mentioned in an earlier section, fifteen books in the corpus were unsuitable for the OCR and NLP processing. As such only 925 books were processed. All books were successfully processed and produced some reasonable topics and entities from the texts, but the methodology is far from perfect. Table 1 shows the output for a books in the corpus. This is a relatively successful example as the terms produced are nearly all related to the subject matter of the text. Other books produced lower quality results, often due to atypical formatting or the inclusion acronyms, heavy abbreviations, or non-standard characters such as **â€˜â¬** being confused for topics. Table 2 details the lackluster results from a book on school cost.

The hierarchical text summarization, only performed on a few books, produced surprisingly good, but unusual summarizations. For example, the hierarchical summary of the book "Psychology and Life", by Hugo Munsterberg, (1899 Houghton Mifflin Co.) reads, *"psychologists aim to produce psychological states of delight and enjoyment. psychic law denies that will acts are for us objects of understanding, interpretation. whatever we think, feel, will, and act can, as psychophysical process, be determined by vitality of active and latent causes. psychologists aim to produce psychological states of delight and enjoyment. psychic law denies that will acts are for us objects of understanding, interpretation. whatever we think, feel, will, and act can, as psychophysical process, be determined by vitality of active and latent causes. childhood lasts from the cradle to the end of adolescence, usually to the twenty-fiveth year. pathological abnormalities in the child's mental life are psychological decidedly instructive. pleasure is not a state of mind, it is a mental state of the mind. psychologists and historians cannot help each other as long as they stick to their own aims. the historian selects his material to transform the reality into a theological system. many mediums believe themselves to be under control of departed souls."* While this is not a typical summary, the approach does seem to capture the lager topics from the text.

Topic	Probability	Entity	Probability
tolls	0.726	the TimeCard Back	0.320
toll	0.721	Julie	0.109
varnishcolored	0.713	Wagon	0.092
stains	0.701	Sharp Corners	0.062
varnished	0.659	Couorine	0.062
ounces	0.629	FUSE	0.062
painting	0.616	Application of Later	0.016
homes	0.606	Onrnbabyss	0.015
airbrush	0.594	Kalsomine	0.013
house	0.588	Faint Formula	0.013
coats	0.542	Blue y.G. 14	0.012
stainsit	0.537	Pulex Speedster	0.010
black	0.536	Range	0.010
roadster	0.535		0.009
handbrushing	0.527	Rose Bengal	0.009
handbrush	0.518	Clothes	0.008
auto	0.506	peter k Sanitation	0.008
es	0.499	Suffer	0.007
stainsfor	0.491	ULETHIE	0.007
motor	0.490	Screw	0.007
ce	0.488	Detail	0.005
bleached	0.486	John Bones	0.005
indigo	0.485	SawBlade	0.005
lead	0.485	Wander.â€"The	0.005
os	0.469	Air	0.005

Table 1: Topics and named entities with their probabilities. Extracted from the book, "Toy Making" by D.M. Mitchell, 1921.

Topic	Probability	Entity	Probability
"	0.431	SS	0.118
of	0.420	Supyeredg	0.048
saunlignadxa	0.328	SL	0.035
scurrying	0.287	GNV	0.032
elysnpuy	0.282	S1	0.028
spunoiy	0.270	Org	0.018
shunlignadxa	0.265	Art	0.018
on	0.265	Surmerq	0.017
ys09	0.264	PUY	0.015
ofs	0.260	OLL	0.015
put	0.132	GULP	0.014
oe	0.125	DER	0.014
see	0.091	LOL	0.013
is	0.072	SISATVNV	0.012
he	0.054	SIGE	0.012
go	0.049	eLysnpuy	0.012
days	0.038	OST	0.011
see see	0.035	PPIOUDE	0.010
8 8	0.035	TOL	0.010
â€˜of staff 1	0.025	Saliaans	0.009
00 00	0.024	SET	0.008
9 9	0.024	SINVISISSV	0.007
say	0.022	PUL	0.007
â€˜of 1	0.021"	OLT	0.007

Table 2: Topics and named entities with their probabilities. Extracted from the book, "Gary Public Schools Costs" by Frank P. Bachman and Ralph Bowman, General Education Board, 1918.

5 Conclusion & Future Work

In conclusion, this approach to topic extraction and summarization for a digital book corpus is surprisingly effective, but much can be done to improve on the method. One of the largest challenges during this process, a challenge common to NLP tasks in general, is pre-processing of the text data. Numerous formatting variations in a book corpus make it an immense challenge to produce a uniform data structure to be utilized NLP libraries. Although much effort went into removing artifacts such as page numbers, index and contents pages, footers, and headers from each text, some of these artifacts still managed to find their way into the processed text. In some instances, these artifacts were even mistaken to be topics by the NLP models. Further pre-processing techniques may ensure unwanted text is more thoroughly removed.

An additional area that can likely be improved upon, is the NLP models themselves. No model is perfect, and all can be improved upon. For these experiment, many options and parameters for individual models were left unexplored. Fine tuning these parameters may yield superior results. Additionally, the process for combining probabilities for each model was a naive approach. Further analysis may inform a method superior to a simple averaging of probabilities. A glaring issue with the results comes from the frequencies produced for the named entity recognition model. As the spaCy NER engine does not produce probabilities associated with each entity, term frequency was used in place of probability. This, another naive approach, can certainly be improved upon.

The most interesting approach from these experiments is the hierarchical summarization technique. These experiments suggest the approach can yield surprisingly good im-

prints of the subject matter. Only the default model was used in these experiments, but several models are available. One bottleneck, which prevented further experimentation, was the lengthy processing time. A single book required hours of processing time using HuggingFace's summarization pipeline. This can be easily improved upon by downloading various models and performing the summarization locally. Additional performance can be gained by employing GPU acceleration. This is an interesting approach, and one that warrants further exploration.

References

[1] Rania Albalawi, Tet Hin Yeap, and Morad Benyoucef. "Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis". In: *Frontiers in Artificial Intelligence* 3 (2020). ISSN: 2624-8212. DOI: 10.3389/frai.2020.00042. URL: https://www.frontiersin.org/articles/10.3389/frai.2020.00042.

[2] S. Sbalchiero and M. Eder. "Topic modeling, long texts and the best number of topics. Some Problems and solutions." In: *Quality and Quantity* 54.4 (2020), pp. 1095-1108 – 1108. ISSN: 15737845. URL: https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-85079875858&site=eds-live&scope=site.

[3] S. Avasthi, R. Chauhan, and D.P. Acharjya. *Processing Large Text Corpus Using N-Gram Language Modeling and Smoothing.* Vol. 166. Lecture Notes in Networks and Systems. (1)Amity University: Springer Science and Business Media Deutschland GmbH, 2021. ISBN: 9789811596889. URL: https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-85101561762&site=eds-live&scope=site.

[4] Uttam Chauhan and Apurva Shah. "Topic Modeling Using Latent Dirichlet allocation: A Survey". In: *ACM Comput. Surv.* 54.7 (Sept. 2021). ISSN: 0360-0300. DOI: 10.1145/3462478. URL: https://doi.org/10.1145/3462478.

[5] Roman Egger and Joanne Yu. "A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts". In: *Frontiers in Sociology* 7 (2022). ISSN: 2297-7775. DOI: 10.3389/fsoc.2022.886498. URL: https://www.frontiersin.org/articles/10.3389/fsoc.2022.886498.

[6] G.B. Mohan and R.P. Kumar. *A Comprehensive Survey on Topic Modeling in Text Summarization.* Vol. 373. Lecture Notes in Networks and Systems. Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham: Springer Science and Business Media Deutschland GmbH, 2022. ISBN: 9789811687204. URL: https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-85126178749&site=eds-live&scope=site.

[7] Aly Abdelrazek et al. "Topic modeling algorithms and applications: A survey". In: *Information Systems* 112 (2023), p. 102131. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2022.102131. URL: https://www.sciencedirect.com/science/article/pii/S0306437922001090.