Data Minding (and Mining) for Modern Science

A THESIS PRESENTED
BY
ROBERT MCKENZIE
TO
THE DEPARTMENT OF STATISTICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELOR OF ARTS (HONORS)
IN THE SUBJECT OF
STATISTICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
APRIL 2023

Thesis advisor: Xiao-Li Meng Robert McKenzie

Data Minding (and Mining) for Modern Science

Abstract

ChatGPT, a chatbot developed by OpenAI, was released in November 2022. In the short few months since that release, statisticians and data scientists have been puzzled by how these tools will affect their profession. While some consider large language models to be achievements of data engineering, not data science, they still promise myriad possible benefits for statisticians. This paper will show how statisticians can use large language models, in particular with fine-tuning, to assist them in a key and neglected role: as data minders. Data minding was first proposed in *Enhancing (Publications on) Data Quality: Deeper Data Minding and Fuller Data Confession*, Meng (2021), arguing that it would benefit research for statisticians to treat data as products of research ”in and of themselves”, rather than as inputs for analysis. Through a rigorous investigation of the entire data lifecycle for a research dataset, from conceptualization and collection, pre-processing, management and storage, data curation and provenance, through to the traditional outputs of research, methods and analysis, results, and visualizations, we can identify adverse influences on data quality and promote deeper and more explicit clarity from authors about their data, a process called data confession. Data minding is a lengthy and time-consuming process, and relies upon statisticians taking time away from their own research to scrutinize others, an unappealing proposal for many. As of this paper, no attempt to conduct data minding at scale exists. This paper aims to change that using the transformative power of language models to develop automated processes to simplify the data minding process. Focusing on the journal *Science*, the pre-eminent journal published by the American Association for the Advancement of Science, this paper will model the distribution of paragraphs in published papers across the different categories of the data life cycle from 1950-2020. With this tool, we can track changes in data confession, and identify trends and moments in the development of science that have led to changes in how authors' talk about their own data. We find that attention to methods and analysis have remained constant across papers over the period studied, but that, accompanying changes in technology, the percentage of a paper devoted to data pre-processing, management and storage, and visualization have all increased over this time period. While paper length has increased over time, this growth has come at the expense of paragraphs devoted to exploring data conceptualization, collection, and curation.

Contents

List of figures

Acknowledgments

I would like to thank a number of people for their help with this project. Firstly, to my amazing friends and fellow-thesis writers who have endured me talk about data quality for months on end: Sam Sharfstein, Eric Hansen, Chris Scazzero, Jean-Luc Henreaux, and Quinn Perrini. For the words of encouragement, the commiseration, and, most importantly, the celebrations to come, I am grateful.

I would also like to thank my parents, Michael and Francine, for their endless support. I would not be here without you. You are the best supporters I could ever ask for.

Finally, I owe a substantial debt to my advisor, Professor Xiao-Li Meng. Without Professor Meng, I never would have even written a thesis, and I am grateful you have pushed me to complete this project. I have so enjoyed our discussions over the last few months, and am excited to see where we can take this work. Professor Meng has consistently found time every week to discuss the ups and downs of this journey, and your guidance, mentorship, and constant feedback has been a blessing.

1 Introduction

1.1 Motivation

Text classification is one of the oldest problems in natural language processing. In the literature, there are two principal methods for classifying text into n categories: rules-based methods, and machine learning based methods [14] [15]. There are a variety of applications of text classification and data mining in systemic review. However, existing methods lack the granularity for systemic analysis without human input.

This paper proposes a new methodology, based on fine-tuning large language models which can develop much more sophisticated semantic embeddings than traditional machine learning algorithms like the Support Vector Machine. We present an initial attempt at classifying text from research papers based on which section of the data life cycle [28] the text pertains to using this novel method. Large language models, which contain tens of millions to billions of parameters and are trained on enormous datasets consisting of billions of words, can develop sophisticated embeddings that enable classification with much higher accuracy than traditional models, with faster and more efficient training.

As large language models become ever larger, with wider applications for research [6], data minding, the rigorous investigation of the data lifecycle for a research dataset, aiming to identify adverse influences on data quality, will become even more important. Scientific articles make up a substantial portion of the high-quality text data used to train these models: improved data confession in training papers will help the models perform ever more sophisticated data minding [6] [13].

1.2 Outline

We will begin with an exploration of the existing literature and background research on this topic, focusing on three aspects of the work: Data Minding, Meta-Analysis with Text Mining, and Fine-Tuning Foundation Models for Text Classification. In Section 3, we will outline the data mining process used, including descriptions of the custom scripts used to download and process data. From there, we will discuss our dataset. This will consist of a description of the data, as well as a thorough data confession, exploring the limitations and justifications of the decisions we made building the dataset.

In Section 4, we will describe the process of analysis on our data, including fine-tuning a large language model for our specific text classification task, and denoising data using the Debauchies wavelet. Finally, in Section 5, we will describe the results of our analysis in the context of papers from *Science*, and open discussion on how editors and authors could adjust their practices based on these results.

2

Background

2.1 The Data Life Cycle, Data Quality, and Data Minding

In her 2019 article ”Data Life Cycle: Definition and Key Stages,” Dr. Jeannete Wing defines the data life cycle as a framework for understanding the various stages that data goes through during its existence [28]. According to Wing, the data life cycle consists of seven stages:

1. Data Generation: Both humans and nature generate data. Every search query, every tracked decision, every piece of experimental evidence provide some information that can be used for analysis.

2. Data Collection: The majority of data we generate is not collected, either because it is not useful or the data generation stream is too fast for analysis to keep up. Researchers must decide to impose a filter to catch a certain type and amount of data.

3. Data Processing: Data processing can refer to anything from cleaning, to wrangling, to compression. These are tasks conducted to enable analysis on collected data.

4. Data Storage: Traditional data storage, in CSV files or relational databases, might seem simple, but, especially for long-term, infrequently accessed storage, there are novel technologies being deployed.

5. Data Management: There are many different ways to store and access different data types: audio, text, vectors, both structured and unstructured; the variety of data types is astounding. To maximize accessibility and ability to build upon and modify data across these axes of variation, we need to use a variety of different metadata.

6. Analysis: When people think of data science, they typically think of data analysis. From machine learning to linear regression, data analysis consists of the methods used to infer causality from data.

7. Visualization: The famous aphorism states that a picture is worth a thousand words. Visualization is the art and science of

communicating results in clear and concise way.

8. Interpretations: Interpretation gives contexts to the methods and visualizations produced from a dataset. This includes the ramifications and implications of conclusions.

Every step in this process informs every other step. There are continuous feedback loops, and as data progresses through the cycle, it should inspire re-evaluations of earlier steps. As well, ethics and privacy concerns underpin any research involving personal or sensitive data.

Meng builds on this concept of data life cycle through introducing the related concept of data minding. While the life cycle is common in industry, it has yet to be fully incorporated into academic reporting on data and data quality. Data quality is a somewhat nebulous concept, especially in research. A simple definition of data quality is a measure of the degree to which data is fit for its intended purpose. High-quality data is reliable, trustworthy, and provides useful insights and information. Data quality is both purpose-dependent and method-dependent, as is clear in research on data defects [12]. Similarly, there are both study-invariant aspects of data quality, for example the process behind collecting data for a public dataset, and study-specific considerations, for example understanding how a specific pre-processing step or storage methodology might bias a particular study. The process of data minding asks for deeper reflection into every step of the data life cycle. However, without sufficient documentation on data conceptualization, collection, pre-processing, curation or provenance, the process of data minding would be severely limited. Currently, statistical journals do not put enough emphasis on ensuring that publications address all of these categories of interest [13] [21].

Meng argues for two substantial changes to the way researchers discuss data. Firstly, there is a case for a specific inclusion of a ”data minding” step into the data life cycle, prior to data analysis. This process of data minding would focus on the internal discussion of authors and the life of their data up to that point. This step would help researchers identify the limitations of their data, inspiring more processing or collection, and would inform the methods and analysis that would best fit the dataset. Second, Meng notes that, while every step in this process is contingent on all previous steps, authors neglect to articulate the decisions they make early in the process, delving deep into their methods or results while discounting the collection, processing, and management that can adversely effect results. This makes external data minding, as well as meta-analysis of data quality, next to impossible. Journals and editors should encourage submissions that include a thorough accounting of the data life cycle. In this way, it will be easier to detect when low-quality data is packaged in such a way as to lead to deceptively convincing results.

For clarity, we have made some adjustments to the data life cycle for more applicable classification. Management and Storage, two of the most neglected categories, have been combined into one category. In addition, an additional category has been defined specifically for research datasets. Research, particularly in the social sciences, often uses data collected by a government, non-profit, or academic institution, and applies some new analysis or methodology to derive novel insights. These datasets are often publicly available, but are still subject to the same data life cycle as any other. Therefore, we have added a category for data curation and provenance.

2.2 Meta-Analysis with Text Mining

As text mining has grown as a field, with applications throughout academia and industry, one under-utilized application of text mining is meta analysis. While some systemic reviews using text exist, these studies are often limited in scope and subject to the same issues as traditional meta-analysis: namely, a large time and energy requirement from domain experts who could be focused on other research [16] [11] [4]. Supplementing quantitative analysis, like bibliometrics, with in-depth content analysis requires enormous involvement from authors.

Some prior work exists aiming to leverage the power of text mining to save researchers' time and energy [11] [16]. More complete meta-analysis has used text-mining to identify key concepts or ideas in a specific sub-domain, like xerostomia [3]. Meanwhile, both in industry and academia, data mining technology is advancing rapidly. Modern tools like Python and associated libraries, as well as more powerful computation, promise to enable a new era in data mining.

One of the most promising areas of development is in medical meta-analysis. There is extensive overlap in medical studies, often with contradictory results. Extracting data from publications to compare results across studies has been shown to be effective even when authors don't include raw data in their publications. This paper owes a debt to Pedder et al (2016), who have published an extensive guide on data mining in medical studies [19]. They introduced a series of best practices for *Data extraction for complex meta-analysis (DECiMAL)*. The DECiMAL guide is a tool for data minders and reviewers, intended for reading before data mining is conducted, and is strongly recommended by this author. Their recommendations for consistent data collection are particularly valuable, as well as the generalizability of the guide for use outside medical studies to any type of data mining for meta-analysis. The DECiMAL authors, however, are wary of text-mining and text data in general, accurately identifying text as an inconsistent and unreliable metric when compared to meta-analysis on binary or categorical variables.

Pham et al (2021) leverages text mining and classification to synthesize knowledge across abstracts to create a ”semi-automated” systemic review workflow, using the specific example of insulin formations for Type 1 diabetes [20]. They find their workflow to be a sensitive, precise, and efficient alternative to the recommended practice of screening abstracts with two reviewers for inclusion in systemic review. They also report remarkable metrics for their classifier. They report their classification tool had 88 percent sensitivity, 99 percent specificity, 71 percent precision, an F1-score of 79 percent, and a 63 percent reduction in the total hours needed to select abstracts. While this workflow is impressive, it is grounded in traditional methods from natural language processing . This includes extensive steps for processing text data, including tokenization, lemmatization, and word stemming, and the creation of term-document matrices [4] [16]. These techniques are time-consuming and both project and method dependent.

Pham et al aim to demonstrate that these traditional tools have become sufficiently advanced to be competitive with manual review. Several real-world systemic reviews have been conducted using these methods, but, for the most part, the work remains in a proof-of-concept stage, despite the possible gains in work reduction and the impressive accuracy of classifiers. Paynter et al, in interviews with eight key informants, aim to understand why text mining has struggled to gain traction among researchers for systemic review [18]. They identify the following barriers to widespread adoption:

1. Cost and time efficiencies: While the informants were univerally interested in the advantages of text mining, they were unsure how to run cost-benefit analysis. Determining the software and staffing costs to create a new text mining tool were difficult to calculate for the interviewees because many of the tools they use have been developed over a long time period, ad-hoc, and varied widely in development time and accuracy.

2. Integrability: Some informants were concerned about how text mining could be integrated into their existing workflows, both in terms of making it easier for staff to work without moving between multiple systems to complete a task and creating a text mining tool they can use without domain expertise or technological know-how.

3. Organizational and technological barriers: Informants mostly reported high staff acceptance to adopting text mining. Despite this, staff were also mentioned as an organizational barrier, specifically librarians/information specialists who may feel their work has been deskilled. In addition, they note that these tools are very much in their infancy, and so it remains unclear how and where they can be best deployed.

4. Lack of Metrics: Overall, informants were hopeful about the future of text mining; but they noted that a lack of standardization was deeply problematic. Without ways of communicating the value of text mining, encouraging adoption remains difficult. In addition, developing time and accuracy

metrics to allow formal evaluation would help researchers pick which tools to use.

This paper aims to leverage the power of developments in machine learning to propose a new framework for text mining in meta-analysis. Our goal is to expand on the potential of text-mining for systemic review. Rather than focusing on text-mining as a means to an end, saving time for researchers focused on a specific topic, we aim to use data mining as a form of exploratory analysis, exposing possible topics for future work. Through a custom text-mining script, we were able to download text files from over 5,000 academic papers from leading journals. The script is included in Appendix C, and we will discuss these downloads, our dataset, and our goals with this dataset in more depth in the relevant section.

There are some ethical concerns with text mining in research. In particular, mining sensitive documents, such as health records or social media logs, can, even when anonymized, still lead to potential abuse. [9]

Though we considered the ethical concerns of mining research articles, there are two important considerations here. One, these papers are submitted for publication to develop knowledge. Meta-analysis, and especially our goal - encouraging more complete data confession in journal submissions, is devoted to the same goal. Second, there is no personally identifying data used here: all data is abstracted and viewed as a component of the whole. Save a few examples drawn from the text, there is no proprietary information included here.

2.3 Fine Tuning using BERT

Since the release of BERT in 2018, data scientists have explored its abilities in various Natural Language Processing tasks, including summarization, entity resolution, and text classification [27] [8].

Prior to BERT, natural language processing methods could be neatly differentiated into two categories: linguistic approaches, also known as rules-based approaches, where domain experts identify certain key features which are then used to classify text. These features could be keywords, n-grams, grammatical or syntactic features, and many more. Linguistic approaches have achieved moderate levels of success on these NLP tasks, especially for domain-specific tasks. The other category, machine learning, involves inferring key features from annotated text, usually converted to vectors through a bag of words or term-frequency inverse-document frequency vectorization.

Both approaches have their pros and cons. However, the rise of computationally intensive algorithms, such as BERT, and, more recently, GPT-3 and GPT-4, has led to machine learning methods improving precision and recall more quickly than linguists could keep up [27] [6] [8]. Crucially, these methods require large corpora of text for training to enable effective prediction, whereas rules-based approaches need no training.

There have been a number of interesting papers published comparing BERT to classical NLP approaches [24] [15]. While it would be an over generalization to say that BERT is definitively superior, the results are clear. Given sufficient training data and time, BERT outperforms classical NLP on text classification tasks.In this paper, we will compare BERT classification to classical NLP as a benchmark, but we will not focus on classical text classification.

To understand the classification discussed in the next section, we must first ensure an understanding of the BERT training process. BERT's architecture is made up of a bi-directional Transformer encoder first described by authors at Google Brain. BERT is made up of 12 layers, each consisting of 2 sub layers: the first sub layer is multi-head attention mechanism and the second is a position-wise feed-forward network. The two sublayers are connected and normalized using the following function:

$$LayerNorm(x + Sublayer(x))$$

To facilitate this connection, each sublayer, as well as the top layers, have a limited output size, with only 512 dimensions.

Starting with the first sublayer, the self-attention mechanism consists of mapping a query and a set of key-value pairs to an output, where the query, the keys, the values, and the output are all vectors. To simplify notation, we can imagine a matrix of packed queries **Q**, and associated matrices of packed keys and values, **K**, and **V**.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The denominator here, d_k, is used for scaling as our key-value matrices become large. It is worth spending a little time discussing the softmax function. Despite its name, softmax is not a maximum. The softmax function is a mathematical function that takes a vector of real numbers and returns a probability distribution over the same set of numbers. Specifically, given a vector of real numbers $z = [z_1, z_2, ..., z_k]$, the softmax function computes a new vector $s = [s_1, s_2, ..., s_k]$, where:

$$s_i = e^{z_i}/(e^{z_1} + e^{z_2} + ... + e^{z_k})$$

The denominator in this formula is the sum of the exponential values of all the elements in the input vector. This ensures that the output vector sums to 1 and thus can be interpreted as a probability distribution.

Instead of performing a single attention function with 512-dimensional keys, values and queries, the team behind BERT found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding dv-dimensional output values. These are concatenated and once again projected. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. In this notation, the projections are parameter matrices $W_i^Q \in R^{d_model x d_k}, W_i^K \in R^{d_model x d_k}$and $W_i^V \in R^{d_model x d_k}$, and finally, $W^O \in R^{hd_v d_model}$

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)WO$$

where

$$head_i = Attention(QW_i^Q, KW_i^K i, VW_i^V)$$

In general, training BERT for specific NLP tasks involves two steps:

1. Pre-training: based on a large corpora of unlabelled text, BERT uses the self-attention mechanism to develop weights for text prediction. BERT tokenizes text using the WordPiece embedding framework, converting text into vectors of floats. There are two main methods for pre-training BERT: Masked Language Modelling and Next Sentence Prediction. In Masked Language Modelling, some percentage of tokens in a sentence are hidden, and the model tries to predict the masked tokens. Next Sentence Prediction consists of feeding BERT two sentences, **A** and **B**. 50 percent of the time, B will follow A in the original text, and 50 percent of the time, B is replaced with a random sentence chosen from elsewhere in the corpus.

2. Fine-tuning: the model is initialized with the pre-trained weights, which are then fine-tuned using a smaller, annotated data set. Because of the simplicity and applicability of the self-attention mechanism, this is actually a very straightforward task. We simply insert our labelled dataset and optimize to reduce a designated loss function, as in traditional supervised learning.

For text classification, the optimal method for fine-tuning BERT takes the final hidden state s of the first token [CLS] as the representation of the whole sequence [**?**]. A simple additional softmax classifier is added to BERT to predict the probability of label c:

$$p(c|s) = softmax(W^s)$$

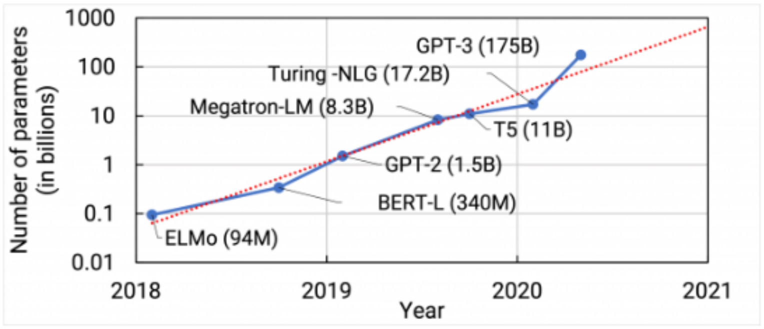
(1) where W is the task-specific parameter matrix. We fine-tune all of the weights in W to maximize log probability of the correct label c given the last hidden state s.

There are a number of possible choices for pre-trained models available for fine-tuning, of which BERT is just one. Figure 1, from NVIDIA, shows the rapid growth of large language models over the last few years, enabled by advances in computational power and data mining, which can create larger training sets [5]. However, these models, most of which are released by private, for-profit organizations, tend to eschew in-depth descriptions of their architecture. BERT, one of the earlier models released, does include this in-depth description. While BERT was not the only language model used for fine-tuning in this paper, we can assume that similar models use a similar process. Unfortunately, we can't guarantee this, since many models can only be interacted with through an API, leaving the actual fine-tuning process opaque. LLAMA, released by Facebook Research in 2023, has made their complete architecture, as well as all 65 billion parameters included in the model, available to research, and we are hopeful that this will encourage future researchers to publish their methodologies as these models continue to increase in scale and ability [26].

It is also worth briefly discussing the potential of one or few-shot training in fine-tuning tasks [6]. One-shot learning is a type of machine learning that involves learning from just one example of a new class or category. In other words, the algorithm is given only one example of a new object or concept, and it must use that example to classify or recognize other instances of the same object or concept. As you might expect, few-shot learning is the same process, but with multiple examples given. One-shot learning is particularly useful in scenarios where obtaining large amounts of labeled data for each new class or category is difficult or impossible. For example, in some medical applications, there may only be a small number of rare diseases, and obtaining enough labeled data for each disease may be difficult. One way to accomplish one-shot learning is to use a similarity metric to compare the one example of the new class to other examples in the training set, and then use that comparison to classify new instances of the new class. Another approach is to use generative models, which can be used to generate additional examples of the new class based on the one example that was given. We use a form of few-shot learning, which we will discuss in our methodology, to develop a larger training set, but there is also the possibility of using this type of learning directly into classification, without fine-tuning at all. For example, we can prompt a model with the following text:

> You are a model tasked with classifying a paragraph of text into one of the following seven categories: Data Collection, Pre-Processing, Data Management and Storage, Data Curation and Provenance, Methods, Visualizations, Interpretations. Below is an example of a labelled paragraph: Text: "In this study, we used a combination of pre-processing techniques to prepare our dataset for machine learning analysis. First, we performed data cleaning to remove any missing or erroneous data points. Next, we normalized the data using z-score normalization to ensure that each feature had a mean of 0 and standard deviation of 1. We also performed feature selection to identify the most relevant features for our analysis, using a combination of correlation analysis and mutual information-based feature selection. Finally, we performed data augmentation to increase the size of our dataset, which involved generating synthetic data points using a generative model. Overall, these pre-processing techniques helped to improve the quality of our dataset and reduce the risk of overfitting in our machine learning

Figure 2.3.1: Trend of state-of-the-art NLP model sizes with time.



models." —END OF TEXT—

Label: Pre-Processing

Once the model has been prompted, we can enter new paragraphs, and the model will output labels for them. Research by OpenAI suggests that few-shot learning can be competitive with fine-tuning on a number of text classification tasks. However, our results do not support that conclusion, which we will discuss in detail in Appendix B. In addition, we have devised a novel sampling method to expand our training data set. With this sampling method in place, we are not concerned with accessibility of training data, and so it was decided not to pursue few-shot learning as a classification tool in this work.

3

Data

The data used for fine-tuning comes from a variety of different sources. The seven categories identified as relevant for data minding are, in order of the data life cycle:

- Data Collection and Conceptualization
- Data Management and Storage
- Data Pre-Processing
- Data Curation and Provenance

- Methods and Analysis
- Results and Interpretations
- Visualization and Communication

Meng finds that across a limited selection of papers, the amount of text devoted to each category is unevenly distributed. This isn't a problem in and of itself: methods and results should, in theory, make up the bulk of the paper. Where we encounter problems is when authors completely neglect discussing one or multiple categories. Not only does this complicate the task of data minders and other reviewers, at every step of the life cycle there are opportunities for adverse influences which will affect every subsequent step; for authors to neglect discussing their decisions during data pre-processing calls into question their analysis. Similarly, neglecting to discuss the provenance of a dataset limits the possible scope of their conclusions. Meng draws from some specific examples, looking at papers which analyze datasets over time, like the Global Terrorism Database.

The first question to examine is how to separate papers into discrete blocks that can be categorized. Meng breaks papers into pages, but we sought a more granular definition, deciding instead to divide papers first by section, and then, finally by paragraph. Both pages and section proved to be insufficiently detailed: many papers will group multiple categories into the **Data** section. The paragraph was an intuitive and easy to process unit for classification. The paragraph is a sufficiently small unit that even throwaway references to data curation or any of the other neglected categories should trigger accurate classification.

At first, data was manually curated to create a training set of 25 examples per category. Papers were sourced from 1950-2020, from 5 major journals: *Science*, *Nature*, *American Economic Review*, *New England Journal of Medicine*, and *The Lancet*. These are the 5 journals with the highest impact factor on Google Scholar, and should, in theory, reflect only the most consequential and rigorous research. We used a simple three-step random sampling process to identify papers for inclusion. Taking *Science* as an example, the sampling process was as follows: 1. Four random numbers are generated to select a Journal, Volume, Issue, and Article (in order of appearance in the issue). 2. The randomly chosen articles are downloaded into a corpus. 3. The corpus is broken down into paragraphs using a combination of Optical Character Recognition and Text Extraction using the `pytesseract` and `textract` packages. Paragraphs were randomly sampled, and subject to manual classification into one of the seven categories. Not every paragraph was relevant to a category: these included references, background, and literature reviews which failed to be easily classified. While the vast majority of paragraphs met our criteria for classification, 6 percent of paragraphs sampled this way did not. Rather than include an eighth category into our classification for these "Other" paragraphs, it was decided to discount these paragraphs from training. Since these paragraphs had little in common besides their resistance to classification, it would have introduced substantial noise into our training set.

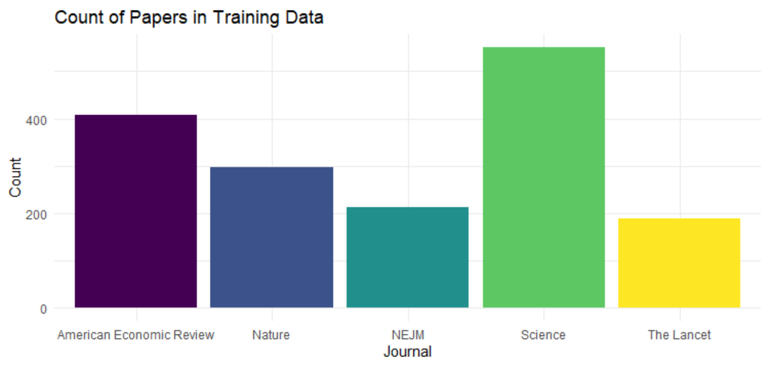
We immediately encountered a foreseeable problem with this methodology. Manual inspection of a paper typically yielded more than 10 examples of paragraphs discussing methods, but only 1 in 4 papers included a paragraph discussing curation and provenance. Therefore, we were much more likely to oversample from these over-represented categories. This led to a very unbalanced training set, created from 210 paragraphs in total, but only including 5 examples from Curation and Provenance, the most under-represented category.

To expand the training set, we used a novel data sampling method powered by tokenizing texts. Using the WordPiece tokenizer, the 25 examples in each category were converted to vectors. Combining text vectors has been shown to preserve semantic meaning in latent space [7]. Thus, the vectors were averaged to create an archetypical semantic vector for each category.

These archetype vectors were then compared to a much larger corpus of over 2000 papers, also from the five aforementioned journals. These papers were also randomly selected, but downloaded as entire issues rather than as individual files. Iterating through the paragraphs in this much larger corpus, the closest paragraphs to each archetype, using cosine similarity as a distance metric, were identified manually checked, and then added to the training set if the classification was correct. In this way, we were able to expand our training set to include 200 examples in each category, while maintaining balance across categories. The distribution of papers making up the training data can be seen in the below figure. In total, the paragraphs come from 1,657 different papers, across disciplines.

Once the training set had been created, it was necessary to download the data we sought to analyze. We decided to focus on *Science*, as it included articles from multiple disciplines, and was also the most represented journal in our training set. It would have been ideal to download the entire corpus of *Science* publications, but the sheer size of that dataset posed problems. Using a similar methodology as the sampling for the training set, 30 papers a year were downloaded from

Figure 3.0.1: Distribution of Papers across Journals in Training Data



issues of Science.

A number of problems arose during the mining of the dataset. The full code used for downloading and processing data can be found in Appendix B. It may be beneficial for future text miners to understand these issues to facilitate further research in this and other areas using these techniques.

Firstly, while JSTOR lists an API under their Labs Projects, this API has not been updated since early 2018 and appears to be deprecated at this time. An API would greatly facilitate meta-analysis at scale. Having to build a custom script to scrape data from JSTOR took up a substantial amount of time and is not feasible for all researchers. Even more importantly, while JSTOR uses a machine-readable and transparent storage system, the archives of individual journals tend to vary widely and are difficult for machine tools to interact with.

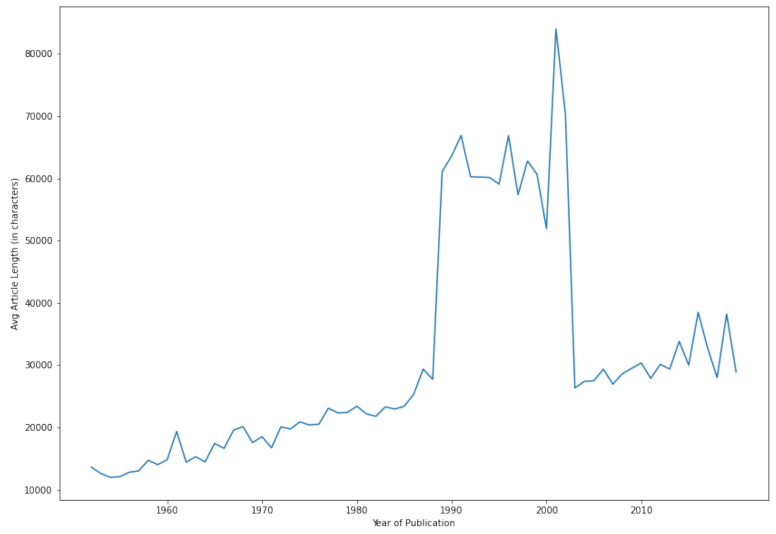
While these journals are clearly sensitive to security concerns, the benefits of improving the machine-readability of their archives, or even of building an API, would be enormous. Constructing a database that users can query, in addition to an API, would greatly facilitate meta-analysis, as well as simpler projects like visual essays building off of the archives. The *Science* archives, which store papers going back to the 19th century, are holding decades of knowledge: it is a shame that nobody is reading and analyzing this knowledge base and expanding access should be a top priority.

This is especially relevant for pre-training or fine-tuning large language models. These models, ever expanding, are hungry for text. BERT, for example, is trained on the BooksCorpus (800 M words) and English Wikipedia (2500 M words). Recent work has shown that at current rates of training expansion, we Academic archives could be a vital additional text source for high quality training data for future large language models, but the unwieldy nature of the archives makes this difficult [27].

There have been 6638 issues of Science published at the time of writing. Generalizing, we see in the data we've downloaded around an average of 10 research papers per issue, not including the expanded digital issues. The Science submission guidelines recommend submissions of 2,000 to 3,000 words: taking the lower bound at 2,000 words, this archive still contains 132 M additional words that could be used for high quality, accurate training for future language modelling.

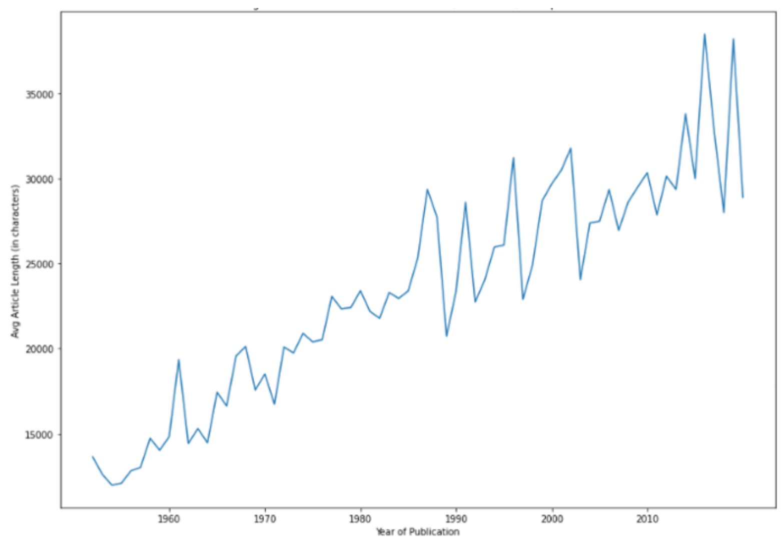
Another issue arose when reading the downloaded files into text. The methods used by Science to archive their articles have changed dramatically over the years. As seen in Figure 2, this had a large impact on the total amount of text read from PDFs into text files. Text was extracted from the files using the `pytesseract` package, a cutting edge python library for optical character recognition.

Figure 3.0.2: Average Article Length in Science Archives in Characters, 1950-2020, with initial OCR text extraction



When data is read in this way, we find an abnormal jump from 1987-2003. Rather than being due to a genuine shift in methodology, this is due to a change in the archival methods. From 1950, the start of data collection, to 1987, papers were stored as scans of the journal issue. In 1987, Science made the shift to a digital PDF format, where article text is packaged into as few pages as possible. This means that OCR read-in of these files incorporated lots of additional paragraphs from other papers included above and below the main text. In 2004, Science switches back to a scanned method, and in 2007, they move to a new digital upload format, where only the main text is included. For this reason, it is necessary to pre-process the data before it can be modelled. Data was pre-processed in two ways. First, **REFERENCES** was used as a cutoff string to remove paragraphs which appear after the main text. Second, all paragraphs were assigned to a set, meaning duplicated paragraphs were removed. In theory, extraneous paragraphs will appear more than once: in the

Figure 3.0.3: Average Article Length in Science Archives in Characters, 1950-2020, with pre-processed OCR text extraction



main text of their own paper, and in the extraneous paragraphs above other papers. This process cleared up the unusual pattern, but introduced some amount of volatility into the dataset. The pre-processed visualization of article length can be seen in Figure 3.

Interestingly, we see a linear increase in article length over time. This is consistent with changes in *Science* submission guidelines. Technical reports in the 1950s were limited to 2 pages; by 2020, this has expanded to a limit of 5 pages. It would be an interesting question to determine the nature of this relationship. It could be top-down, driven by changes in submission guidelines, but it could also be bottom-up, driven by authors' and readers' desire for more detail in their papers.

3.1 Data Confession

One of the principal objectives behind data minding is to encourage more complete data confession in journals. With that goal in mind, it is worth discussing some of the limitations in our own data. As previously discussed, we encountered a number of difficulties mining and extracting data, and, while some of these issues, like article length, have been addressed, it is possible that some of the variation in our data set is due to errors.

Firstly, it was not always possible to completely remove all extraneous article text, in particular from scanned documents. We see a large amount of variation in format, style, and layout, both within *Science* and across the other journals studied. While we made our best attempts to standardize all data used for training and for our analysis, it is likely that some extra paragraphs are included. Since these paragraphs come either at the beginning or end of other papers, this could lead to overcounting in the Data Collection and Visualization categories.

Second, our training data only contains papers from five journals. While these journals cover a wide variety of domains, from medicine, to social science, to natural sciences, it is likely that our results do not generalize across all domains, in particular if they were excluded from training data. This is a problem with data minding at scale generally. Different disciplines have different expectations and standards for data confession. For example, in social sciences, data curation and provenance will encompass describing the geneology of the large public dataset used, but in medicine, which often uses small samples from experiments conducted by physicians, there is a very different understanding of what data provenance means.

4 Methods

4.1 Fine-Tuning

For fine-tuning on this data, the cross entropy multiclass loss function was used, optimized using the AdamW method.

The cross entropy loss function is defined as follows: Let there be C classes and let y_i be a one-hot encoding of the true class (i.e., $y_i = 1$ if the true class is i and $y_i = 0$ otherwise). Let p_i be the predicted probability of class i. Then, the loss function is:

$$CE_{Loss} = -sum(y_i * log(p_i))$$

AdamW is an optimization algorithm for training neural networks that builds upon the popular Adam optimizer. It was proposed by Loshchilov and Hutter in 2017. AdamW is similar to Adam in that it uses a combination of momentum and adaptive learning rate to optimize the weights of the neural network. However, AdamW includes an additional weight decay term in the update rule that helps prevent overfitting of the model. This weight decay term is added directly to the gradients of the loss function during the optimization process.

The update rule for AdamW is as follows:

1. Compute the gradients of the loss function with respect to the model parameters.

2. Calculate the first and second moments of the gradients using the exponentially decaying moving average:

$$m_t = beta_1 \times m_{t-1} + (1 - beta_1) \times g_t$$
$$v_t = beta_2 \times v_{t-1} + (1 - beta_2) \times g_t^2$$

where g_t is the gradient of the loss function at time step t, m_t and v_t are the first and second moments of the gradients at time step t, and $beta1$ and $beta2$ are hyperparameters that control the decay rate of the moving averages.

3. Adjust the bias-corrected first and second moment estimates:

$$\hat{m}_t = m_t/(1 - beta_1^t)$$
$$\hat{v}_t = v_t/(1 - beta_2^t)$$

4. Update the model parameters:

$$w_t = w_{t-1} - \alpha * (\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon) + w_d * w_{t-1})$$

where alpha is the learning rate, epsilon is a small constant added to avoid division by zero, and w_d is the weight decay coefficient.

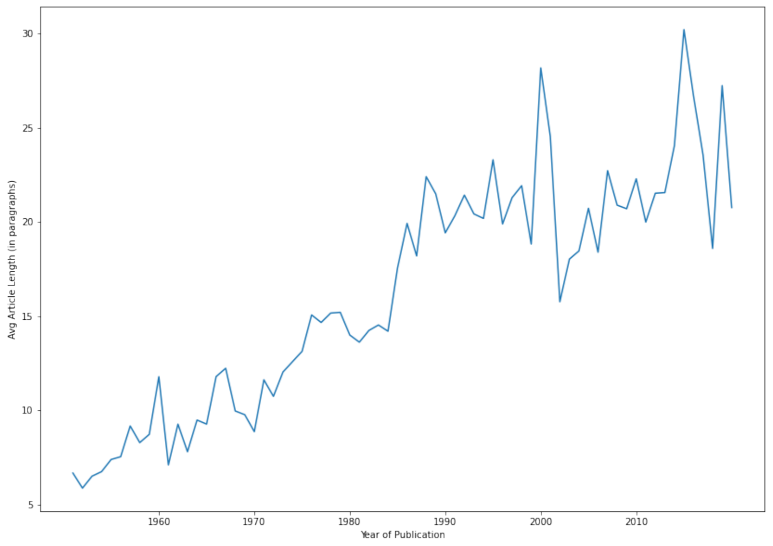
Papers were processed into discrete collections of paragraphs. The average number of paragraphs per year is displayed in the following figure.

As referenced in the Data section, additional pre-processing was done to ensure these lists of paragraphs reflect the actual text captured in the papers.

Once paragraphs were extracted, they were again filtered for length. Paragraphs consisting of under 200 tokens were dropped. Accuracy on these short paragraphs was very low, and manual inspection revealed that they typically consisted of references, section headers, captions, and other text not included in the main portion of the text.

On average, these short paragraphs made up around 6 percent of the total text in the articles, and this number was relatively constant

Figure 4.1.1: Average Article Length in Science Archives in Paragraphs, 1950-2020, with pre-processed OCR text extraction



across the years studied.

Once the paragraphs were suitably processed, they were tokenized using the BERT tokenizer. BERT uses a word-piece tokenizer which breaks larger words into portions which can be more easily matched to a dictionary. In addition, BERT uses positional encoding to represent the semantic relationships in the text, a much more powerful technique than standard NLP "bag-of-words" methods. This means that a BERT embedding is not just an array: it is a rank-2 matrix where each row represents a word in the input, and each column corresponds to that word's position in the 512-dimension BERT embedding space. These embedding matrices are also known as tensors.

As a simple example, we can look at a paragraph from our training data, labelled as **Data Conceptualization and Collection**, from

Stavroulaki et al (2018) [22].

Data on pedestrian movement were collected from Amsterdam, London and Stockholm via tracking anonymised Wi-Fi signals from mobile phones (Berghauser Pont et al., 2019; Stavroulaki et al., 2019). The pedestrian survey was done within a 3-week period in October 2017 and included 19 neighbourhoods in Stockholm, 18 in Amsterdam and 16 in London. The detection devices were positioned at every street crossing in the selected areas. [...] The data set included 219 street segments in Stockholm, 247 in Amsterdam and 212 in London.

There are 198 words in the non-truncated version of this paragraph. Therefore, the tensor produced by BERT tokenization will have 198 rows and 512 columns.

In certain cases, paragraphs in our dataset contained more than 512 words. These paragraphs were truncated during the tokenization process. In addition, to enable robust classification, paragraphs with fewer than 512 words had padding tokens injected. The final dataset used for training consisted of 2000 labelled tensors, all of which were 512x512.

Training was conducted using the `pytorch` and `transformers` package. Both these packages are invaluable for machine learning research. The `transformers` package includes a number of different pre-trained models, including BERT, GPT-J, and more. `Pytorch` features a robust API that allows for intuitive and flexible fine-tuning on a variety of tasks, including text classification. Because of the flexibility of this API, we were able to test a number of alternative models.

Using Pytorch, tensors were prepared for training and split into train, test, and validation sets, with a 0.5, 0.25, 0.25 split respectively.

There are several hyper-parameters required for fine-tuning a largel language model. The optimizer and loss function used have already been explained, but it is also vital to determine the optimal number of epochs for training.

An epoch refers to a complete pass through the training data during the training phase. In other words, an epoch is a measure of the number of times the model has seen each example in the training set during training.

During an epoch, the model iteratively adjusts its parameters to minimize the loss on the training data. After each iteration, the model updates its parameters based on the gradient of the loss function with respect to those parameters, using the AdamW optimization function.

If the number of epochs is too small, the model may not have enough training time to converge to an optimal solution [24]. On the other hand, if the number of epochs is too large, the model may start to overfit the training data and generalize poorly to new data. To determine the number of epochs required for this fine-tuned task, we iterated through options until our accuracy was maximized. Since our training set is representative of the task at large, and is sufficiently large, overfitting was not a concern except when the number of epochs is extremely large. The results of this experiment, as well as the total time to train on a GPU with 12 GB of RAM, can be seen in Appendix B. The optimal number of training epochs, as determined by this methodology, was 6.

After 6 epochs of training BERT, we achieved the results displayed in Table 1.

Epoch	Validation Loss	Accuracy
1	1.315018	0.498952
2	0.995708	0.616352
3	0.934290	0.664570
4	0.871753	0.691374
5	0.853981	0.727819
6	0.846249	0.744516

Table 4.1.1: Validation Loss and Accuracy for BERT Fine-Tuning

These results must be considered in context of the difficulty of the task. Training papers come from a variety of different journals and disciplines, and contain many hundreds of interchangeable tokens. In particular, there are many unique tokens indexed to specific sub-domains or projects: these can skew our results.

For example, if we break down our training data into our text categories, we can see that the NEJM and the Lancet make up almost all the examples of data curation and provenance in our dataset. There are a number of possible reasons for this: the simplest is that medical journals are much better at encouraging this specific type of data confession. However, when applied to journals which are not focused on medicine, the predictive power of our model is greatly weakened. In addition, our classifier is complex: we have 7 possible outcomes. If we restrict our results to only include predictions where the model predicts a given label with at least 60 percent confidence, we achieve a much greater level of accuracy, at the expense of greatly reducing the number of predictions we make. A detailed overview of the model and parameter selection process can be found in Appendix

Confidence Cutoff	Accuracy	Number of Predictions above Confidence Cutoff
0	0.74	1000
0.2	0.79	832
0.4	0.86	741
0.6	0.87	695
0.8	0.87	431

Table 4.1.2: Accuracy of BERT Fine-Tuned Model at different Confidence Cutoffs on the same Test Set

B.

Still, in contrast to classical NLP approaches, which were conducted using a combination of TF-IDF from the **sklearn** python package and traditional algorithms, we see that BERT fine-tuning far outperforms them on accuracy, which we would expect given the literature.

Model	Accuracy
BERT	0.744516
Naive Bayes	0.616352
Support Vector Machine	0.664570
Random Forest	0.664570
Linear Regression	0.664570

Table 4.1.3: Accuracy for BERT Fine-Tuning and classical text classifiers

4.2 Wavelet Denoising

Upon initial application of the fine-tuned BERT model on a new corpus of papers collected from *Science*, it was difficult to identify trends and turning points due to the high level of noise in our data.

In recent years, the wavelet transformation has become very popular in a variety of applications, including engineering, medicine, mathematics, and statistics [17] [2]. The wavelet builds on traditional denoising based on frequency like the Fourier transform for non-stationary data. We can think of our seven categories as waves, progressing through our time series. We see periodicity; the Fourier transform will thus perform poorly. In some research, denoising using the wavelet transformation has led to substantial improvements in forecasting time series, but here we are only interested in denoising for exploratory analysis [**?**]. The wavelet transform can be summarized in the following step-by-step process:

1. Perform a wavelet transform: A wavelet transform is applied to the time series data. This transform decomposes the time series into a set of coefficients, each representing a different frequency range of the signal.

2. Threshold the wavelet coefficients: The wavelet coefficients are thresholded to remove high-frequency noise. This means that any coefficient with a magnitude below a certain threshold value is set to zero.

3. Reconstruct the denoised signal: The thresholded wavelet coefficients are used to reconstruct a denoised version of the time series. This is done by performing an inverse wavelet transform on the thresholded coefficients.

We ran our classifier on papers collected from over 70 years of *Science*. We then used a Debauchies wavelet transform with a threshold of 2.2 for denoising. More details on the wavelet transformation, as well as the process of threshold and function selection, can be found in Appendix B.

5 Results

5.1 Trends in Science 1950-2020

Applying our classifier to our corpus of articles, and processing results into percentages by paper, we produce the following visualization.

While it is possible to read and understand this graph, it is much easier to interpret after denoising. Applying a Debauchies wavelet transform, we produce an alternate depiction of the above.

Figure 5.1.1: Percentage of Article classified as belonging to each category using a BERT fine-tuned model, 1950-2020

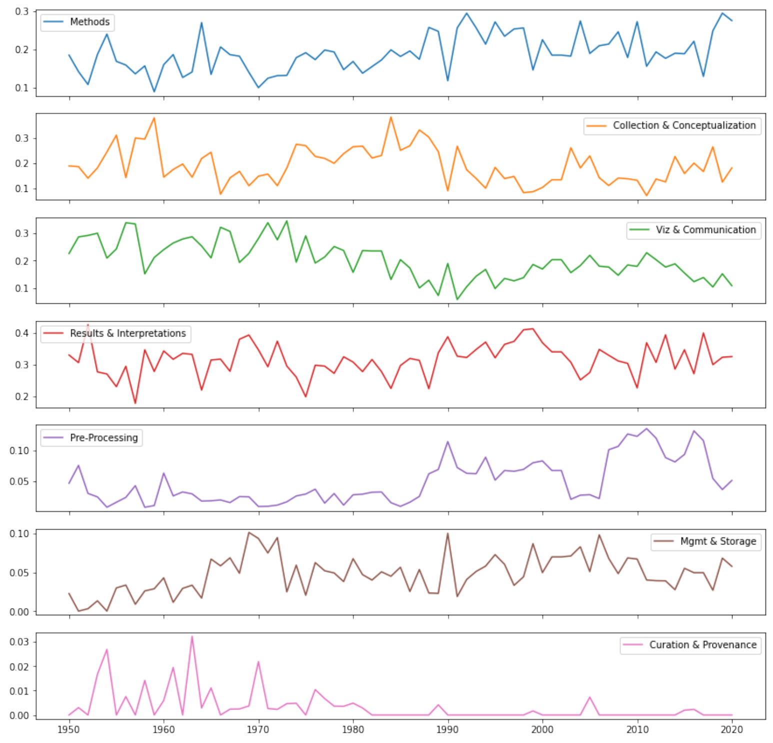


Figure 5.1.2: Percentage of Article classified as belonging to each category after DWB 2 Denoising, 1950-2020

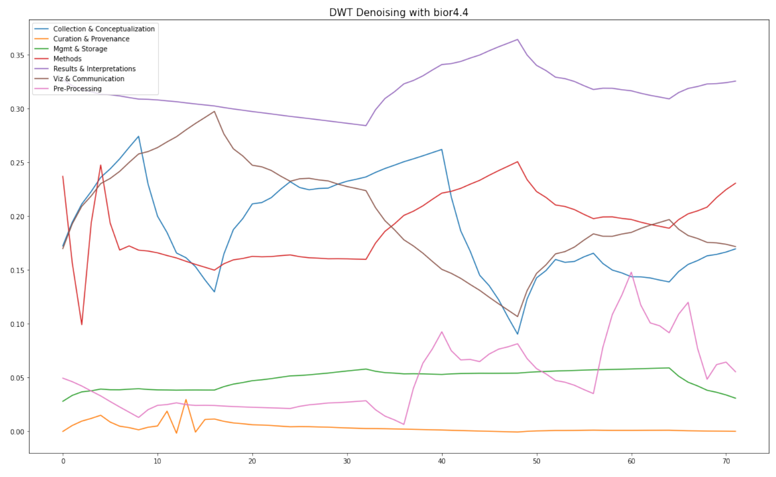
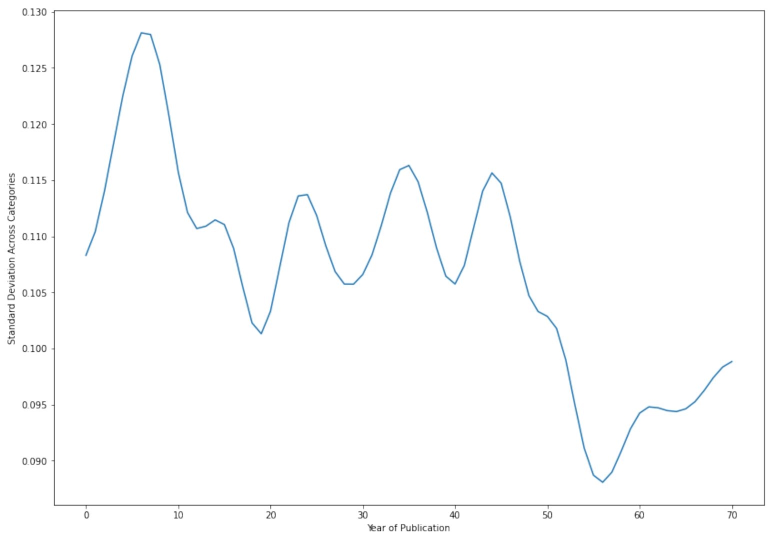


Figure 5.1.3: Standard Deviation of Category Classification (Methods excluded), 1950-2020



Finally, we can measure the volatility across the period using the standard deviation of the seven categories. Methods has been excluded from this calculation since its proportion of the articles remains relatively constant from 1950-2020.

5.1.1 Methods and Analysis

While we see substantial volatility in some categories, the proportion of papers dedicated to methods and analysis has remained relatively constant across the period studied. In the 1950s, authors spent around a third of their papers devoted to methods and analysis, and in 2020, that proportion has not changed. This is consistent both with expert analysis of selected data and with our intuition [13] [21]. Methods is the largest category, and this is true for the entire study period.

5.1.2 Results, Interpretations, and Visualizations

After methods, the highest proportion of papers is dedicated to results and interpretations and data Visualization. We see that there is a negative correlation between these two categories. This also makes some intuitive sense. With limited space in publications, authors must choose the most effective ways to demonstrate their results. Sometimes this will be through interpreting results in context, in text, and sometimes this will involve figures and other visualizations. In future work, it is worth considering whether these categories should be grouped together. While they have different methods, both interpretations and visualizations have the same goal: to communicate results to an audience. As quantitative tools have grown, it has become more common for visualization to take priority over results, but there are exceptions to every rule, and determining how best to communicate is very study-specific.

5.1.3 Pre-Processing and Management and Storage

As we might expect, we see a steady growth in the percentage of papers dedicated to these categories over time. Starting in the 1980s, a time when statistical tools became more popular [Ali 2016], we see that researchers have begun to devote more text to discussing their deliberations as to how to process and manage their data. Microsoft Excel was released in 1985, and other statistical tools, like R, Stata, and Python, were quick to follow. We see that the trend in pre-processing actually begins slightly before the release of these tools. This would suggest that, rather than the tools inspiring more paragraphs dedicated to pre-processing, this was a general trend in research that is correlated with, but not caused by, the development of new data technologies.

5.1.4 Data Collection, Conceptualization, and Curation

The growth in the aforementioned categories must come at the expense of space devoted to other categories. Instead of seeing an even decline in the other categories, we see that these two specific categories see a notable decline in the percentage of each paper devoted to their discussion in the time period studied. Data Curation in *Science* makes up a very small portion of papers, even at its peak: this also fits our intuition. *Science* is focused on natural sciences, like chemistry, physics, and biology; data curation has less meaning in these domains in regards to data quality. However, that's not to say it plays no role, and yet authors spend little to no text on curation.

Data conceptualization and collection, the first step in the data life cycle, is a bigger issue. This step affects all proceeding steps, including analysis. To reference the earlier adage, garbage in, garbage out. Even in the natural sciences, it is important to articulate in-depth decisions made collecting data for analysis. Unlike curation, we do see that most papers in our dataset do reference data collection, at least in a few paragraphs. However, the proportion of text dedicated to this category has seen a precipitous decline since the aforementioned statistical developments in the 1980s.

6 Conclusion

The rise of quantitative tools in scientific research has brought about significant advancements in our understanding of the world around us. From physics to biology, the ability to collect and analyze large amounts of data has enabled researchers to uncover new insights and make more accurate predictions. While there are still challenges to overcome, such as ensuring the validity of data and avoiding the pitfalls of over-reliance on statistical analysis, the benefits of quantitative tools are undeniable. As technology continues to evolve, we can expect to see even more sophisticated methods of data collection and analysis emerge. In this context, data minding is more important than ever. However, to enable accurate data minding, we must first promote greater data confession from authors.

This paper finds that as *Science* has evolved, from 1950-2020, advancements in technology are reflected in different makeup of papers. Unfortunately, this has not led to better data confession. The expansion of quantitative tools has led to papers focusing more on the steps in the data life cycle involving preparing data for analysis, including pre-processing and data management. As well, we see that data visualization, while not growing linearly, has also become more popular, perhaps also a reflection of increased tools for researchers to communicate their results more effectively. However, this has not come without a cost. Data collection and curation are two of the most vital steps in any research to measure and validate data quality. At a very basic level, if data is garbage going in, then no matter how much processing is done, the analysis will also be of low quality. For this reason, it is equally important to address these categories in papers. Unfortunately, we find that authors have tended to neglect these categories historically, and that the trend is worsening. We are hopeful that this meta-analysis will be persuasive to editors and reviewers, and will encourage a more complete data confession from authors going forward. Because of the feedback loops in the data life cycle, for data minders to truly understand the data quality in a given study, data collection and curation must be prioritized.

7

References

References

[1] Ali, Zulfiqar, and SBala Bhaskar. “Basic Statistical Tools in Research and Data Analysis.” Indian Journal of Anaesthesia, vol. 60, no. 9, 2016, p. 662. DOI.org (Crossref), https://doi.org/10.4103/0019-5049.190623.

[2] Alrumaih, Rumaih M., and Mohammad A. Al-Fawzan. “Time Series Forecasting Using Wavelet Denoising an Application to Saudi Stock Index.” Journal of King Saud University - Engineering Sciences, vol. 14, no. 2, 2002, pp. 221–33. DOI.org (Crossref), https://doi.org/10.1016/S1018-3639(18)30755-4.

[3] Beckman, Micaela F., et al. “A Computational Text Mining-Guided Meta-Analysis Approach to Identify Potential Xerostomia Drug Targets.” Journal of Clinical Medicine, vol. 11, no. 5, Mar. 2022, p. 1442. DOI.org (Crossref), https://doi.org/10.3390/jcm11051442.

[4] Berry, Michael W., et al., editors. Survey of Text Mining II: Clustering, Classification, and Retrieval. Springer, 2008.

[5] Bommasani, Rishi, et al. On the Opportunities and Risks of

Foundation Models. 2021. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2108.07258.

[6] Brown, Tom B., et al. Language Models Are Few-Shot Learners. 2020. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2005.14165.

[7] Coecke, Bob, et al. Mathematical Foundations for a Compositional Distributional Model of Meaning. 2010. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1003.4394.

[8] Devlin, Jacob, et al. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. 2018. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1810.04805.

[9] Ford, Elizabeth, et al. "Toward an Ethical Framework for the Text Mining of Social Media for Health Research: A Systematic Review." Frontiers in Digital Health, vol. 2, Jan. 2021, p. 592237. DOI.org (Crossref), https://doi.org/10.3389/fdgth.2020.592237.

[10] Gao, Leo, et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. 2021. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2101.00027.

[11] Hung, Jui-Long, and Ke Zhang. "Examining Mobile Learning Trends 2003–2008: A Categorical Meta-Trend Analysis Using Text Mining Techniques." Journal of Computing in Higher Education, vol. 24, no. 1, Apr. 2012, pp. 1–17. DOI.org (Crossref), https://doi.org/10.1007/s12528-011-9044-9.

[12] Meng, Xiao-Li. "Statistical Paradises and Paradoxes in Big Data (I): Law of Large Populations, Big Data Paradox, and the 2016 US Presidential Election." The Annals of Applied Statistics, vol. 12, no. 2, June 2018. DOI.org (Crossref), https://doi.org/10.1214/18-AOAS1161SF.

[13] Meng, Xiao-Li. “Enhancing (Publications on) Data Quality: Deeper Data Minding and Fuller Data Confession.” Journal of the Royal Statistical Society Series A: Statistics in Society, vol. 184, no. 4, Oct. 2021, pp. 1161–75. DOI.org (Crossref), https://doi.org/10.1111/rssa.12762.

[14] Minaee, Shervin, et al. “Deep Learning–Based Text Classification: A Comprehensive Review.” ACM Computing Surveys, vol. 54, no. 3, Apr. 2022, pp. 1–40. DOI.org (Crossref), https://doi.org/10.1145/3439726.

[15] Luo, Xiaoyu. “Efficient English Text Classification Using Selected Machine Learning Techniques.” Alexandria Engineering Journal, vol. 60, no. 3, June 2021, pp. 3401–09. DOI.org (Crossref), https://doi.org/10.1016/j.aej.2021.02.009.

[16] O’Mara-Eves, Alison, et al. “Using Text Mining for Study Identification in Systematic Reviews: A Systematic Review of Current Approaches.” Systematic Reviews, vol. 4, no. 1, Dec. 2015, p. 5. DOI.org (Crossref), https://doi.org/10.1186/2046-4053-4-5.

[17] Oster, Robert A. “An Examination of Statistical Software Packages for Categorical Data Analysis Using Exact Methods—Part II.” The American Statistician, vol. 57, no. 3, Aug. 2003, pp. 201–13. DOI.org (Crossref), https://doi.org/10.1198/0003130031928.

[18] Paynter R, Bañez LL, Berliner E, et al. EPC Methods: An Exploration of the Use of Text-Mining Software in Systematic Reviews [Internet]. Rockville (MD): Agency for Healthcare Research and Quality (US); 2016 Apr. Results. Available from: https://www.ncbi.nlm.nih.gov/books/NBK362048/

[19] Pedder, Hugo, et al. “Data Extraction for Complex Meta-Analysis (DECiMAL) Guide.” Systematic Reviews, vol. 5, no. 1, Dec. 2016, p. 212. DOI.org (Crossref), https://doi.org/10.1186/s13643-016-0368-4.

[20] Pham, Ba’, et al. “Text Mining to Support Abstract Screening for Knowledge Syntheses: A Semi-Automated Workflow.” Systematic Reviews, vol. 10, no. 1, Dec. 2021, p. 156. DOI.org (Crossref), https://doi.org/10.1186/s13643-021-01700-x.

[21] Shamseer, L., et al. “Preferred Reporting Items for Systematic Review and Meta-Analysis Protocols (PRISMA-P) 2015: Elaboration and Explanation.” BMJ, vol. 349, no. jan02 1, Jan. 2015, pp. g7647–g7647. DOI.org (Crossref), https://doi.org/10.1136/bmj.g7647.

[22] Stavroulaki, Gianna, et al. Methodology and Results of an International Observational Study on Pedestrian Movement Tracking Anonymised Wi-Fi Signals from Mobile Phones. 2018. DOI.org (Datacite), https://doi.org/10.13140/RG.2.2.21825.20321.

[23] Silver, David, et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search.” Nature, vol. 529, no. 7587, Jan. 2016, pp. 484–89. DOI.org (Crossref), https://doi.org/10.1038/nature16961.

[24] Sun, Chi, et al. How to Fine-Tune BERT for Text Classification? 2019. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1905.05583.

[25] Tickle, S. O., et al. “A Computationally Efficient, High-Dimensional Multiple Changepoint Procedure with Application to Global Terrorism Incidence.” Journal of the Royal Statistical Society Series A: Statistics in Society, vol. 184, no. 4,

Oct. 2021, pp. 1303–25. DOI.org (Crossref), https://doi.org/10.1111/rssa.12695.

[26] Touvron, Hugo, et al. LLaMA: Open and Efficient Foundation Language Models. 2023. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2302.13971.

[27] Vaswani, Ashish, et al. Attention Is All You Need. 2017. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1706.03762.

[28] Wing, Jeannette M. "The Data Life Cycle." Harvard Data Science Review, June 2019. DOI.org (Crossref), https://doi.org/10.1162/99608f92.e26845b4.

[29] Zaheer, Manzil, et al. Big Bird: Transformers for Longer Sequences. 2020. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2007.14062.

8 Appendices

8.1 Appendix A: Few-shot Learning

While the mechanics of few-shot learning have been briefly discussed, they are worth mentioning in detail. Few-shot learning is one of the most promising emergent powers of language models, involving training a model to learn new concepts or tasks with only a few examples, sometimes as few as one. This is in contrast to traditional machine learning, which typically requires thousands or even millions of examples to achieve high accuracy. The majority of modern AI tools, like ChatGPT, are the product of two parallel developments in artificial intelligence: few-shot learning and reinforcement learning with human feedback. Few-shot learning, which can also be thought of as a type of prompt engineering, is used to separate Chat and other models from the base models without altering the model parameters or weights. Instead, a simple prompt outlining the model's goals and instructions is provided before the model ever encounters a user in the wild [6]. It is interesting that few-shot learning, while it has led to notable advancements in a number of natural language processing tasks, is an emergent property of these models. While we know it works, researchers are still trying to determine why this method is so successful.

Reinforcement learning with human feedback uses a type of adversarial learning to improve a model's ability to provide a certain type of response to input. The methodology is simple: model output is scored by human auditors on a series of metrics. These metrics are then converted into a reward function, and the two models are trained together, similar to training AIs in an adversarial network, or GAN, which is similar to the techniques used in a number of seminal models, like AlphaGo, which can play complex games developed by humans at a level higher than any human [23]. The initial language model, rather than simply try and optimize next-token probabilities, is instead used to optimize the reward function.

Taken in tandem, these twin training techniques have led to rapid development in how models interact with users, and have greatly reduced hallucinations, where the model generates false text, as well as reduced model capacity for harmful or malicious output. While they are not perfect, as jailbreakers and hackers across the internet can attest, a simple change to the base prompt (changing the ”one” shot) leads to a tremendous difference in model output. We were interested in the capacity for large language models to develop text classification skills using few-shot learning, rather than needing to fine-tune base models, which costs time and computation and requires building an extensive training set.

We looked at 3 large language models, all of which have been identified by their authors as having the potential for few-shot learning: BERT, GPT-3, and GPT-J. These models were fed a detailed prompt, with several examples of text classification. Then, another 30 examples were provided to each model, and accuracy and precision were assessed. We would have liked to test this methodology on a larger test set, but were hampered due to rate limitations, as well as limited access to some of these models. Future work would do well to include LLAMA, an open-source model that can run on your own PC, rather than interact with a model API, as we did. For clarity, the full prompt used was as follows:

You are a model tasked with classifying a paragraph of text into one of the following seven categories: Data Collection, Pre-Processing, Data Management and Storage, Data Curation and Provenance, Methods, Visualizations, Interpretations. Below are some examples of labelled paragraphs. You will be prompted with a text, followed by an —END OF TEXT— token. Respond only with "Label: your best guess.

Text: "In this study, we used a combination of pre-processing techniques to prepare our dataset for machine learning analysis. First, we performed data cleaning to remove any missing or erroneous data points. Next, we normalized the data using z-score normalization to ensure that each feature had a mean of 0 and standard deviation of 1. We also performed feature selection to identify the most relevant features for our analysis, using a combination of correlation analysis and mutual information-based feature selection. Finally, we performed data augmentation to increase the size of our dataset, which involved generating synthetic data points using a generative model. Overall, these pre-processing techniques helped to improve the quality of our dataset and reduce the risk of overfitting in our machine learning models."—END OF TEXT—

Label: Pre-Processing

Text: "GAP- DRG (General Approach for Patient- oriented Outpatient- based Diagnoses- Related Groups) is a research database with reimbursement data for outpatient services of sickness funds and data on hospital stays provided by the Austrian Federal Ministry of Health from the years 2006 and 2007. Data from insurance institutions containing a Unique Person Identifier (UPI) were linked to data from inpatient sector (no UPI) through probabilistic record linkage. The reimbursement data for outpatient services include basic demographic descriptors and information on all drug prescriptions filled in the years 2006 and 2007 for all people with health insurance in Austria. Drugs are classified by Anatomical Therapeutic Chemical (ATC) groups, according to the WHO Collaborating Centre for Drug Statistics Methodology (2018). Hospital data comprise date of discharge, length of stay, and main and associated diagnoses, which are coded by the International Statistical Classification of Diseases and Related Health Problems (ICD- 10)." —END OF TEXT—

Label: Data Curation and Provenance

Text: "A possible interpretation also consistent with our results is that neutralization as a biological function mediated a large proportion of the vaccine efficacy, but the specific day 29 ID50 and ID80 immune markers studied—measured with a particular immunoassay—had inadequate sensitivity to quantify low-level neutralization below the positivity cutoff that could be present and functionally important.” —END OF TEXT—

Label: Results and Interpretations

The results of this process are displayed in the following table. Because of the limited size of our test set, precision and recall were not evaluated using this process.

Model	Accuracy
BERT	0.46
GPT-3	0.56
GPT-j	0.56

Table 8.1.1: Model Accuracy and Precision on Text Classification with Few-Shot Learning

These results are surprising. The most state-of-the-art analysis seems to indicate that for most tasks, few-shot learning is sufficient to improve model accuracy to a reasonable level [6], but we did not find that to be the case. The more advanced models, GPT-3 and GPT-J, did outperform BERT, which suggests that there is improvement in how much few-shot learning can help; but the results are far worse than our fine-tuned models. It is outside the scope of this paper to investigate why this is, but I have two possible theories. The first is that the model consistently started to break down after a few guesses and needed to be reminded of the format in which it was supposed to answer. BERT was the most guilty of this: after a few correct labels, it would often revert to generating random text within a category or hallucinating labels and categories that did not exist. There may be some obstacle here due to the fact we are interacting with these models using an API; their capacity to remember instructions may be weaker than the base models. Again, future testing on truly open-source models, like LLAMA, could prevent this possibility.

Second, our specific task is very unique. One of the main reasons few-shot learning is effective is that these models have a huge pre-trained corpus to draw text from. For a simple chatbot, or a question and answer program, or even simple sentiment classification, it is quite likely that some examples of this exist in the training data. This principle is called transfer learning, and refers to the model's ability to identify patterns between the new prompts and its existing capacity. However, as we know, there are no attempts to conduct data minding at scale in the literature, and thus, no data in the training corpus. In fact, if prompted, while these models can make educated guesses as to the nature of data minding, none are familiar with the exact definition. Therefore, it is difficult for transfer learning to occur and the model cannot be as effective with only a few prompted examples to learn from.

8.2 Appendix B: Pre-Trained Models and Denoising

This appendix will focus on figures and exploratory analysis, discussing how we decided on which pre-trained model to use for classification, how to set parameters for that model, and how to set parameters for wavelet denoising.

The figures will tell a story of trial and error to obtain useful results; interspersed with text we hope they will be useful for any modelers interested in exploring transformers for their particular natural language task. We'll begin with model selection. The `transformers` library contains dozens of pre-trained models for use in analysis, fine-tuning, or inference. We selected five possible models for inclusion, based on their size, predicted power, and documentation. Some of the best models, like GPT-3, have kept descriptions of their model architecture out of the literature. For this reason, it is difficult to use them in academic research, as it requires taking a substantial leap of faith. While most model architectures are similar, originating from the seminal BERT whitepaper, variation between models is not transparently explained [27] [8]. The five models we chose for testing were: GPT-J, GPT-2, BERT, and BigBird. All of these models vary along a number of axes.

The following tables and figures will explore the differences between these models.

Model	Training Data	Training Data Size (in words)
GPT-2	WebText (custom script)	45 M
GPT-J	ThePile [10]	825 M
BERT	Books Corpus + English Wikipedia	3.3 B
BigBird	"C4" (Colossal Clean Crawled Corpus)	138 B

Table 8.2.1: Training Data across selected Models

While the larger models were marginally superior to BERT, at least over one epoch, they were substantially slower to train and run, and took up much more space in the cloud. There are some advantages: for one, improved accuracy, but, notably, BigBird is also capable of

Figure 8.2.1: Selected Models: Training Time for one epoch using a 12 GB GPU

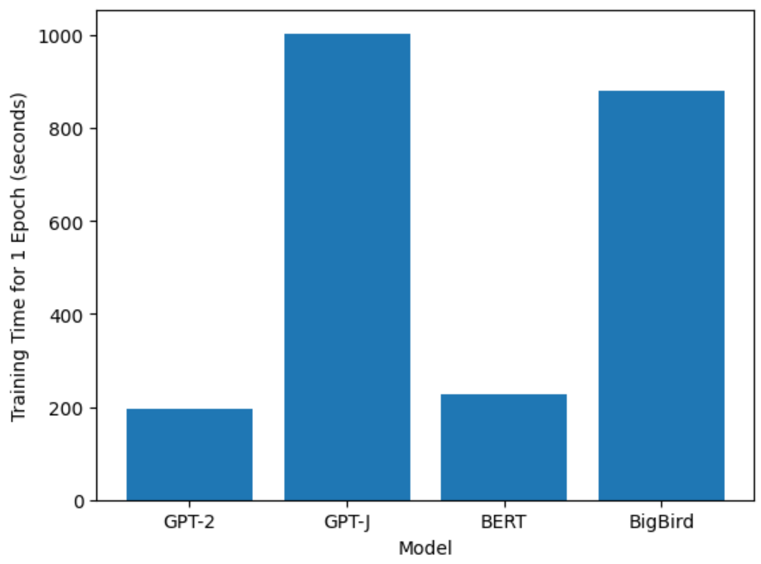
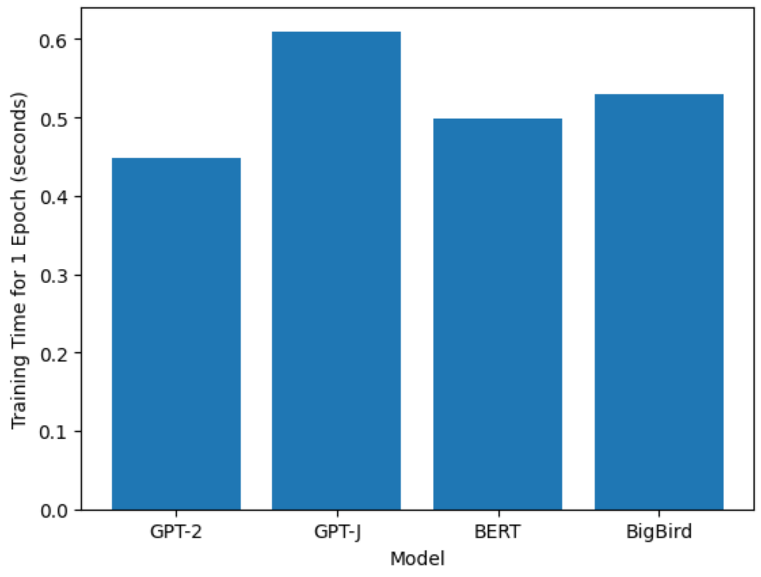


Figure 8.2.2: Selected Models: Accuracy after one epoch



handling much longer strings of text for inference than the alternatives. Since lots of our training and review data exceeded the BERT token limit of 512, this would have meant we could include the full paragraphs, rather than truncating them. However, we found that BigBird performed very poorly on precision and recall when evaluated on the whole set, especially in the more limited categories. For this reason, the decision was made to move forward with BERT as the primary model used in our review. Another main advantage of BERT was its excellent documentation, as well as deep contributions from literature that enabled optimal parameter selection and fine-tuning.

Category	Recall)
Curation	0.34
Pre-Processing	0.42
Management and Storage	0.49
Methods and Analysis	0.74

Table 8.2.2: BigBird Precision and Recall across selected Categories after five training epochs

Once we had selected BERT as our model, it was simple to choose a loss function and optimizer based on the literature [24]. The last remaining task was selecting the number of epochs to train our model. While the model training was quite fast, taking only 3-4 minutes per epoch, and our training set was sufficiently large and diverse that overfitting was not too much of a concern, this was still not an easy problem. We tested results empirically until our accuracy converged. On average, across 10 tests, this took six epochs, and so that was the number chosen for the final analysis.

8.2.1 Denoising

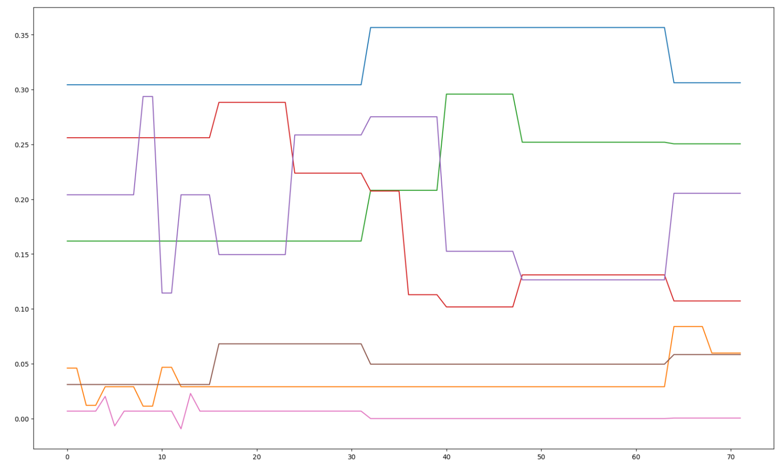
The complete process of wavelet denoising is not within the scope of this paper, but it was a tool we used for clearer visualization. It merits a brief discussion as to the mechanics of the wavelet transform, as well as our process in selecting both a wavelet function and a threshold.

The wavelet transform is a mathematical technique used for analyzing signals from noisy data, particularly in time series. It is a close cousin of the Fourier transform, but, unlike the Fourier transform, which decomposes a signal into its frequency components, the wavelet transform decomposes a signal into a set of wavelets, which are small functions that are localized in both time and frequency.

The wavelet transform is based on a family of wavelet functions, which are functions that can be scaled and translated to analyze different features of data. By applying the wavelet transform to a noisy signal, one can identify features such as edges, spikes, and trends, as well as their location and strength.

There are many different functions in this family. Conducting a literature review led us to the Daubechies wavelet, based on the work of Ingrid Daubechies. There are sub-variations of this function, with the number in the name indicating the number of vanishing moments in the wavelet. The Debauchies wavelet is preferred for signal denoising because it can analyze both smooth and jagged signals, and accomodates both sharp and smooth transitions. This was an apt description of our data, and so we used the debauchies wavelet with 2 vanishing moments.

Figure 8.2.3: Wavelet Denoising with Biorthogonal Wavelet



Orthogonality is one of the most important reasons the Daubecheies wavelet is useful for denoising. When using an orthogonal wavelet basis for denoising, the basis functions are mutually orthogonal, meaning they do not overlap and do not interfere with each other. Since the noise in a signal is often random and uncorrelated with the signal itself, it tends to be spread out evenly across all wavelet coefficients. This means that in an orthogonal wavelet basis, the noise is effectively spread out over all basis functions, while the signal is concentrated in a smaller subset of basis functions. By thresholding out the coefficients corresponding to the noise-dominated basis functions, we can remove the noise from the signal without significantly affecting the signal itself. The following figures show the outcome of denoising using other wavelet functions. These are also informative as to the signal in our data, but the Debauchies allowed for a clearer visual interpretation. All of these were set with 2 vanishing moments and a calculated optimal threshold using the

Figure 8.2.4: Wavelet Denoising with Haar Wavelet

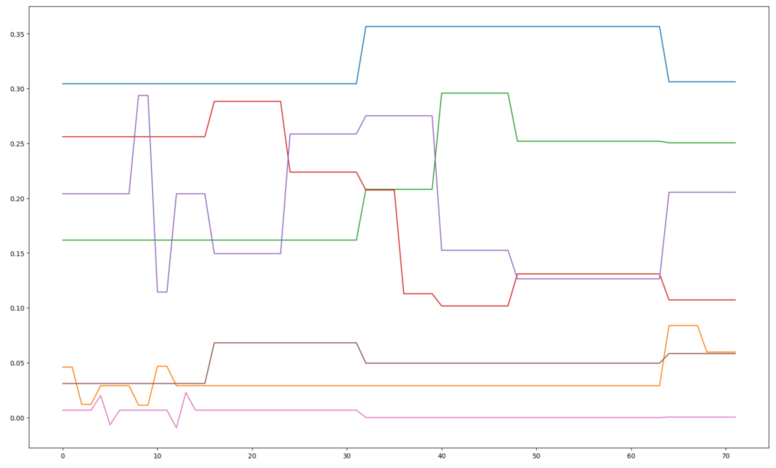


Figure 8.2.5: Wavelet Denoising with Symlet

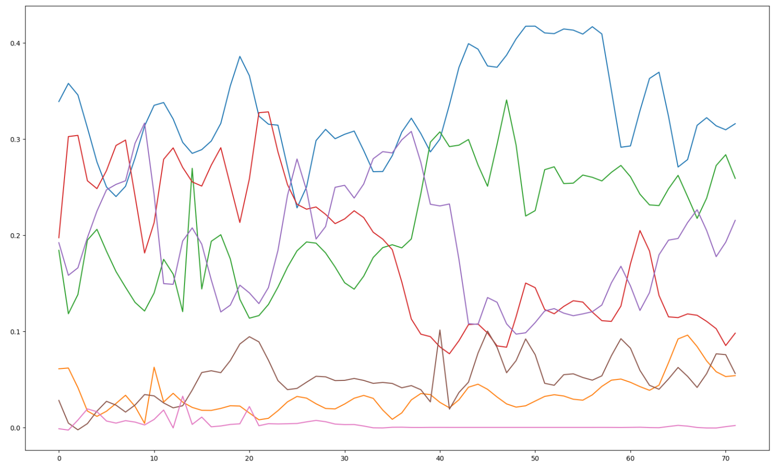
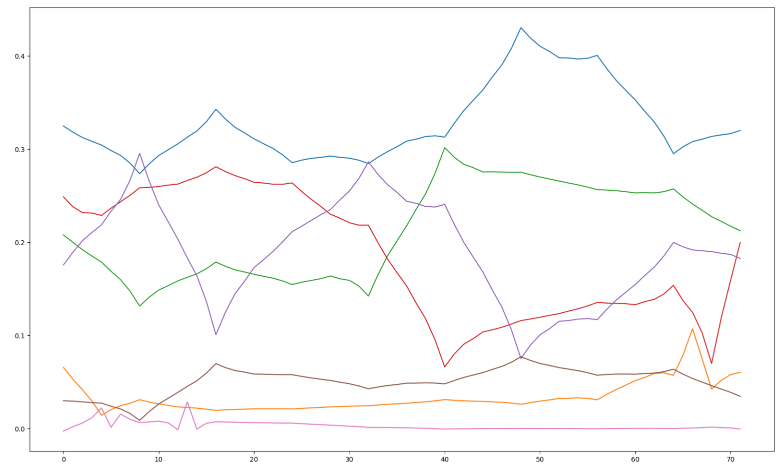


Figure 8.2.6: Wavelet Denoising with Coiflet Wavelet



`pywavelet` library.

8.3 Appendix C: Text Mining from JSTOR

For reproducibility, the scripts used to mine and process text data from JSTOR are included below. The first script uses `requests` to download papers from JStor. Users must provide their own authentication for this script to work, as well as complete capchas periodically or insert waiting periods in the script. One interesting application of large language models, especially multi-modal models, is they can complete capchas. The second uses `pytesseract` and `textract` to obtain text from these downloaded files. The two scripts are written as functions, which can be looped for multiple files.

Download Script

```
import requests
import Selenium
from requests_oauthlib import OAuth1Session
```

```
def main(url, query, button_xpath):

# complete authentication using requests OAUTH
    time.sleep(1)
    user = query
    response = oauth.get(f'{url}')

    # Check to see if response has succesfully pulled up webpage
    if response.status_code != 200:
        raise Exception(response.json())

    # use Selenium to click download button
    #set chromodriver.exe path
    driver = webdriver.Chrome(executable_path="C:\chromedriver.exe")
    driver.implicitly_wait(0.5)
    #launch URL
    driver.get(response)
    #identify element
    l =driver.find_element_by_xpath(f'{button_xpath}')
    #perform click
    l.click()
    print("Page_title_is:_")
    driver.quit()
    return 200
```

OCR Script

```
# Requires Python 3.6 or higher due to f-strings

# Import libraries
import platform
from tempfile import TemporaryDirectory
from pathlib import Path
import os
import pytesseract
from pdf2image import convert_from_path
from PIL import Image

if platform.system() == "Windows":

        pytesseract.pytesseract.tesseract_cmd = (
                r"C:\Program_Files\Tesseract-OCR\tesseract.exe"
        )

        path_to_poppler_exe = Path(r"C:\Program_Files\poppler-0.68.0\bin")


def main(text_file, PDF_file):
        ''' Main execution point of the program'''

    # Declare empty list to hold image files
        image_file_list = []

        with TemporaryDirectory() as tempdir:
                # Create a temporary directory to hold our temporary images.

                """
                Part #1 : Converting PDF to images
                """
                if platform.system() == "Windows":
                        pdf_pages = convert_from_path(
                                PDF_file, 500, poppler_path=path_to_poppler_exe
                        )
                else:
                        pdf_pages = convert_from_path(PDF_file, 500)
                # Read in the PDF file at 500 dots per inch. Resolution can be adjusted
        # if the program is slow. Each page will be stored as a single image.

                # Iterate through all the pages
                for page_enumeration, page in enumerate(pdf_pages, start=1):

                        # Create a temporray file to hold our image
                        filename = f"{tempdir}\page_{page_enumeration:03}.jpg"

                        # Declaring filename for each page of PDF as JPG
                        # For each page, filename will be:
                        # PDF page 1 -> page_001.jpg
                        # PDF page 2 -> page_002.jpg
                        # PDF page 3 -> page_003.jpg
                        # ....
                        # PDF page n -> page_00n.jpg

                        # Save the image of the page in our temp directory
                        page.save(filename, "JPEG")
                        image_file_list.append(filename)

                """
                Part #2 - Recognizing text from the images using OCR
                """

                with open(text_file, "a") as output_file:
                        # Open the file in append mode,
            # so that each page's contents will be collated together

                        # Iterate through the pages
                        for image_file in image_file_list:

                                # Read in the text in the image using pytesseract
                                text = str(((pytesseract.image_to_string(Image.open(image_file)))))

                                # The recognized text is stored as a variable
                                # Here, basic formatting has been done to remove line breaks
                # Any additional formatting can be done here
                                text = text.replace("-\n", "")

                                # Finally, write the processed text to the file.
                                output_file.write(text)


        # End of main function!


if __name__ == "__main__":
        # We only want to run this if it's directly executed!
        main(PDF_file = Path(f"{PATH_TO_YOUR_PDF}"), text_file = PATH(f"{PATH_TO_YOUR_OUTPUT_DIRECTORY}"))
```