Natural Language Sequence analysis for Automatic Keyword Assignment

An Application in Scientific Publications

School of Computer Science & Applied Mathematics
University of the Witwatersrand

Pieter Breuning
295821

Supervised by Prof Turgay Celik

December 27, 2021

Ethics Clearance Number: N/A

A research report submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science by coursework and research report

Abstract

This study aims to investigate the task of keyword assignment in text documents, which is a sparse multi-label classification of a discrete input over a large set of target variables. Most existing approaches focus on keyword extraction, which cannot produce descriptive keywords that do not appear in the text. Domain-specific word embeddings in conjunction with neural classifiers have proven to be effective for the task of keyword assignment in the literature. This work takes a similar approach, using word embeddings from pre-trained transformer models that are able to capture context deeply and bi-directionally. Two language models are used to produce such word embeddings *i.e.* BERT and OpenAI GPT, such that both bi-directional and left-to-right context models can be compared. As a basis for comparison, bag-of-words representations as well as pre-trained Word2Vec language models are also implemented. The datasets used for training and inference consist of publicly-available abstracts from scientific journals. Separate datasets are collected and used for pre-training of the language models, and for training the classifiers on the tasks of both domain prediction and keyword prediction (assignment). Several language model variations are tested to evaluate the impact of pre-training, and several classification architectures are tested in search of best performance. The results show little differentiation in model performance for the simpler task of domain prediction, but significant differentiation in the complex task of keyword assignment. It is found that the transformer models produce the best results in this complex task, and that the results are thematically coherent with respect to input texts. It is further found that performance is improved by pre-training transformer models using domain-specific text corpora.

Declaration

I, Pieter Breuning, hereby declare the contents of this research report to be my own work. This report is submitted for the degree of Master of Science in Computer Science by coursework and research report at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Condensing text documents into keywords that accurately represent and summarise the document at a thematic level is a fundamental requirement for tasks such as archiving, indexing, classification, novelty detection, summarization, information retrieval, and similarity ranking [Zhang and Xu 2009; Blei *et al.* 2003]. The task of keyword assignment is crucial to enable readers to find publications relevant to their needs, and for content recommendation engines to recommend the same. In the context of scientific publications, automation of accurate keyword assignment is of particular interest to publishers because it enables analysis of research trends over time, recommending of keyword tags, and highlighting missing citations to authors [Mahata *et al.* 2018]. There are two distinct approaches to this task i.e. keyword extraction and keyword assignment. Keyword extraction is the process of selecting thematically descriptive words or phrases that appear within the text, whereas keyword assignment includes "implicit" keywords that do not appear in the text [Hu *et al.* 2018; Schluter 2015; Zhang and Xu 2009; Beliga 2014; Singhal *et al.* 2017].

Keyword extraction has been extensively studied and there is an abundance of algorithms, both supervised and unsupervised, for analysing texts and extracting the most representative words and phrases. Early approaches used statistical methods to evaluate documents in terms of word frequencies and co-occurrence (n-gram statistics, term frequency-inverse document frequency [TF-IDF], Latent Semantic Indexing [LSI], Latent Dirichlet Allocation [LDA], among others) [Radford 2018; Onan *et al.* 2016]. One major advantage of statistical methods is that they typically do not require labelled corpora but suffer from the inherent limitation that important thematic keywords may appear in a text relatively few times and may therefore be filtered out from consideration [Hu *et al.* 2018; Beliga 2014]. Using statistical methods for text classification can also be improved when coupled with machine learning approaches such as ensemble classifiers [Onan *et al.* 2016]. Graph-based text representations have proven to be highly effective in keyword extraction. Documents are modelled as graphs in which vertices represent words and edges represent relationships between words, where these relationships can be established on the basis of co-occurrence, syntactic-, or semantic relationships [Hu *et al.* 2018; Beliga 2014; Ying *et al.* 2017]. In the domain of scientific publications, Behrouzi *et al.* [2020] used link prediction techniques in keyword networks for predictive modelling of research topic trends over time.

Fundamentally, keyword extraction algorithms cannot assign keywords that do not

appear in the text. There are many instances in scientific publication where mapping texts to implicit keywords is desirable for classification. Authors do not always explicitly mention the fields and disciplines to which their works belong, particularly not in the abstract, which is the publicly available component and therefore most appropriate for automatic indexing. To tackle this keyword assignment task, supervised learning is typically required to learn mappings from text to controlled vocabularies over a particular domain [Beliga 2014; Onan *et al.* 2016]. In general, keyword assignment is a difficult task to solve, requiring an abundance of high-quality labelled data that adequately represents the domain in which the keyword assignment algorithm must operate [Beliga 2014; Ramasubramanian *et al.* 2020]. There are often very many different keywords spanning any given domain and therefore target vectors are sparse, making learning a challenging task [Hu *et al.* 2018; Beliga 2014].

This study focuses primarily on the keyword assignment task across various thematically distinct domains in scientific literature. The aim is to learn mappings from text to sets of keyword phrases that form part of a controlled vocabulary. Specifically, word embeddings obtained using different modelling strategies (bag-of-words, Word2Vec, and transformer models) are used with custom multi-class neural classifiers and applied to the abstracts of scientific publications over six distinct domains, with the aim of accurately assigning the correct sets of keywords from a controlled vocabulary. The task of predicting the journal from which a text originates is also studied, with the aim of establishing a basis for comparison with respect to the complexity of the classification task. The contributions of this study are as follows:

- Development of custom keyword assignment pipelines using both generalised and domain-specific word embeddings, coupled with neural classifiers.
- Comparison of bag-of-words, Word2Vec, and transformer-based models as feature extraction techniques in the keyword assignment pipeline.
- Evaluation of the impact on classifier performance, of context sensitivity (left-to-right or bi-directional) of the transformer-based word embedding models.
- Evaluation of the impact on classifier performance, of additional domain-specific pre-training of word embedding models.
- Evaluation of classifier performance over a range of hyper-parameter *i.e.* number of hidden layers, and number of neurons per layer.
- Evaluation of the impact on classifier performance, of experimental design choices *i.e.* optimisation strategy, data splitting/selection strategy, and target variable dimensionality reduction.
- Evaluation of the impact on classifier performance, of the complexity of the classification task.

The rest of this document is laid out as follows: **Chapter 2** presents an overview of previous works in this field, **Chapter 3** presents details of the data and models investigated, **Chapter 4** presents the research methodology employed, **Chapter 5** presents the results obtained and the analysis thereof, and **Chapter 6** concludes the document

with closing commentary. Each chapter is furnished with a conclusion to summarise the main insights from the chapter. It must also be noted that the terms **"keyword assignment"** and **"keyword prediction"** are used interchangeably in this document; the terms **"journal prediction"** and **"domain prediction"** are also used interchangeably.

Chapter 2

Background and Related Work

The keyword assignment task faces challenges including varying document length, structural inconsistency, and inconsistency in semantic and syntactic structuring [Mahata *et al.* 2018]. This last challenge may arise from variations in the skill level of document authors, both in their fields and in the language. There is an inherent need for significant amounts of accurately labelled data to sufficiently learn the underlying thematic distributions; the quality of such data is subject to the skill of labelers, control over the labelling process, etc. [Vanyushkin and Graschenko 2020]. Developing strategies that perform well in different domains also requires consideration of the categories (keywords) represented in each candidate domain [Mahata *et al.* 2018]. If sufficient data of suitable quality is available, then supervised learning approaches are the best-suited to the keyword assignment problem.

While unsupervised learning approaches are typically better suited to keyword extraction than to keyword assignment, we may still leverage significant benefit from unsupervised pre-training of language models in subsequent supervised learning tasks [Devlin *et al.* 2019; Ramasubramanian *et al.* 2020]. Language models can be pre-trained in an unsupervised or semi-supervised manner on large text corpora (generalised or domain-specific) to learn underlying semantic and syntactic interactions characteristic of the language and area of application. This approach enables words to be converted into dense real-valued vectors that can encode underlying information about the document in which the words exist, including co-occurrence statistics, context, and so forth. These vectors, known generally as word embeddings, have become a crucial component of most modern Natural Language Processing/Understanding (NLP/NLU) architectures.

Developments in the expressive capabilities of word embeddings are now the cornerstone of advancements in performance of most natural language tasks e.g. question answering, textual entailment, sentiment analysis, named entity recognition, sequence tagging and classification, among others [Devlin *et al.* 2019; Pennington *et al.* 2014; Melamud *et al.* 2016; Peters *et al.* 2017 2018; Mikolov *et al.* 2013; Wu *et al.* 2016].

2.1 Word Embeddings

Mikolov *et al.* [2013] developed Word2Vec, a set of neural architectures that enable the efficient computation of real-valued vector representations of words, known as word embeddings. When trained using a given corpus, these word vectors encode the semantic and syntactic similarities between words as they appear in the training corpus. Word embeddings have been foundational to furthering the state-of-the-art in many natural language applications since their inception and are considered as universal feature extractors for natural language [Ramasubramanian *et al.* 2020]. Word2Vec is a two-layer neural network and may be trained using different word representations, depending on the requirements of the downstream task. Two architectures are proposed for training Word2Vec [Mikolov *et al.* 2013]:

- Common Bag of Words (CBOW): input word sequences are processed sequentially, and the network is required to predict the current word, given a set of words in the same context, where the order of the words is not considered.

- Continuous Skip-gram Model: similar in architecture to CBOW, the objective of the skip-gram model is to use the current word to predict words in the same context.

CBOW performs well on rare words and does not require a lot of data, while the skip-gram performs well for more abundant words and is less computationally expensive.

Pennington *et al.* [2014] observe that the linear substructures captured by models like Word2Vec may be enriched by incorporating a mechanism to capture global word co-occurrence statistics in the word embedding representations. To this end, they developed the GloVe (Global Vectors for Word Representation) model, which furthered the state-of-the-art in word analogy, word similarity, and named entity recognition tasks.
While Word2Vec and GloVe have been among the most popular word embedding models for many tasks, Bojanowski *et al.* [2016] observe that the morphology of words and phrases are not captured by these models. The word embedding models generate vectors for each word individually, which fails to account for morphological variations that are common in some languages. To address this limitation, the specification of the skip-gram model is modified such that each word is represented as bag of character n-grams [Bojanowski *et al.* 2016]. Words are then represented as combinations of the vectors of the character n-grams of which they are composed. This strategy caters for morphologically rich languages and rare words that may not appear often in the training corpus.
Apart from failing to capture the morphological variations of words, Word2Vec and GloVe models also fail to capture multi-word phrases. For many natural language tasks, learning phrase embeddings intermixed with unigram word embeddings can improve performance and accuracy [Mahata *et al.* 2018; Yin and Schütze 2014]. Yu and Dredze [2015] present the Feature-rich Compositional Transformation, a method for composing phrase embeddings from features extracted from the component words via learned transformation functions.

2.2 Context-capturing Word Embeddings

The Word2Vec and GloVe word embedding models are also limited in their ability to capture context because they only consider context within a limited distance from the target word, and the order of the words in the context window is not strongly modelled. This means that the syntactic substructures and co-occurrence statistics captured by word embedding vectors fail to capture the underlying interactions resulting from the ordering of words in either direction of the target word. There have been many recent works that have aimed to develop more advanced word embedding models that can capture context more thoroughly using LSTM neural networks [Melamud *et al.* 2016; Peters *et al.* 2017 2018; Devlin *et al.* 2019; Radford 2018]. A common theme that has emerged in these works is that large corpora of unlabelled texts can be leveraged to pre-train context-capturing word embedding models that allow for shorter training times and improved performance in downstream natural language tasks. Effectively capturing context allows for modelling of complex characteristics of word use, as well as how word use varies across different contexts (polysemy).

The most promising of these works are derivatives of the transformer architecture proposed by Vaswani *et al.* [2017]. These architectures are able to achieve deeply bi-directional context sensitivity between all words in a given input sequence, with significant computational efficiency. Most other state-of-the-art language models capture context through sequence transduction mechanisms that rely on recurrent or convolutional neural networks in conjunction with encoder-decoder structures. In these networks, the computational complexity of relating signals between arbitrary input and output positions grows as the distance between these positions increases [Vaswani *et al.* 2017]. In contrast, transformers can achieve this relation to arbitrary distances with a constant number of operations, through a self-attention mechanism in conjunction with encoder-decoder structures. Apart from being able to achieve context awareness to arbitrary depths without incurring scaling computational penalty, the transformer architecture is not dependent on sequential processing to produce outputs, as is the case with recurrent models [Vaswani *et al.* 2017]. This means that transformer models allow for a high degree of parallelization, such that models can be trained in much shorter times on GPUs and TPUs [Vaswani *et al.* 2017; Devlin *et al.* 2019].

Radford [2018] made use of the transformer architecture to develop the OpenAI-GPT model, which is able to capture context deeply (across the entire sequence length) from left-to-right of any input token. The OpenAI-GPT model achieved state-of-the-art results in sentence-level tasks from the GLUE benchmark at the time of publication. Subsequently, Devlin *et al.* [2019] enhanced the transformer-based model architecture to produce BERT, which is able to capture context deeply in both directions from any input token. The BERT model obtained a new state-of-the-art in eleven NLP tasks at the time of publishing and remains one of the most successful and efficient language models at the time of writing.
Both OpenAI-GPT and BERT models benefit from pre-training the language models on large unlabelled text corpora. Pre-training greatly reduces training time and increases accuracy of many downstream natural language tasks because only the custom

(task-specific) model parameters need to be learned from scratch, while the language model parameters need only be fine-tuned to the task at hand. Furthermore, it is the same error signal and backpropagation updates that achieves both of these tasks simultaneously when training on downstream tasks. The ability to transfer learning via pre-training between tasks also means that domain-specific pre-training can be affected to produce higher performance in domain-specific tasks [Beltagy *et al.* 2019].

2.3 The Keyword Assignment Task

Word embeddings have formed the basis of previously successful keyword assignment tasks. Mahata *et al.* [2018] used a variant of Word2Vec including phrase embeddings and proposed a framework for selecting and ranking keywords for two benchmark datasets (Inspec [Hulth 2003] and SemEval [Kim *et al.* 2010]). Ramasubramanian *et al.* [2020] attempted to solve the task of automatic assignment of keywords to texts in the Earth Science domain. They trained a domain-specific Word2Vec model and developed a custom neural classifier to automatically assign keywords to the abstracts of Earth Science publications. This research demonstrated the value of domain-specific pre-training in the keyword assignment task, which achieved a notably higher performance than the generic Word2Vec model, and high performance overall (>78% precision).
In contrast, Zhang and Xu [2009] used a citation-KNN methodology to tackle the keyword assignment task and achieved relatively poor precision (<36%). Their approach modified the K-Nearest Neighbour algorithm objective to use a compound classification decision function that considers a weighted contribution of publications in the citation-quotation network of the test sample. The classification decision function makes use of the class labels of the nearest neighbours, taking into account the cosine similarity between documents, quotation frequency and PageRank value [Langville and Meyer 2004]. It must be noted that Zhang and Xu [2009] achieved better results with citation-KNN than the standard KNN algorithm, but poorer results than other methods that make use of word embeddings.

2.4 Conclusion

Word embedding techniques are a ubiquitous and effective means of extracting features from input texts, which can then be used in downstream Natural Language Processing (NLP) and Natural Language Understanding (NLU) tasks. The means by which word embedding representations are generated determine the amount of information carried in the embeddings, with respect to the corpus-level occurrence and co-occurrence statistics of words, as well as the contextual interactions of words in the training corpus. In general, one expects that word embedding techniques that are able to effectively capture more complex interactions in the training corpus will perform better in complex downstream tasks,when compared to those that capture more simple corpus-level statistics. Previous works have found success in using Word2Vec representations, coupled with multi-layer neural classifiers for complex keyword assignment.

Chapter 3

Data and Models

This Chapter presents the high-level approach, research questions, data, data preparation, and models investigated in this study.

3.1 High-level Approach

In this study, a similar approach to Ramasubramanian *et al.* [2020] is taken to solving the task of assigning keywords to the abstracts of scientific publications. Word embeddings from language models of varying complexity are used in conjunction with multi-layer neural classifiers, with the aim of comparing the relative performance of each architecture when applied to the keyword assignment task, as well as exploring the impact of various design parameters and hyper-parameters. Three classes of word embeddings are investigated:

- Bag-of-Words (BOW): A simple representation used as a baseline for comparison. The input texts are represented as a sparse one-hot encoded vector having the same length as the input vocabulary. The BOW representation only encodes which words are present in the input and does not contain any information about the occurrence statistics, co-occurrence statistics, word order, or context.

- Word2Vec: As discussed in Section 2.1, the Word2Vec embedding is derived from a simple neural network architecture with a learning objective that allows the model to capture corpus-level occurrence statistics, and co-occurrence statistics between words. The resulting dense real-valued vector representations of words do not encode any information with respect to word order, and do not capture word context deeply across the input. It is common practice to pre-train models like Word2Vec on large text corpora, such that the learned word embeddings can be used subsequently as an input to downstream natural language tasks [Devlin *et al.* 2019]. For highly domain-specific tasks, it may be preferable to use a large domain-specific text corpus and pre-train the Word2Vec model from scratch [Ramasubramanian *et al.* 2020]. If the domain-specific text corpus is not large enough, it may be beneficial to start with a generic pre-trained model and conduct further pre-training using the domain-specific corpus.

- Transformers: As discussed in Section 2.2, transformer models produce dense real-valued representations of words that encode corpus-level and document-level contextual interactions between words, as well as their relative positioning in each document. Transformers also offer the benefit of transfer learning, where language models that are pre-trained on large and diverse text corpora are used as the starting point. The weights of the language models and are then fine-tuned when coupled to downstream neural network architectures and trained on downstream tasks. Two categories of transformer models are investigated:

  - Left-to-right: These models capture context across the entire input sequence from left to right. This capability results from a causal language modelling (CLM) pre-training objective [Radford 2018]. In the CLM task, random tokens in the input sequence are masked and the model has the task of predicting the identity of the masked token, given all the preceding tokens. When using these embeddings for downstream classification tasks, the embedding of of the last token in the sequence is used as input to the classifier. This study uses the **OpenAI-GPT (Generative Pre-trained Transformer)** model as the representative example of this class of language model.
  - Bi-directional: These models capture context deeply in both directions from each token. This is made possible by exploiting the self-attention mechanism of the transformer and introducing a masked language modelling (MLM) pre-training objective [Devlin *et al.* 2019]. The MLM task operates by masking random tokens in the input sequence and the model has the task of predicting the identity of the masked tokens, given all the other tokens in the sequence (both before and after). Each word is effectively able to "see" all other words in the sequence, as well as itself, and these interactions are captured as positional embeddings at the input side of the transformer. Bi-directional context awareness has been shown to improve performance in natural language tasks such as question answering, when compared to similar architectures using left-to-right models. A special "classification" token is included at the start of every sequence, and it is the embedding of this token that is used for downstream classification tasks. The **BERT (Bi-directional Encoder Representations from Transformers)** model is used as the representative example of this class of language model.

This selection of models allows for the study of how the choice of word embeddings affects classification accuracy when using similar multi-layer neural network classification model architectures, with the word embedding vectors as input. Specifically, the goal is to find the most effective architecture for keyword assignment in scientific publications. In addition to the choice of word embeddings, the following parameters are explored:

- Classifier architecture (number of hidden layers and neurons per layer)
- Learning rate and optimisation strategy
- Classifier threshold selection and classifier sensitivity

- Variations in pre-training data
- Data splitting and selection strategy
- Target variable dimensionality reduction.

3.2 Research Questions

This study aims to address the following questions:

- Which language model produces the best performance (BOW, Word2Vec, BERT, or OpenAI GPT)?
- Can performance be improved by introducing additional domain-specific pretraining of Word2Vec and transformer models?
- Can performance be improved by training separate models for each journal, instead of one model over a combination of journals?
- Can performance be improved by pruning infrequently-occurring keywords from the data?
- Which classifier architecture produces the best performance over the hyper-parameter combinations tested?
- Which optimisation and data splitting strategies produce the best performance of those tested?
- Are the proposed methods suitable for an automated keyword assignment pipeline that could be used by publishers of academic research, to assist authors and editors with accurate keyword suggestions?

3.3 Data

All data used in this study was collected by means of frontend web scraping from the respective journal websites. All data used is open to public access and is not, to the best of my knowledge, restricted in any way that would prohibit this research.

3.3.1 Primary Datasets

The primary datasets used in this study consist of the abstracts of scientific journal articles and their associated assigned keywords. These datasets are used to train and evaluate the models on the task of keyword assignment and journal prediction.

Nature Dataset

This dataset consists of all published abstracts and letters from the following four Nature journals:

- Nature Climate (1044 Abstracts)
- Nature Medicine (1334 Abstracts)
- Nature Materials (1480 Abstracts)
- Nature Physics (1370 Abstracts)

These fields were chosen heuristically with the aim of producing a dataset with distinct thematic and semantic variation, as well as having a comparable number of examples from each field. There is a total of 5228 abstracts in the dataset. In submitting publications to Nature, the authors supply keywords, which are then subject to review and by the journal editors, who have the task of "choosing keywords to maximize visibility in online searches and that are suitable for indexing services" [https://www.nature.com/nature-research/for-authors/publish]. It is therefore assumed that the keywords (data labels) are sufficiently accurate and consistent for the classification task at hand. There are 1234 unique keyword phrases in the dataset, with an average of 2.73 keywords per paper.

IEEE Dataset

This dataset consists of all published abstracts from the following two IEEE journals:

- IEEE Transactions on Geoscience and Remote Sensing (11569 Abstracts)
- IEEE Transactions on Pattern Analysis and Machine Intelligence (6196 Abstracts)

The IEEE journals employ a slightly different approach to assigning keywords compared to Nature journals. Every paper can have up to four sets of keywords:

- IEEE Keywords: Assigned by the journal editors from a controlled keyword vocabulary (IEEE). All papers have a set of IEEE Keywords. There are 3174 unique IEEE keywords in the dataset, with an average of 8.31 keywords per paper.
- Author Keywords: Assigned by the authors of the paper. These keywords tend to be more specific to the nuances of the research than the IEEE keywords. Approximately 66.95% of papers in this dataset have a set of author keywords. Due to the specificity of the keywords chosen by authors, there are far more of these keywords in total than there are IEEE keywords: there are 24674 unique author keywords in the dataset, with an average of 5.01 keywords per paper.
- INSPEC: Controlled Indexing: These keywords are not present for all papers and are disregarded in this study.
- INSPEC: Non-Controlled Indexing: These keywords are not present for all papers and are disregarded in this study.

The IEEE dataset represents a larger collection of papers per domain, as well as a wider variety of keyword assignment methodologies to explore. In combination with the Nature dataset, this allows for investigation of the effect of sample size, label size, and label quality on model performance for the keyword assignment task.

3.3.2 Pre-training Dataset

Additional pre-training of the language models is be carried out using abstracts from the following Elsevier journals:

- Applied Energy (15364 Abstracts)
- Archives of Medical Research (2125 Abstracts)
- Ecological Indicators (6775 Abstracts)
- Environmental Science and Policy (2515 Abstracts)
- Environmental and Sustainability Indicators (91 Abstracts)
- Human Resource Management Review (785 Abstracts)
- Journal of Applied Developmental Psychology (1530 Abstracts)
- Journal of Choice Modelling (260 Abstracts)
- Materials Research Bulletin (14710 Abstracts)
- Journal of Materials Science & Technology (3316 Abstracts)
- Journal of Physics and Chemistry of Solids (9490 Abstracts)
- Journal of Vocational Behavior (3168 Abstracts)
- Resources, Conservation and Recycling (4147 Abstracts)
- Results in Physics (3211 Abstracts)
- Reviews in Physics (83 Abstracts)
- Social Sciences & Humanities Open (109 Abstracts)

This dataset has a total of 67679 abstracts, containing 11.2M words spanning a vocabulary of 98.5k words. The purpose of this dataset is to undertake additional pre-training of language models that have already been pre-trained on large text corpora. It must be noted that this is a relatively small amount of data to undertake pre-training from scratch [Liu *et al.* 2019], however it is postulated that a pre-trained model may benefit from additional domain-specific pre-training before being applied to the domain-specific task.

3.3.3 Data Preparation

Data pre-processing is required to remove special symbols, numbers, and punctuation from the data before it is used for modelling. The objective is to reduce the data into only the semantic content which carries the most information that can be used for training the classifiers. Removing all such characters also reduces model complexity, which arises from both the addition of tokens to the vocabulary as well as activation of the corresponding nodes in the network.
The following processing is done on all input text:

- All characters are converted to lowercase.
- All punctuation is removed.
- All numbers and special symbols are removed, unless they form part of a compound alphanumeric term.

Additional pre-processing is carried out on all keyword terms:

- Stopwords are removed (as defined in genism library).
- Words are lemmatized using NLTK library and Wordnet Lemmatizer. This process converts words to their base form, which is done to deal with potential inconsistencies in the data labels (synonyms, plurals, etc.) and thereby reducing the potential for redundant labels.

When using the bag-of-words model, this additional pre-processing is also carried out on the input texts. When using transformer models, the remaining words in the input texts are then tokenized according to the requirements of the model [Devlin *et al.* 2019; Radford 2018]. Any tokens that are not present in the pre-trained vocabulary are added to the vocabulary. These processing steps are done to standardise and simplify the input. We are interested only in the words that carry information, and the order in which the words appear. Numerical values present in the abstracts typically reflect quantities related to the results of the article, which should not provide information that aids in the classification of the text.

3.4 Models

Before specifying the text representations investigated in this study, it is necessary to note the following:

- V is the set of tokens in the vocabulary *i.e.* the set of unique tokens appearing in the train, test, and validation datasets.
- The term ”sample” and ”document” are interchangeable.

3.4.1 Bag-of-Words

The bag-of-words method is used as a baseline representation of the input texts. This is simply a one-hot encoded vector that indicates which words are present in the input text, out of all possible words in the vocabulary. This is represented as follows:

$$U_{\mathrm{i}} \in \mathbb{Z}^{\mathrm{D}} \tag{3.1}$$

$$U_{\mathrm{i}} = \begin{bmatrix} w_{\mathrm{i,1}}, & w_{\mathrm{i,2}}, & \dots & w_{\mathrm{i,D}} \end{bmatrix} \tag{3.2}$$

$$w_{\mathrm{i,k}} = \begin{Bmatrix} 1 & if & v_{\mathrm{k}} \in x_{\mathrm{i}} \\ 0 & if & v_{\mathrm{k}} \notin x_{\mathrm{i}} \end{Bmatrix} \tag{3.3}$$

Where,

- $\boldsymbol{U_i}$ is the bag-of-words representation of the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{D}$ is the number of tokens in the vocabulary
- $\boldsymbol{w_{i,\,k}}$ is the integer [0,1] representation of the $\boldsymbol{k^{th}}$ token in the vocabulary for the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{v_k}$ is the $\boldsymbol{k^{th}}$ token in the vocabulary
- $\boldsymbol{x_i}$ is the set of tokens in the $\boldsymbol{i^{th}}$ sample

In this case, each token in the vocabulary is simply a unique word *i.e.* words are not broken up into components.

3.4.2 Word2Vec

Word2Vec embeddings used in this study consist of 300-element real-valued vectors corresponding to each token in the vocabulary. The embeddings for each sample are defined as follows:

$$e_{\mathrm{i,n}} \in \mathbb{R}^{300} \tag{3.4}$$

$$U_{\mathrm{i}} \in \mathbb{R}^{300} \tag{3.5}$$

$$U_{\mathrm{i,j}} = \frac{1}{N_{\mathrm{i}}} \sum_{n=1}^{N_{\mathrm{i}}} e_{\mathrm{i,n,j}} \tag{3.6}$$

Where,

- $\boldsymbol{e_{i,n}}$ is the vector representation for the $\boldsymbol{n^{th}}$ token of the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{e_{i,n,j}}$ is the $\boldsymbol{j^{th}}$ element of the vector representation of the $\boldsymbol{n^{th}}$ token of the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{U_i}$ is the Word2Vec representation of the $\boldsymbol{i^{th}}$ sample

- $U_{i,j}$ is the j^{th} element of the word embedding representation of the i^{th} sample
- N_i is the number of tokens in the i^{th} sample

In the case of Word2Vec, each token in the vocabulary is simply a unique word *i.e.* words are not broken up into components. Two variants of the Word2Vec model are investigated in this study:

- Word2Vec: This is the standard pre-trained Word2Vec model, pre-trained on the Google News dataset [Mikolov *et al.* 2013].
- Word2Vec Extra: The standard Word2Vec model is further pre-trained using the custom pre-training dataset (Section 3.3.2) for 2000 epochs to produce the Word2Vec Extra model. The pre-training dataset is not large enough to adequately train a Word2Vec model from scratch [Ramasubramanian *et al.* 2020], which is why this approach is taken.

3.4.3 BERT (Bi-directional Encoder Representations from Transformers)

The outputs produced by the BERT language model are real-valued vectors for each token in the input text, and each vector consists of 768 elements. The embeddings for each sample are defined as follows:

$$e_{\mathrm{i,n}} \in \mathbb{R}^{768} \tag{3.7}$$

$$E_{\mathrm{i}} = \begin{bmatrix} e_{\mathrm{i},1} \\ e_{\mathrm{i},2} \\ \vdots \\ e_{\mathrm{i},\mathrm{N_i}} \end{bmatrix} \tag{3.8}$$

$$U_{\mathrm{i}} = e_{\mathrm{i},1} \tag{3.9}$$

Where,

- $e_{i,n}$ is the vector representation for the n^{th} token of the i^{th} sample
- E_i is the output of the BERT model for the i^{th} sample
- N_i is the number of tokens in the i^{th} sample
- U_i is the word embedding representation that is used for the i^{th} sample

In the case of BERT, each word is tokenized by splitting words into their components (e.g. base + suffix). This means that the vocabulary accounts separately for affixes, which will be fed into the model as individual tokens. Three variants of the BERT model are tested in this study:

- BERT: This is the standard BERT model, which has been pre-trained on the Google Books Corpus and English Wikipedia Corpus (Devlin *et al.* [2019]).

- BERT Extra: The standard BERT model is further pre-trained using the custom pre-training dataset (Section 3.3.2), using masked language modelling for 100 Epochs, to produce the BERT Extra model.
- SciBERT: This model has the same architecture as BERT and is pre-trained on a large corpus of scientific publications. This model shows improved performance in natural language tasks involving scientific publications when compared to the BERT model with standard pre-training [Beltagy *et al.* 2019].

3.4.4 OpenAI-GPT (Generative Pre-trained Transformer)

The outputs produced by the OpenAI-GPT language model are real-valued vectors for each token in the input text, and each vector consists of 768 elements. The embeddings for each sample are defined as follows:

$$e_{\mathrm{i,n}} \in \mathbb{R}^{768} \tag{3.10}$$

$$E_{\mathrm{i}} = \begin{bmatrix} e_{\mathrm{i,1}} \\ e_{\mathrm{i,2}} \\ \vdots \\ e_{\mathrm{i,N_i}} \end{bmatrix} \tag{3.11}$$

$$U_{\mathrm{i}} = e_{\mathrm{i,N_i}} \tag{3.12}$$

Where,

- $\boldsymbol{e_{i,n}}$ is the vector representation for the $\boldsymbol{n^{th}}$ token of the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{E_i}$ is the output of the OpenAI-GPT model for the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{N_i}$ is the number of tokens in the $\boldsymbol{i^{th}}$ sample
- $\boldsymbol{U_i}$ is the word embedding representation that is used for the $\boldsymbol{i^{th}}$ sample

In the case of OpenAI-GPT, each word is tokenized in a similar manner to BERT. The standard pre-trained OpenAI-GPT model is used in this study, which has been pre-trained on the BooksCorpus dataset [Radford 2018].

3.5 Conclusion

Two primary datasets are used in this study, consisting of journals from 2 separate publishers (Nature and IEEE) with different standards for manual keyword assignment. Journals from a third publisher (Elsevier) are used for pre-training. Three different feature extraction strategies are investigated *i.e.* **bag-of-words**, **Word2Vec**, and **transformer** models. Two different **transformer** architectures are studies *i.e.* **BERT** and **OpenAI-GPT**. The impact of pre-training datasets are investigated for the **BERT** and **Word2Vec** models.

Chapter 4

Research Methodology

This Chapter presents the classification tasks, model architectures and experimental design employed in this study.

4.1 Classification Tasks

Two classification tasks are explored in this study *i.e.* **domain prediction** and **keyword prediction**. The domain prediction task is to predict the journal (domain) that each abstract belongs to. The keyword prediction task is to predict a set of keywords (labels) for each abstract, from a controlled vocabulary of keywords.

4.1.1 Domain Prediction

For the Nature dataset there are 4 distinct domains *i.e.*

- Climate
- Medicine
- Materials
- Physics

For the IEEE dataset there are two distinct domains *i.e.*

- Geoscience and Remote Sensing
- Pattern Analysis and Machine Intelligence

4.1.2 Keyword Prediction

For both primary datasets, the affect of reducing the dimensionality of the target vector is investigated by pruning out the least frequently occurring keywords from the dataset. Three variations are investigated *i.e.* 100%, 90%, and 80% of the most frequently-occurring keywords are retained, while the rest are discarded. The only exception to this approach is when using the Author keywords of the IEEE dataset, in which

case all keywords appearing less than 5 times in the dataset are removed, which is the same approach implemented by Ramasubramanian *et al.* [2020]. The distribution of keywords in each case is provided below, along with the counts of the maximum-, minimum- and mean number of occurrences of keywords in the dataset.

- Nature Dataset:
  - 100% - 1234 Keywords (Max: 384, Min: 1, Mean: 11.5)
  - 90% - 499 Keywords (Max: 384, Min: 5, Mean: 25.7)
  - 80% - 300 Keywords (Max: 384, Min: 11, Mean: 38)

  The occurrence statistics of unique keyword terms in papers from the different journals is represented in Figure 4.1 below.

- IEEE Dataset: For the majority of experiments carried out in this study, only the IEEE keyword terms were used, while the Author keywords and INSPEC keywords were disregarded. Only the BERT Extra model is tested on the task of predicting Author keywords.
  - 100% - 3174 IEEE Keywords (Max: 2044, Min: 1, Mean: 46.5)
  - 90% - 805 IEEE Keywords (Max: 2044, Min: 28, Mean: 165.1)
  - 80% - 467 IEEE Keywords (Max: 2044, Min: 69, Mean: 253.2)

  The distribution of Author keywords after removing all keywords that occur less than 5 times is as follows:
  - Unique Author Keywords: 1802
  - Max: 902
  - Min: 5
  - Mean: 16.6

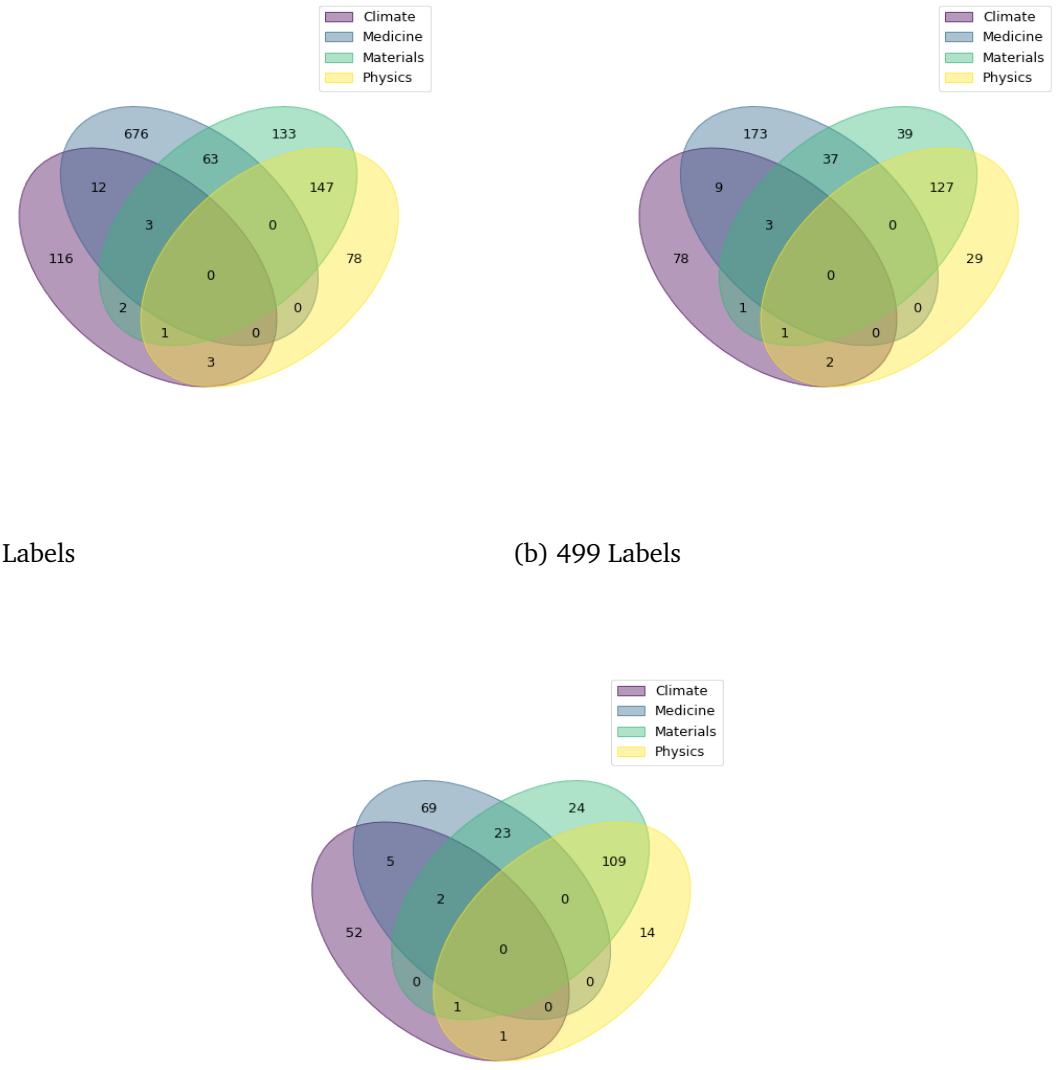
  The occurrence statistics of IEEE- and Author keyword terms are represented in Figure 4.2 below.

For both primary datasets (Nature and IEEE), and in addition to the exhaustive experiments carried out with each language model and classification architecture, the BERT Extra model is used to predict the keywords for abstracts for each journal separately. This is carried out with the best-performing classification architecture identified in the main body of the study. The data is segmented by journal and individual models are trained to perform keyword prediction for each journal, using 100% of the keywords for the respective journal *i.e.* without label-pruning. This results in the following data distribution:

- Nature Dataset:
  - Climate - 137 Keywords, 1044 Abstracts
  - Medicine - 754 Keywords, 1334 Abstracts

- Materials - 349 Keywords, 1480 Abstracts
- Physics - 229 Keywords, 1370 Abstracts

- IEEE Dataset:
  - Geoscience/Remote Sensing - 2548 IEEE Keywords, 11569 Abstracts
  - Pattern Analysis/Machine Intelligence - 2253 IEEE Keywords, 6196 Abstracts

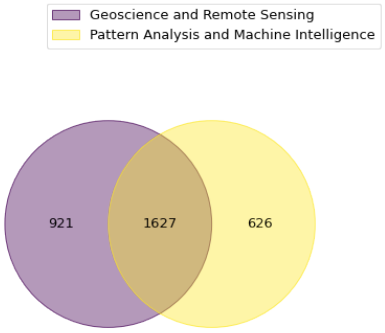


(a) 1234 Labels

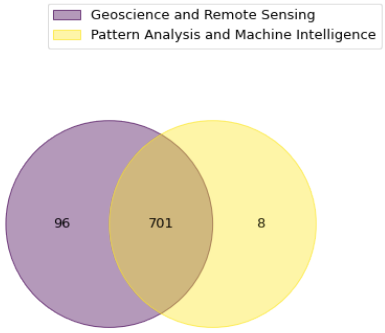
(b) 499 Labels

(c) 300 Labels

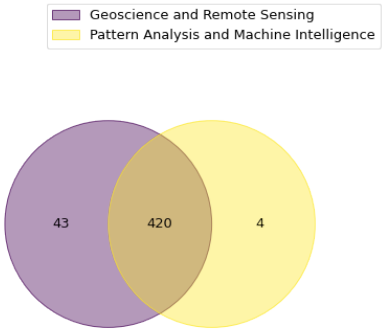
Figure 4.1: Distribution of unique keyword terms per journal in the Nature dataset



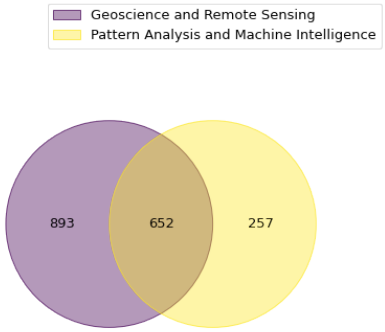
(a) 3174 Labels - IEEE Keywords



(b) 805 Labels - IEEE Keywords



(c) 467 Labels - IEEE Keywords



(d) 1802 Labels - Author Keywords

Figure 4.2: Distribution of unique IEEE keyword terms per journal in the IEEE dataset

The distribution of keywords in the Nature dataset is not balanced. The Medicine journal has the highest number of keywords, with very few being shared by other domains. The Climate journal has the lowest number of keywords and also has minimal overlap with other domains. There is significant overlap between the Materials and Physics journals, and a smaller degree of overlap between the Materials and Medicine journals. The Physics journal does not overlap with Medicine at all, and has negligible overlap with the Climate journal. In general, we expect an inverse relationship between model performance and the average number of unique keywords per sample. It should

also be noted that the imbalance in the distribution of keywords is significantly altered by the process of label pruning. This means that the Medicine journal represents a large proportion of the infrequent keywords in the dataset *i.e.* nichè classifications. The most abundant keywords are mostly those that are shared between journals, however each journal does contain some abundant keywords that are not shared by other journals.

In contrast to the Nature dataset, the IEEE dataset represents a large overlap of IEEE keywords between the two journals studied. This is particularly clear after label pruning because most of the abundant keywords are those occurring in both journals. In the case of Author keywords for the IEEE dataset, there is a very large number of low-abundance keywords, which represent nichè topics or specifics of the paper that are not widely applicable classifications. After pruning all keywords that appear less than 5 times in the dataset, the number of keywords remaining is less than the un-pruned IEEE keywords, but more than double the number of the top 90% of IEEE keywords. There is still significant overlap between the two journals after pruning the Author keywords, but there are significantly more keywords that are exclusively in the IEEE Transactions of Geosciences and Remote Sensing.

4.2 Classification Architectures

The same design philosophy is applied to all cases, irrespective of the dataset or language model chosen. The language model is seen as a feature extractor which produces a representation that feeds into neural classifiers. The classifiers consist of fully-connected linear layers, with varying numbers of both hidden layers and neurons per layer. In the case of domain prediction, determining the predicted domain is done by simply identifying the output node with the highest value. In the case of keyword prediction, the model outputs are normalised with a sigmoid function and then binarised with a threshold to make predictions. As an alternative method, the classifier outputs can be ordered by magnitude, and a fixed number of keywords selected from the top n outputs. In the tables presented in this section, "Vocab Size" refers to the number of unique tokens in the vocabulary, and "N_{Labels}" refers to the number of unique keywords in the dataset (labels).

4.2.1 Bag-of-Words Architectures

The bag-of-words representation is used as an input to neural networks of two sizes *i.e.* 5- or 10 hidden layers. The architectures studied are shown in Table 4.1.

	Model 1	Model 2
Hidden Layers	5	10
Contracting Layers	Yes	No
Hidden Layer 1 Size	(Vocab Size X 768)	(Vocab size X 768)
Hidden Layer 2 Size	(768 X 672)	(768 X 768)
Hidden Layer 3 Size	(672 X 576)	(768 X 768)
Hidden Layer 4 Size	(576 X 480)	(768 X 768)
Hidden Layer 5 Size	(480 X 416)	(768 X 768)
Hidden Layers 6-10 Size	-	(768 X 768)
Classifier Size	(416, N_{Labels})	(768, N_{Labels})

Table 4.1: Proposed classifier architectures for the bag-of-words models

The architecture of **bag-of-words Model 1** was chosen because it follows the same strategy as that presented by Ramasubramanian *et al.* [2020] *i.e.* steadily decreasing hidden layer size from one layer to the next. The size of the first layer was chosen to be equal to the embedding size of the transformer models. The architecture of **bag-of-words Model 2** was chosen to investigate the impact of additional model complexity by leaving the number of neurons per hidden layer constant and doubling the number of hidden layers.

4.2.2 Word2Vec Model Architectures

Word embeddings from the Word2Vec model are used as an input to neural networks of two sizes *i.e.* 4- or 8 hidden layers. Each of the input tokens is represented by a real-valued word embedding vector having 300 elements; the input embeddings are averaged to produce the input to the classifiers (Section 3.4.2). The architectures studied are shown in Table 4.2.

	Model 1	Model 2
Hidden Layers	4	8
Contracting Layers	Yes	No
Hidden Layer 1 Size	(300 X 260)	(300 X 300)
Hidden Layer 2 Size	(260 X 220)	(300 X 300)
Hidden Layer 3 Size	(220 X 200)	(300 X 300)
Hidden Layer 4 Size	(200 X 180)	(300 X 300)
Hidden Layers 4-8 Size	-	(300 X 300)
Classifier Size	(180, N_{Labels})	(300, N_{Labels})

Table 4.2: Proposed classifier architectures for the Word2Vec models

The architecture of **Word2Vec Model 1** was chosen because it is the same as that presented by Ramasubramanian *et al.* [2020]. The architecture of **Word2Vec Model 2** was chosen to investigate the impact of additional model complexity by leaving the number of neurons per hidden layer constant and doubling the number of hidden layers.

4.2.3 Transformer Classifier Architectures

The BERT model is trained with the inclusion special classification token at the start of every input sequence [Devlin *et al.* 2019]. This means that the classification token is used throughout pre-training and fine-tuning operations. The BERT model produces real-valued vector representation of every word in the input sequence, each having 768-elements. It is the embedding of the classification token that is used as an input to downstream classifiers when using models based on the BERT architecture (Section 3.4.3). The OpenAI-GPT model has is no specific classification token; the embedding of the last token in the input sequence is used for this purpose (due to the causal language modelling training objective). The OpenAI-GPT model also produces 768-element word embeddings (Section 3.4.4). In all cases, the embedding used for classification is first passed through a fully-connected linear layer (the pooling layer) to scale and transform the embedding, as recommended by Devlin *et al.* [2019] and Mikolov *et al.* [2013]. The architectures studied are shown in Table 4.3.

	Model 1	**Model 2**	**Model 3**	**Model 4**
Hidden Layers	2	2	4	4
Contracting Layers	Yes	No	Yes	No
Pooling Layer Size	(768 X 768)	(768 X 768)	(768 X 768)	(768 X 768)
Hidden Layer 1 Size	(768 X 576)	(768 X 768)	(768 X 672)	(768 X 768)
Hidden Layer 2 Size	(576 X 416)	(768 X 768)	(672 X 576)	(768 X 768)
Hidden Layer 3 Size	-	-	(576 X 480)	(768 X 768)
Hidden Layer 4 Size	-	-	(480 X 416)	(768 X 768)
Classifier Size	(416, N_{Labels})	(768, N_{Labels})	(416, N_{Labels})	(768, N_{Labels})

Table 4.3: Proposed classifier architectures for the transformer models

The same rationale is again employed in the architecture selection strategy for the **transformer** models *i.e.* contracting layers after the example of Ramasubramanian *et al.* [2020], and non-contracting layers to investigate the impact of added model complexity. Both of these cases are tested with both 2- and 4 hidden layers, with 4 hidden layers (contracting) again being after the example of Ramasubramanian *et al.* [2020]. In addition to the above, experiments are also carried out with **no hidden layers (Model 5)**, using only the output from the pooling layer as an input into a fully connected classification layer. The architecture of **Model 5** is selected because this is what is what is recommended by Devlin *et al.* [2019].

4.2.4 Model Size Comparison

The total number of trainable parameters for each model is summarised in Table 4.4.

	Model 1	Model 2	Model 3	Model 4	Model 5
BOW	9.6M	14.2M	-	-	-
Word2Vec	8M	8.5M	-	-	-
BERT	109.8M	110.4M	110.5M	111.6M	109.2M
OpenAI-GPT	116.8M	117.4M	117.5M	118.6M	116.2M

Table 4.4: Number of trainable parameters per model

4.2.5 Architecture Design Parameters

All pooling layers and hidden layers make use of activation functions and Dropout [Srivastava *et al.* 2014]. The classification layer does not use any activation function. The design parameters for each model are summarised in Table 4.5.

	BOW	Word2Vec	BERT	OpenAI-GPT
Pooling Activation	-	-	Tanh	ReLU
Pooling Dropout	-	-	0.1	0.1
Hidden Activation	ReLU	ReLU	ReLU	ReLU
Hidden Dropout	0.1	0.1	0.1	0.1
Classifier Activation	-	-	-	-

Table 4.5: Proposed design parameters per model

The design parameters of the **bag-of-words** and **Word2Vec** models are the same as those implemented by Ramasubramanian *et al.* [2020]. The design parameters of the **BERT** models are the same as those implemented by Devlin *et al.* [2019]. The design parameters of the **OpenAI-GPT** models are also the same as those implemented by Devlin *et al.* [2019], with the exception of the pooling layer activation function. The ReLU activation was selected in favour of Tanh in this case, because this choice produced better results in preliminary testing with the the **OpenAI-GPT** models. It must also be noted that preliminary testing of the **BERT** models with ReLU activation functions for the pooling layer performed worse than Tanh, hence the proposed design parameters.

4.3 Experimental and Implementation Design

4.3.1 Data Split

The primary datasets are split into training, validation, and test sets in proportions of 70%, 15% and 15%, respectively. To reduce selection bias, abstracts from each journal are permuted and split by the requisite proportions separately, thus ensuring proportionate representation from each domain in the training, validation, and test sets. In addition to this data splitting strategy, the **BERT Extra** model is validated by 6-fold cross validation using the Nature dataset with the top 80% most abundant keywords. This cross-validation is carried out by permuting the data and splitting it

into 6 equally-sized partitions. Six models are then trained, each time holding out a different data partition as the validation set during training. Each of the 6 models is then evaluated by splitting the respective hold-out dataset in half *i.e.* one for validation (which is used to find the optimal threshold) and one for testing (to evaluate model performance at the optimal threshold).

4.3.2 Optimisation

This study makes use of Stochastic Gradient Descent (SGD) optimisation with Nesterov momentum and cyclic learning rate, as described by Smith [2017]. The learning rate bounds are found heuristically by inspection of the graph of training loss vs. learning rate; the upper bound is set 5-10x higher than the lower bound. The number of epochs in a half-cycle is set between 4-8 for all experiments. Cross-entropy loss function is used for the domain prediction task, while binary cross-entropy loss is used for the keyword prediction task. In all experiments, training is stopped when there is no improvement to the validation loss for 10 epochs, which is the same approach used by Ramasubramanian *et al.* [2020]. The **BERT Extra** model is trained for comparison using the ADAM optimiser (Section 5.3.4), global cyclic learning rate, and the Nature dataset with the top 80% most abundant keywords.

4.3.3 Pre-training

Pre-training is carried out using the pre-training dataset (Section 3.3.2), with the aim of improving the BERT and Word2Vec models for the classification task at hand. The BERT model has been pre-trained with texts from English Wikipedia and Google Books; the Word2Vec model has been pre-trained with texts from Google News.

BERT:

The BERT model is further pre-trained for 100 epochs to produce the BERT Extra model. The pre-training regime and configuration used is the same as that of Devlin *et al.* [2019], using masked language modelling with a masking probability of 15%.

Word2Vec:

The Word2Vec model is further pre-trained for 2000 epochs to produce the Word2Vec Extra model. This is done using the *gensim* python library with default parameters.

4.3.4 Fine-tuning

Bag-of-words:

The bag-of-words representation is not produced by any trainable model and therefore the term ”fine-tuning” is not strictly applicable. In the BOW case, training consists only of learning the weights of the fully-connected linear network that takes the BOW representation as input.These models have a relatively low complexity compared to

transformers and are able to train very quickly relative to most others used in this study.

Word2Vec:

In the case of Word2Vec, the word embedding for each word in the vocabulary is stored in an embedding layer in the model. This means that the embedding of each word has an associated weight to scale the embedding values *before* the embeddings of the input tokens are averaged to produce the input to the fully-connected linear layers. In this way, the embeddings of the words never change throughout training, but the model is able to learn scaling factors for each word, which is useful for preventing common words that carry little information from disproportionately impacting the input representation. The Word2Vec models have even lower complexity than the BOW models, and far fewer trainable parameters than the transformers.

Transformers:

All the transformer models have significantly more trainable parameters than the BOW and Word2Vec models, and take a much longer time to train. The parameters of the transformer model are trained by the same learning signal by which the fully-connected classification layers are trained. This means that the word embeddings change throughout training as a result of fine-tuning the parameters of the language model to the task at hand.

4.3.5 Implementation

All aspects of this study are implemented in python, using the following libraries:

- Web Scraping: Scrapy, Selenium
- Text Processing: Gensim, NLTK, Numpy, Pandas
- Modelling: Gensim (Word2Vec), Hugging Face (transformer APIs), PyTorch, PyTorch Lightning
- Visualisation: Matplotlib, Venn, HiPlot

4.3.6 Workflow

The study is carried out as follows:

1. All model architectures are trained on the domain prediction task using the Nature dataset.
2. All model architectures are trained on the keyword prediction task using the Nature dataset, over all three combinations of label pruning (100%, 90%, 80%).

3. For each model, the best-performing architecture from the preceding keyword prediction task is selected and trained on the IEEE dataset. Both domain prediction and keyword prediction are carried out, the latter using the **IEEE keywords** over all three combinations of label pruning.

4. The **BERT Extra** model is then further tested as follows:

    - Training on each domain of the Nature dataset in the keyword prediction task over 100% of keywords.
    - Training on each domain of the IEE dataset in the keyword prediction task over 100% of IEEE keywords.
    - 6-fold cross validation using the Nature dataset with the top 80% most abundant keywords.
    - Training using the ADAM optimiser and the Nature dataset with the top 80% most abundant keywords.
    - Training on the IEEE dataset with all **Author keywords** appearing more than 5 times in the dataset.

4.3.7 Performance Metrics

Domain Prediction

The domain prediction task is a single-label multi-class classification, which has a straightforward accuracy measure over a batch of samples:

$$A = \frac{C}{N} \tag{4.1}$$

Where,

- A is the accuracy
- C is the number of correct predictions
- N is the number of samples

Keyword Prediction

The keyword prediction task is more complex, with each sample having multiple labels, and the number of labels per sample is not fixed. To define performance metrics for keyword prediction, the following variables are first defined:

- N is the number of samples.
- Y_i is the set of assigned keywords for the i^{th} sample.
- x_i is the i^{th} sample.
- $h(x_i)$ is the set of keywords predicted by the model for the i^{th} sample by binarising the outputs and applying a threshold.

- $h(x_{i,\,n})$ is the set of n most probable keywords predicted by the model for the i^{th} sample, where n is set manually.
- $\mathbb{1}$ is the indicator function.
- l is the set of all labels appearing throughout the corpus.
- l_k is the k^{th} label.

Two approaches are used to evaluate model performance in the keyword prediction setting:

1. Threshold Prediction: The classifier outputs are normalised at each node with a sigmoid function and inference is done by applying a global threshold. In this way, the number of labels (keywords) returned will be variable, leading to the performance metrics given below:

    - Precision: Fraction of correct keywords that are returned, relative to the number of predictions returned, averaged over all samples:

$$Precision = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_{\mathrm{i}} \cap h(x_{\mathrm{i}})|}{|h(x_{\mathrm{i}})|} \tag{4.2}$$

    - Recall: Fraction of correct keywords that are returned, relative to the total number of correct keywords in the target set, averaged over all samples:

$$Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_{\mathrm{i}} \cap h(x_{\mathrm{i}})|}{|Y_{\mathrm{i}}|} \tag{4.3}$$

    - F_1 Score: Harmonic mean of Precision and Recall scores:

$$F_1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.4}$$

    - Jaccard Index: Intersection over Union of the set of keywords returned and the target set of keywords, averaged over all samples. Since the number of keywords required per text is not constant, the Jaccard index provides a single measure that takes into account both the number of keywords returned, and the accuracy of the returned keywords. The Jaccard index is defined as follows:

$$Jaccard\ Index = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_{\mathrm{i}} \cap h(x_{\mathrm{i}})|}{|Y_{\mathrm{i}} \cup h(x_{\mathrm{i}})|} \tag{4.5}$$

    - Exact Match Ratio: The fraction of samples that are perfectly classified:

$$Exact\ Match\ Ratio = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(Y_{\mathrm{i}} = h(x_{\mathrm{i}})) \tag{4.6}$$

The optimal threshold value for each model is found by traversing a range of thresholds in [0,1], and selecting the threshold value that maximises the F1 score for the validation set.

2. Ranked List Prediction: The classifier outputs are ranked by descending magnitude and the top *n* labels are selected. In this way, the number of labels inferred for each sample is constant and performance is measured as the proportion of true labels for the sample that are present in the ranked list. This is defined as follows:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_{\mathrm{i}} \cap h(x_{\mathrm{i,\,n}})|}{|h(x_{\mathrm{i,\,n}})|} \tag{4.7}$$

The measures presented so far are high-level performance indicators, but do not give insight into the distribution of performance over each of the labels. The following measures are included for analysis of label-level model performance:

- True Positive Rate: For a given label, the rate at which that label is correctly within the predicted labels, averaged over the corpus:

$$TP_{\mathrm{k}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}([l_{\mathrm{k}} \in Y_{\mathrm{i}}] \wedge [l_{\mathrm{k}} \in h(x_{\mathrm{i}})]) \tag{4.8}$$

  Where TP_k is the true positive rate for the k^{th} label.

- False Positive Rate: For a given label, the rate at which that label is incorrectly within the predicted labels, averaged over the corpus:

$$FP_{\mathrm{k}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}([l_{\mathrm{k}} \notin Y_{\mathrm{i}}] \wedge [l_{\mathrm{k}} \in h(x_{\mathrm{i}})]) \tag{4.9}$$

  Where FP_k is the false positive rate for the k^{th} label.

- True Negative Rate: For a given label, the rate at which that label is correctly absent from the predicted labels, averaged over the corpus:

$$TN_{\mathrm{k}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}([l_{\mathrm{k}} \notin Y_{\mathrm{i}}] \wedge [l_{\mathrm{k}} \notin h(x_{\mathrm{i}})]) \tag{4.10}$$

  Where TN_k is the true negative rate for the k^{th} label.

- False Negative Rate: For a given label, the rate at which that label is incorrectly absent from the predicted labels, averaged over the corpus:

$$FN_{\mathrm{k}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}([l_{\mathrm{k}} \in Y_{\mathrm{i}}] \wedge [l_{\mathrm{k}} \notin h(x_{\mathrm{i}})]) \tag{4.11}$$

  Where FN_k is the false negative rate for the k^{th} label.

Manual inspection of the classifier predictions is also useful for qualitative assessment: we expect good models to predict keywords that are closely related to the subject matter of the abstract, even if those keywords are not among the listed keywords for that paper. All evaluations are done on the test datasets.

4.4 Conclusion

The various word embedding methods are coupled with multi-layer neural classifiers with various different configurations in this study. Both domain prediction and keyword prediction tasks are investigated. The classifier hyper-parameters explored in this study are the number of hidden layers, and the number of neurons per layer. Classifier performance is evaluated with varying degrees of pruning of the infrequently-occurring target labels. Further dimensions explored in this study include: data separation strategy (journal-specific classifiers vs. global classifiers), optimisation strategy, and data splitting strategy (train/validation/test with proportionate representation vs. 6-fold cross-validation).

Chapter 5

Results and Discussion

The results of this study are grouped first by dataset, then by classification task and model architecture, and then an analysis of the impact of keyword frequency on classifier performance per dataset . The additional experiments performed with the BERT Extra model are presented thereafter, and the section is then concluded with commentary on the insights from the experimental results. All metrics presented in this section are obtained using the respective test dataset, with the exception of Figures 5.4a, 5.4b, 5.4c, 5.15a, 5.15b,and 5.15c, which are generated using the corresponding validation datsaset.

5.1 Nature Dataset

5.1.1 Domain Prediction

This section presents the results of the domain prediction task using the Nature dataset.

Bag-of-words Model Results:

The results for the bag-of-words models are presented in Table 5.1.

	Hidden Layers	Contracting	Accuracy
Model 1	**5**	**Yes**	**0.8786**
Model 2	10	No	0.2837

Table 5.1: Results of domain prediction experiments using the bag-of-words models with the Nature dataset (4 labels)

The **BOW** architecture performs best with the smaller configuration *i.e.* 5 hidden layers, with decreasing number of neurons from layer to layer. With this configuration, the performance in domain prediction is surprisingly high, with an average test accuracy of 87.86% for the best model. Since the **BOW** representation only encodes the identity of the words appearing in each text, these results indicate that the words used in abstracts from each journal are sufficiently distributed across the vocabulary to allow for good accuracy in domain prediction. It must also be noted that the input

representations are necessarily very sparse vectors and the first layer of the classification network provides a weight for each word in the vocabulary. It is speculated that the high-dimensionality of the input allows the network to learn a manifold that is effective in separating samples over a small number of labels. It must also be noted that the **BOW** model performs significantly worse when 10 non-contracting hidden layers are used, which shows that the **BOW** representation is not robust under these different circumstances.

Word2Vec Model Results:

The results for the Word2Vec and Word2Vec Extra models are presented in Table 5.2.

	Hidden Layers	**Contracting**	**Accuracy**
Word2Vec			
Model 1	4	Yes	0.8965
Model 2	**8**	**No**	**0.8992**
Word2Vec Extra			
Model 1	4	Yes	0.8901
Model 2	**8**	**No**	**0.8992**

Table 5.2: Results of domain prediction experiments using the Word2Vec models with the Nature dataset (4 labels)

Both **Word2Vec** and **Word2Vec Extra** model architectures perform best with the larger configuration *i.e.* 8 hidden layers, with a constant number of neurons from layer to layer. With this configuration, both **Word2Vec** and **Word2Vec Extra** achieved the same result in domain prediction, with an average test accuracy of 89.92%. The **Word2Vec** model performs marginally better than the **Word2Vec Extra** model when 4 contracting hidden layers are used. Both **Word2Vec** and **Word2Vec Extra** perform better than the best-performing **BOW** model. The tight distribution of accuracy measures observed in these experiments demonstrate that the Word2Vec-based models are more robust than the **BOW** models for the domain prediction task.

Transformer Model Results:

The results for the transformer models are presented in Table 5.3.

The **transformer** models perform similarly to the Word2Vec-based models in the domain prediction task, although the best **SciBERT** model achieves the highest accuracy overall (91.07%). The number and configuration of hidden layers that produce the best results differs for each of the **transformer** models. The **OpenAI-GPT** model shows the lowest performance (89.14% - less than the **Word2Vec** model), followed by **BERT** (90.03%) and **BERT Extra** (90.94%). For this dataset, the **SciBERT** and **BERT Extra** result indicate that a scientific pre-training corpus boosts model performance. The **BERT Extra** model shows better performance than **BERT** but does not perform as well as **SciBERT**, which is likely due to the relatively small size of the scientific pre-training dataset.

	Hidden Layers	**Contracting**	**Accuracy**
BERT			
Model 1	2	Yes	0.8940
Model 2	2	No	0.8992
Model 3	4	Yes	0.8748
Model 4	4	No	0.8927
Model 5	**0**	**-**	**0.9003**
BERT Extra			
Model 1	2	Yes	0.9054
Model 2	**2**	**No**	**0.9094**
Model 3	4	Yes	0.8903
Model 4	4	No	0.8927
Model 5	0	-	0.8901
SciBERT			
Model 1	**2**	**Yes**	**0.9107**
Model 2	2	No	0.9067
Model 3	4	Yes	0.8991
Model 4	4	No	0.9029
Model 5	0	-	0.9080
OpenAI-GPT			
Model 1	2	Yes	0.8827
Model 2	2	No	0.8825
Model 3	4	Yes	0.8865
Model 4	**4**	**No**	**0.8914**
Model 5	0	-	0.8776

Table 5.3: Results of domain prediction experiments using the transformer models with the Nature dataset (4 labels)

Model comparison

A Summary of the results from the best-performing models is presented in Table 5.4.

	Hidden Layers	Contracting	Accuracy
Bag-of-words	5	Yes	0.8786
Word2Vec	4	Yes	0.8901
BERT	0	-	0.9003
BERT Extra	2	No	0.9094
SciBERT	**2**	**Yes**	**0.9107**
OpenAI-GPT	4	No	0.8914

Table 5.4: Summary of results of the best-performing models in the domain prediction experiments with the Nature dataset (4 labels)

The relative ordering of models in terms of performance is as expected. It should be noted that the more complex architectures of the **transformers** and **Word2Vec** did not greatly improve on the performance in this task. This is especially true of the **transformers**, which have vastly more parameters than both **BOW** and **Word2Vec** models (Section 4.2.4).

5.1.2 Keyword Prediction

This section presents the results of the keyword prediction task using the Nature dataset. All metrics presented in this section (except the *Accuracy* metric) were obtained by application of a threshold value that maximses the *F1 Score* for the model on the validation dataset. The *Accuracy* metric corresponds to the *top 5 label accuracy* (Section 4.3.7), which is independent of any threshold.

Bag-of-words Model Results:

The results for the bag-of-words models are presented in Table 5.5.

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	**0.0403**	**0.1607**	**0.0644**	**0.0353**	**0.0**	**0.0915**
Model 2	0.0535	0.1135	0.0727	0.0412	0.0013	0.0903
499 Labels						
Model 1	**0.0903**	**0.2481**	**0.1324**	**0.0738**	**0.0**	**0.2114**
Model 2	0.0446	0.0891	0.0594	0.0342	0.0	0.0891
300 Labels						
Model 1	**0.1484**	**0.3180**	**0.2024**	**0.1224**	**0.0014**	**0.3203**
Model 2	0.0442	0.1608	0.0694	0.0391	0.0	0.1058

Table 5.5: Results of keyword prediction experiments using the bag-of-words models with the Nature dataset with 1234, 499, and 300 labels

In the keyword prediction task, the **BOW** architecture also performs best with the smaller configuration. On this task, the limitations of the **BOW** representation are more apparent. Model performance improves significantly when the number of labels is reduced. Neither of the models is able to produce sensible outputs over 1234 labels, but performance improves significantly over 499 labels, with the best performance over 300 labels. These results are to be expected because the sparsity of the target vectors at 1234 labels provides a small learning signal. However, the performance with 300 labels is surprisingly good, considering how little information is contained in the input representation.

Word2Vec Model Results:

The results for the Word2Vec models are presented in Table 5.6.

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	**0.1326**	**0.2534**	**0.1741**	**0.0988**	**0.0**	**0.2598**
Model 2	0.0392	0.1280	0.0600	0.0341	0.0	0.0812
499 Labels						
Model 1	**0.2138**	**0.3526**	**0.2662**	**0.1618**	**0.0013**	**0.3866**
Model 2	0.0566	0.0846	0.0679	0.0431	0.0013	0.0914
300 Labels						
Model 1	**0.2332**	**0.3748**	**0.2875**	**0.1799**	**0.0095**	**0.4490**
Model 2	0.0488	0.1319	0.0712	0.0422	0.0	0.1044

Table 5.6: Results of keyword prediction experiments using the Word2Vec models with the Nature dataset with 1234, 499, and 300 labels

The best-performing **Word2Vec** model over all target sets is that with the smaller model configuration *i.e.* 4 hidden layers with reducing number of neurons per layer. The best-performing **Word2Vec** model improves significantly on the best-performing **BOW** in all three cases.

Word2Vec Extra Model Results:

The results for the Word2Vec models are presented in Table 5.7.

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	**0.1328**	**0.2515**	**0.1738**	**0.0988**	**0.0**	**0.2600**
Model 2	0.0677	0.0710	0.0693	0.0408	0.0013	0.0882
499 Labels						
Model 1	**0.2190**	**0.3384**	**0.2659**	**0.1617**	**0.0013**	**0.3876**
Model 2	0.0565	0.0846	0.0678	0.0431	0.0013	0.0914
300 Labels						
Model 1	**0.2400**	**0.3619**	**0.2886**	**0.1824**	**0.0149**	**0.4428**
Model 2	0.0487	0.1339	0.0714	0.0422	0.0	0.1044

Table 5.7: Results of keyword prediction experiments using the Word2Vec Extra models with the Nature dataset with 1234, 499, and 300 labels

Just as in the case of **Word2Vec**, the best-performing **Word2Vec Extra** model over all target sets is that with the smaller model configuration *i.e.* 4 hidden layers with reducing number of neurons per layer. In terms of the performance achieved, the **Word2Vec Extra** model performs very similarly to the **Word2Vec** model, indicating that the additional pre-training carried out to produce the **Word2Vec Extra** model was not useful for the tasks at hand.

Transformer Model Results:

BERT

The results for the **BERT** are presented in Table 5.8.

The **BERT** model performs best in this task with the simplest architecture *i.e.* 0 hidden layers. The **BERT** model outperforms the **Word2Vec** model in each case, with the most significant performance improvement being shown in the experiments with 300 labels.

BERT Extra

The results for the **BERT Extra** are presented in Table 5.9.

As it was with **BERT**, the best results for **BERT Extra** are obtained with the simplest architecture. Notably, the **BERT Extra** model improves on the performance of **BERT** with all three target sets. This improvement is most pronounced in experiments with 1234 labels.

SciBERT

The results for the **SciBERT** are presented in Table 5.10.

The **SciBERT** model performs best with the simplest architecture with 1234 labels and with 499 labels, in which case the model performance is the best of the lot. In the case of 300 labels, the best performance is obtained with 2 non-contracting hidden layers and **SciBERT** is not able to perform as well as either **BERT** or **BERT Extra**.

OpenAI-GPT

The results for the **OpenAI-GPT** are presented in Table 5.11.

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	0.1918	0.2857	0.2295	0.1420	0.0102	0.3086
Model 2	0.2123	0.2509	0.2300	0.1425	0.0064	0.3211
Model 3	0.0481	0.0949	0.0638	0.0392	0.0	0.0858
Model 4	0.0882	0.2201	0.1259	0.0726	0.0	0.1870
Model 5	**0.2292**	**0.3179**	**0.2663**	**0.1625**	**0.0051**	**0.3707**
499 Labels						
Model 1	0.0426	0.1113	0.0616	0.0358	0.0	0.0900
Model 2	0.2441	0.2898	0.2650	0.1670	0.0181	0.3725
Model 3	0.0454	0.0762	0.0569	0.0360	0.0013	0.0783
Model 4	0.1900	0.2911	0.2300	0.1388	0.0026	0.3400
Model 5	**0.2918**	**0.3875**	**0.3329**	**0.2110**	**0.0181**	**0.4586**
300 Labels						
Model 1	0.2616	0.4900	0.3411	0.2244	0.0313	0.5279
Model 2	0.3362	0.4598	0.3884	0.2704	0.0516	0.5794
Model 3	0.0447	0.1629	0.0701	0.0395	0.0	0.1044
Model 4	0.0553	0.1256	0.0768	0.0448	0.0027	0.1007
Model 5	**0.4943**	**0.5942**	**0.5400**	**0.4133**	**0.1440**	**0.7363**

Table 5.8: Results of keyword prediction experiments using the BERT model with the Nature dataset with 1234, 499, and 300 labels

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	0.2074	0.2756	0.2367	0.1442	0.0051	0.3228
Model 2	0.2192	0.2797	0.2458	0.1502	0.0063	0.3216
Model 3	0.0516	0.1147	0.0712	0.0416	0.0	0.0858
Model 4	0.0430	0.1230	0.0637	0.0349	0.0	0.0900
Model 5	**0.2726**	**0.3529**	**0.3076**	**0.1891**	**0.0128**	**0.4037**
499 Labels						
Model 1	0.2022	0.3843	0.2650	0.1606	0.0052	0.3800
Model 2	0.2239	0.2766	0.2475	0.1547	0.0078	0.3679
Model 3	0.0452	0.1134	0.0646	0.0361	0.0	0.0980
Model 4	0.0508	0.1361	0.0739	0.0432	0.0	0.0951
Model 5	**0.2965**	**0.3805**	**0.3332**	**0.2137**	**0.0233**	**0.4658**
300 Labels						
Model 1	0.3211	0.4784	0.3843	0.2685	0.0516	0.5735
Model 2	0.3788	0.4725	0.4205	0.3004	0.0747	0.6033
Model 3	0.0484	0.1534	0.0736	0.0417	0.0	0.1109
Model 4	0.0486	0.1483	0.0732	0.0426	0.0	0.1128
Model 5	**0.5535**	**0.5623**	**0.5579**	**0.4433**	**0.1848**	**0.7441**

Table 5.9: Results of keyword prediction experiments using the BERT Extra models with the Nature dataset with 1234, 499, and 300 labels

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	0.2199	0.2712	0.2429	0.1497	0.0077	0.3463
Model 2	0.2021	0.2916	0.2387	0.1469	0.0077	0.3420
Model 3	0.0407	0.1512	0.0642	0.0365	0.0	0.0894
Model 4	0.0935	0.2541	0.1367	0.0700	0.0013	0.1782
Model 5	**0.2850**	**0.3819**	**0.3264**	**0.2028**	**0.0089**	**0.4299**
499 Labels						
Model 1	0.2250	0.3050	0.2590	0.1613	0.0091	0.3943
Model 2	0.2308	0.3560	0.2800	0.1758	0.0104	0.3926
Model 3	0.0360	0.1948	0.0608	0.0331	0.0	0.0953
Model 4	0.2250	0.3053	0.2591	0.1652	0.0091	0.3888
Model 5	**0.3203**	**0.3295**	**0.3248**	**0.2094**	**0.0194**	**0.4680**
300 Labels						
Model 1	0.3522	0.5422	0.4270	0.3026	0.0625	0.6191
Model 2	**0.4211**	**0.5310**	**0.4697**	**0.3401**	**0.0829**	**0.6656**
Model 3	0.2688	0.4893	0.3469	0.2309	0.0285	0.5290
Model 4	0.3490	0.4616	0.3975	0.2674	0.0380	0.5567
Model 5	0.2686	0.4423	0.3343	0.2185	0.0190	0.5105

Table 5.10: Results of keyword prediction experiments using the SciBERT models with the Nature dataset with 1234, 499, and 300 labels

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Model 1	0.1984	0.2659	0.2273	0.1390	0.0064	0.3179
Model 2	0.1952	0.2914	0.2338	0.1400	0.0038	0.3236
Model 3	0.0363	0.1519	0.0587	0.0321	0.0	0.0707
Model 4	0.0481	0.1139	0.0676	0.0400	0.0013	0.0817
Model 5	**0.2584**	**0.3720**	**0.3050**	**0.1865**	**0.0102**	**0.4133**
499 Labels						
Model 1	0.2641	0.3293	0.2931	0.1843	0.0130	0.4208
Model 2	0.2483	0.3034	0.2731	0.1725	0.0130	0.3981
Model 3	0.0484	0.1339	0.0711	0.0400	0.0	0.0986
Model 4	0.1774	0.2937	0.2212	0.1314	0.0	0.3294
Model 5	**0.3270**	**0.3429**	**0.3347**	**0.2212**	**0.0311**	**0.4703**
300 Labels						
Model 1	0.2504	0.4503	0.3218	0.2019	0.0122	0.4818
Model 2	0.2690	0.3365	0.2990	0.1927	0.0177	0.4576
Model 3	0.0508	0.1259	0.0724	0.0394	0.0	0.1014
Model 4	0.0349	0.2679	0.0617	0.0328	0.0	0.1014
Model 5	**0.3225**	**0.4124**	**0.3620**	**0.2421**	**0.0367**	**0.4999**

Table 5.11: Results of keyword prediction experiments using the OpenAI-GPT models with the Nature dataset with 1234, 499, and 300 labels

Once again, the best performing architecture is the simplest in the case of **OpenAI-GPT**, for all three target sets. The **OpenAI-GPT** model performs better than both **BERT** and **BERT Extra** with 1234 labels, performs better than all other models with 499 labels, but performs worst of all the **transformers** with 300 labels.

Model comparison

A Summary of the results from the best-performing models is presented in Table 5.12.

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
1234 Labels						
Bag-of-words	0.0403	0.1607	0.0644	0.0353	0.0	0.0915
Word2Vec	0.1326	0.2534	0.1741	0.0988	0.0	0.2598
Word2Vec Extra	0.1328	0.2515	0.1738	0.0988	0.0	0.2600
BERT	0.2292	0.3179	0.2663	0.1625	0.0051	0.3707
BERT Extra	0.2726	0.3529	0.3076	0.1891	0.0128	0.4037
SciBERT	**0.2850**	**0.3819**	**0.3264**	**0.2028**	**0.0089**	**0.4299**
OpenAI-GPT	0.2584	0.3720	0.3050	0.1865	0.0102	0.4133
499 Labels						
Bag-of-words	0.0903	0.2481	0.1324	0.0738	0.0	0.2114
Word2Vec	0.2138	0.3526	0.2662	0.1618	0.0013	0.3866
Word2Vec Extra	0.2190	0.3384	0.2659	0.1617	0.0013	0.3876
BERT	0.2918	0.3875	0.3329	0.2110	0.0181	0.4586
BERT Extra	0.2965	0.3805	0.3332	0.2137	0.0233	0.4658
SciBERT	0.3203	0.3295	0.3248	0.2094	0.0194	0.4680
OpenAI-GPT	**0.3270**	**0.3429**	**0.3347**	**0.2212**	**0.0311**	**0.4703**
300 Labels						
Bag-of-words	0.1484	0.3180	0.2024	0.1224	0.0014	0.3203
Word2Vec	0.2332	0.3748	0.2875	0.1799	0.0095	0.4490
Word2Vec Extra	0.2400	0.3619	0.2886	0.1824	0.0149	0.4428
BERT	0.4943	0.5942	0.5400	0.4133	0.1440	0.7363
BERT Extra	**0.5535**	**0.5623**	**0.5579**	**0.4433**	**0.1848**	**0.7441**
SciBERT	0.4211	0.5310	0.4697	0.3401	0.0829	0.6656
OpenAI-GPT	0.3225	0.4124	0.3620	0.2421	0.0367	0.4999

Table 5.12: Summary of results of the best-performing models in the keyword prediction experiments with the Nature dataset with 1234, 499, and 300 labels

In the keyword prediction task, the **BOW** model performs the worst, followed by the **Word2Vec** models, with the **transformer** models performing the best. For each target set, a different model showed the best performance: **SciBERT** (1234 labels), **OpenAI-GPT** (499 labels), and **BERT Extra** (300 labels). The value of domain-specific pre-training is demonstrated by the generally strong performance of **SciBERT**, and by the performance improvement of **BERT Extra** over **BERT**. Additional pre-training of the

Word2Vec model did not yield any significant change in model performance.
The performance summary for each model on each target set are also represented in Figures 5.1, 5.2, and 5.3.

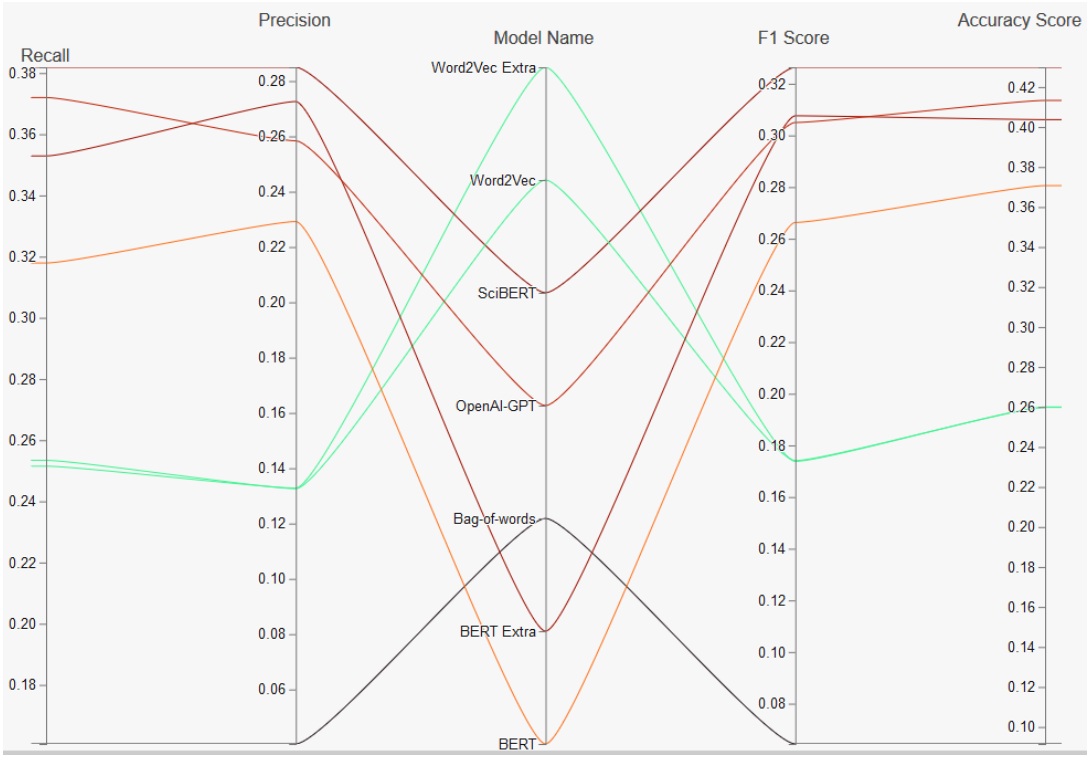


Figure 5.1: Summary of results of the best-performing models in the keyword prediction experiments with the Nature dataset with 1234 labels

The threshold traversal graphs for the best-performing models on each target set are represented in Figure 5.4. These graphs show Precision, Recall and F1 scores as a function of threshold, using the respective validation set. The threshold corresponding to the highest F1 score is applied to the test set. The optimal threshold for each of the models is presented in Table 5.13

Model	**N_{Labels}**	**Optimal Threshold**
SciBERT	1234	0.12
OpenAI-GPT	499	0.20
BERT Extra	300	0.16

Table 5.13: Optimal threshold values for the best-performing models in keyword prediction experiments with the Nature dataset, with 1234 499, and 300 labels

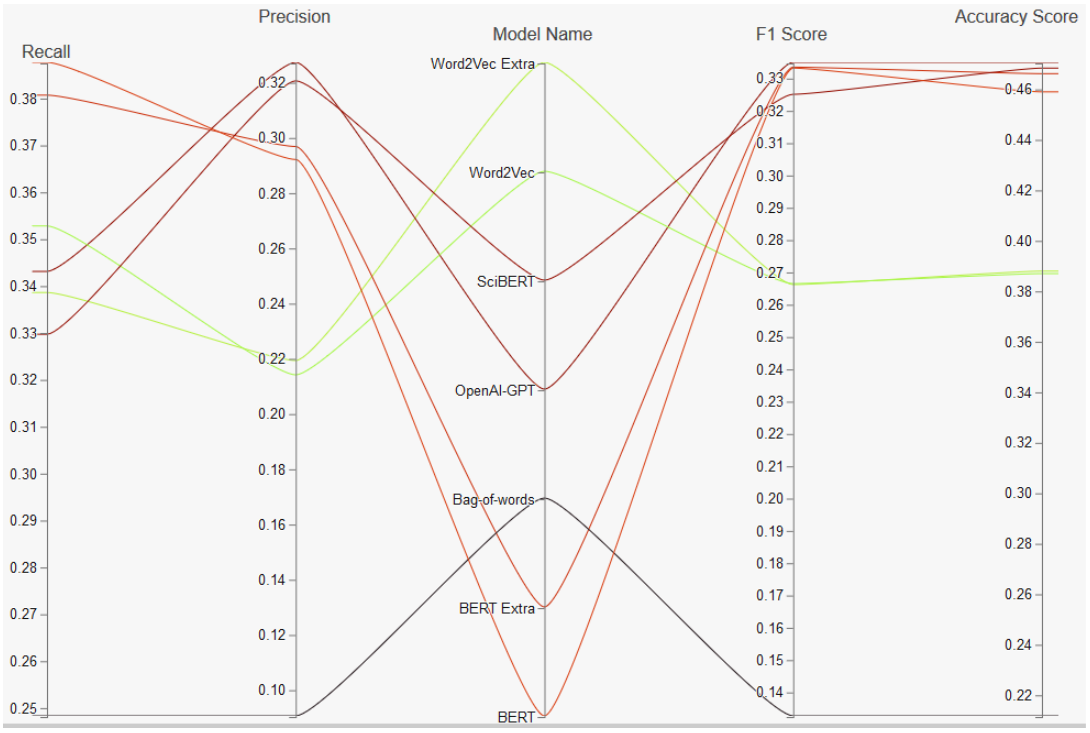


Figure 5.2: Summary of results of the best-performing models in the keyword prediction experiments with the Nature dataset with 499 labels

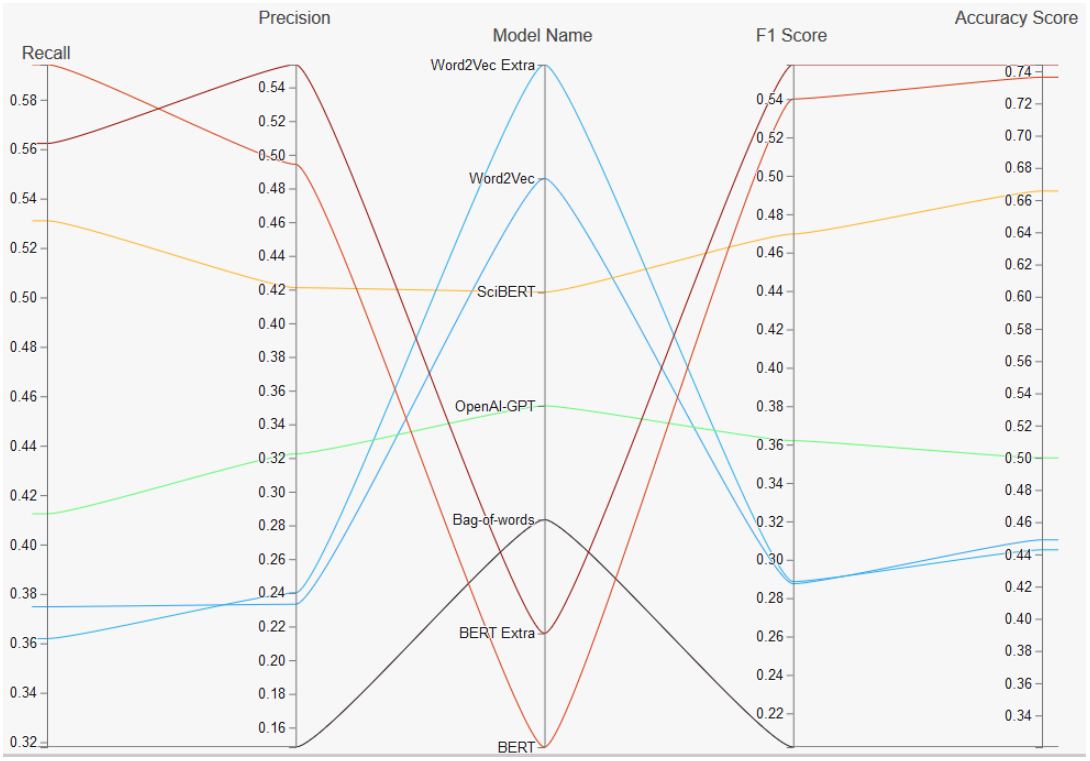
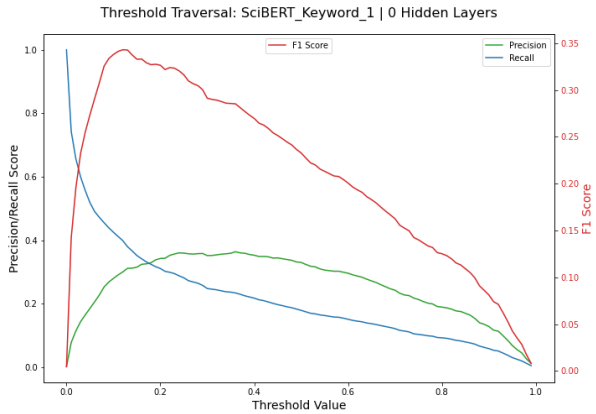
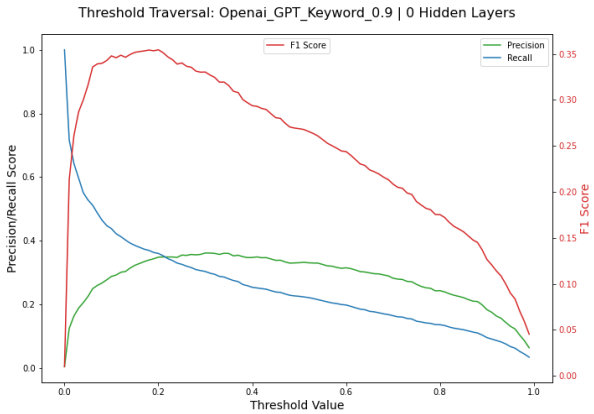


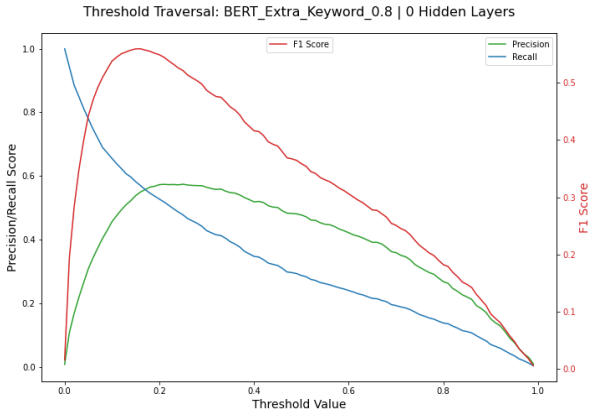
Figure 5.3: Summary of results of the best-performing models in the keyword prediction experiments with the Nature dataset with 300 labels



(a) 1234 Labels: SciBERT



(b) 499 Labels: OpenAI-GPT

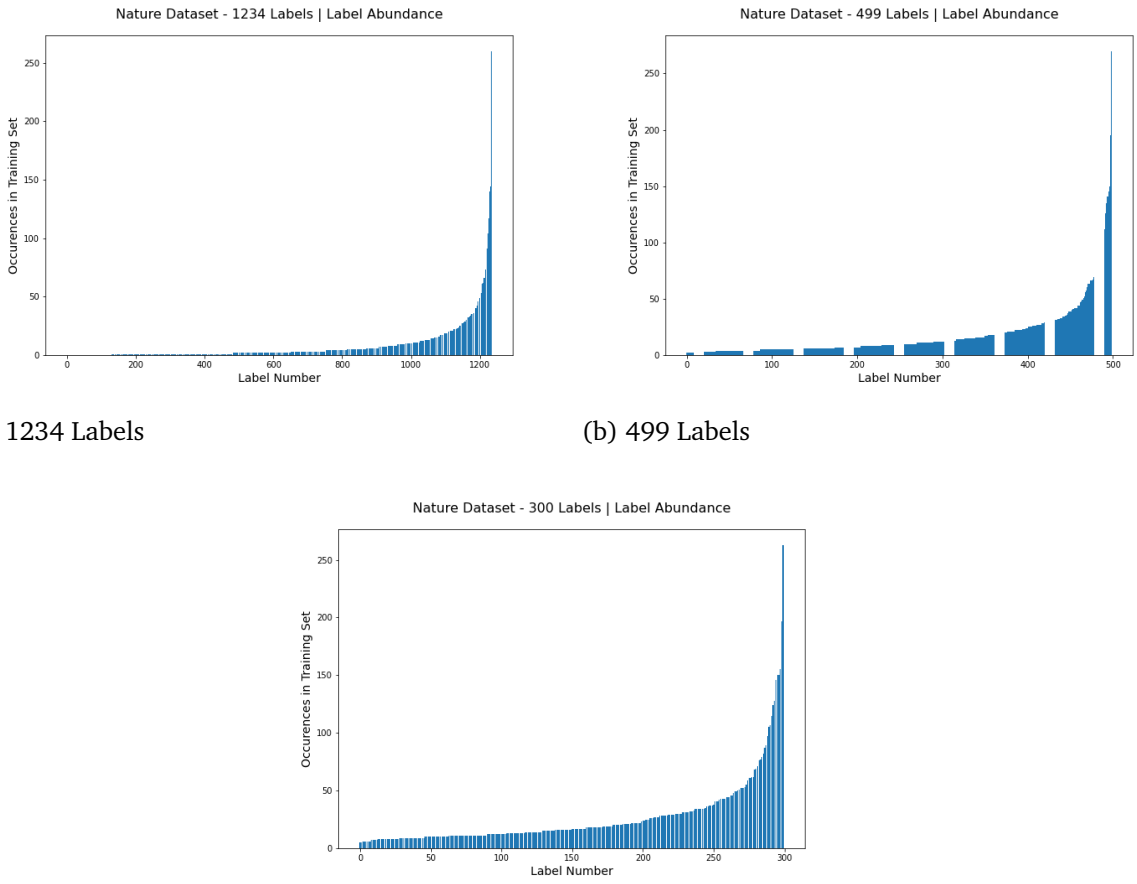


(c) 300 Labels: BERT Extra

Figure 5.4: Precision, recall, and F_1 scores vs. threshold for the best-performing models in the keyword prediction experiments with the Nature dataset with 1234, 499, and 300 labels

5.1.3 Analysis of Keyword Frequency on Classification Accuracy

It is to be expected that frequently-occurring keywords are more likely to be predicted correctly than those that occur infrequently. The results presented in this chapter so far are high-level performance indicators, but do not give insight into the distribution of performance over each of the labels. This section presents the classifier performance for each label, using the best-performing models for the Naure dataset. The abundance of labels in the training set of the Nature dateset is presented in Figure 5.5.



(a) 1234 Labels

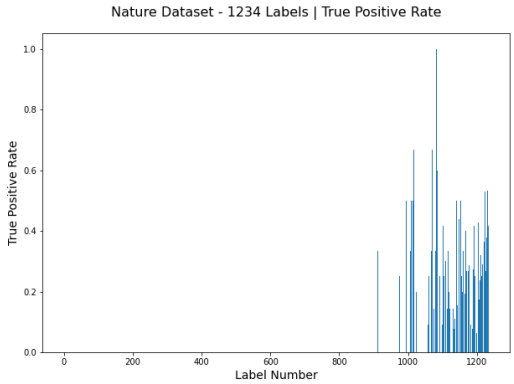
(b) 499 Labels

(c) 300 Labels

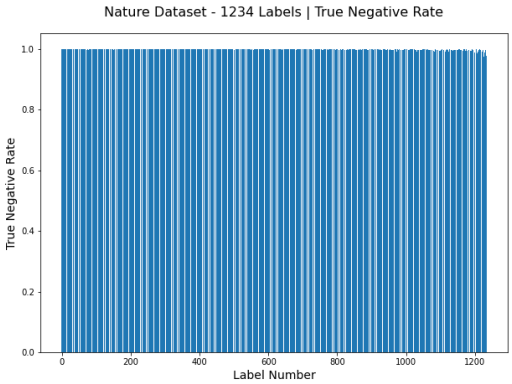
Figure 5.5: Distribution of unique keyword terms - Nature dataset

The True Positive, True Negative, False Positive, and False Negative rates (section 4.3.7) per label are presented in Figures 5.6, 5.7, 5.8, 5.9, 5.10, and 5.11.

Each abstract in the dataset has a small number of labels compared to the total number of labels that are possible *i.e.* target vectors are sparse. This means that for each sample, correct model predictions would necessitate a large number of *true negative* predictions. The more labels the model predicts that are not correct (*false positives*), the less likely a model is to be useful in keyword assignment. However, a *false positive* prediction does not necessarily mean that the label is thematically irrelevant, just that it is not among the assigned keywords. When labels assigned to an abstract do not

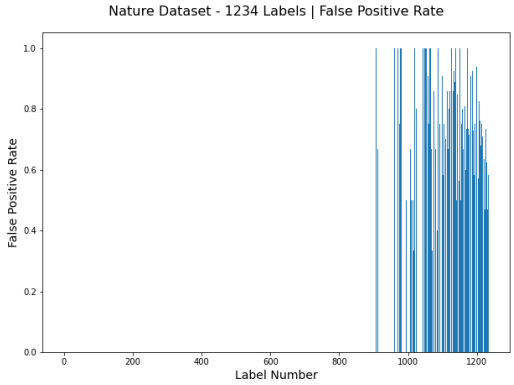


(a) True Positive Rate

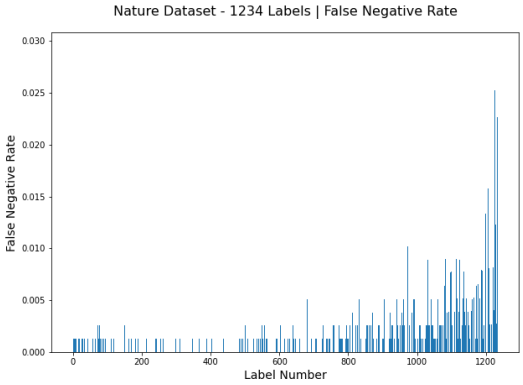


(b) True Negative Rate

Figure 5.6: True positive and true negative rates for the **SciBERT** model with the Nature dataset with 1234 labels

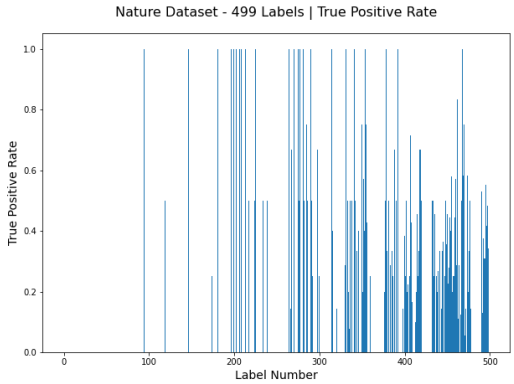


(a) False Positive Rate

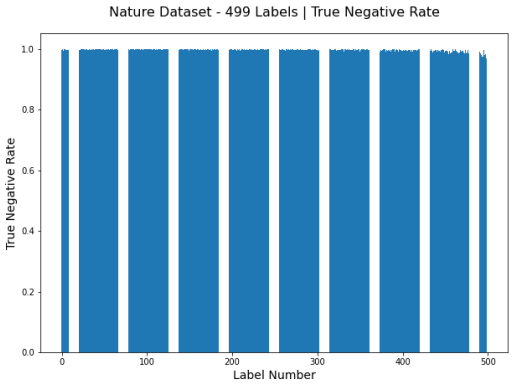


(b) False Negative Rate

Figure 5.7: False positive and false negative rates for the **SciBERT** model with the Nature dataset with 1234 labels

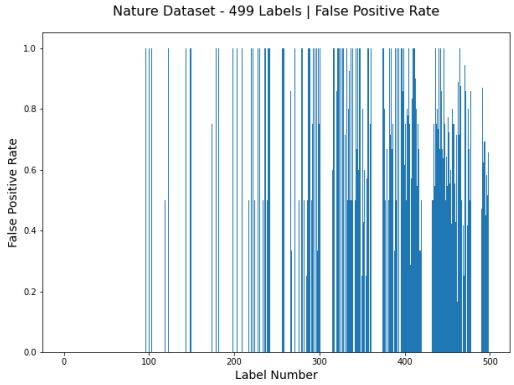


(a) True Positive Rate

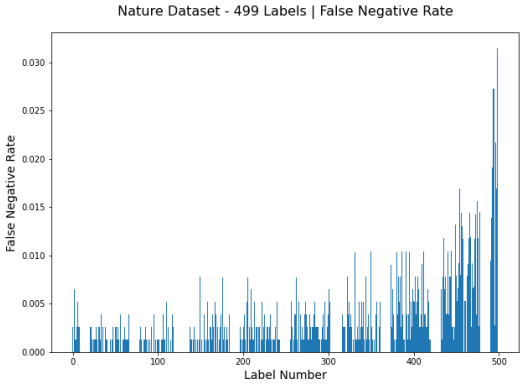


(b) True Negative Rate

Figure 5.8: True positive and true negative rates for the **OpenAI-GPT** model with the Nature dataset with 499 labels

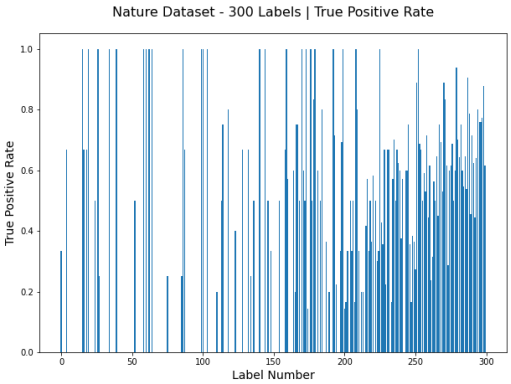


(a) False Positive Rate

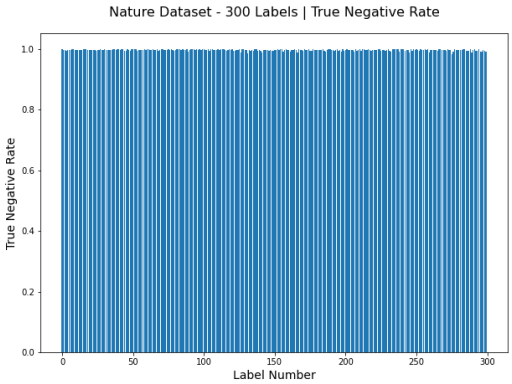


(b) False Negative Rate

Figure 5.9: False positive and false negative rates for the **OpenAI-GPT** model with the Nature dataset with 499 labels

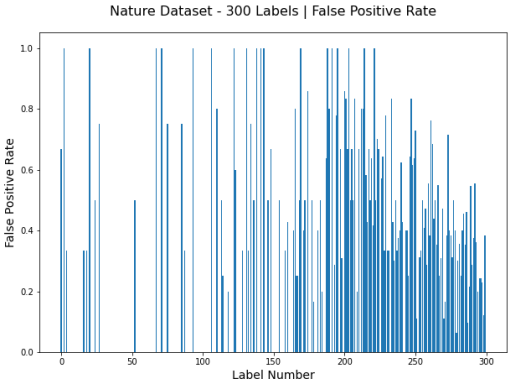


(a) True Positive Rate

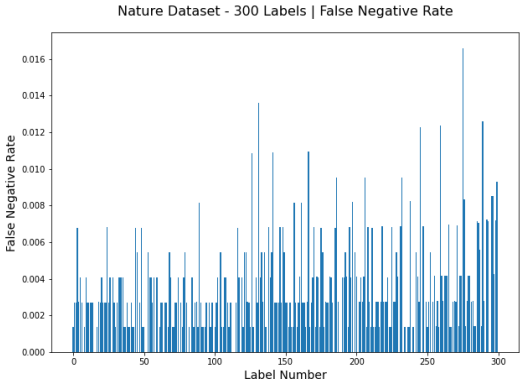


(b) True Negative Rate

Figure 5.10: True positive and true negative rates for the **BERT Extra** model with the Nature dataset with 300 labels



(a) False Positive Rate



(b) False Negative Rate

Figure 5.11: False positive and false negative rates for the **BERT Extra** model with the Nature dataset with 300 labels

appear in the model predictions (*false negatives*), then important categorizations may be missed. *False negatives* reduce model usefulness, but are less likely to be an issue than *false positives*. In general, the *true positive* rate for a label is analogous to model precision, and is the inverse of the *false positive* rate for the label. For example, of all the instances in which a given label is predicted by the model, if the label is predicted correctly 40% of the time then it must be predicted incorrectly the remaining 60%. The *false negative* rates are also the inverse of the *true negative* rates.

With 1234 labels, there are many labels with low abundance. The *true positive* and *false positive* rates for most of the low-abundance labels are both 0. This means that these labels are not being predicted by the model at all *i.e.* the model only learns to predict higher-frequency labels. For all labels, the *true negative* rate is high and *false negative* rate is low, which is to be expected with such sparse target vectors. The *true positive* rate is generally lower than the *false positive* rate, which is to be expected based on the performance metrics presented in Table 5.12 *i.e.* the model precision is less than 50%.

The same trends are observed with 499- and 300 labels, except that with less low-abundance labels there are less labels that are not predicted at all by the respective model. There are also relatively more *true positives* as the number of target labels decreases.

5.2 IEEE Dataset

The experiments conducted on the IEEE dataset make use of the best-performing architecture for each model on the Nature dataset. The configurations selected are as follows:

- BOW: 5 contracting hidden layers
- Word2Vec: 4 contracting hidden layers
- Transformers: 0 hidden layers

5.2.1 Domain Prediction

The results for each model are presented in Table 5.14.

	Hidden Layers	Contracting	Accuracy
Bag-of-words	5	Yes	0.9644
Word2Vec	4	Yes	0.9375
Word2Vec Extra	4	Yes	0.9454
BERT	0	-	0.9775
BERT Extra	0	-	0.9738
SciBERT	**0**	**-**	**0.9790**
OpenAI-GPT	4	No	0.9689

Table 5.14: Summary of results of the best-performing models in the domain prediction experiments with the IEEE dataset (2 labels)

All models perform better on the IEEE dataset than on the Nature dataset in domain prediction. This is to be expected because there are more samples in the IEEE dataset and half the amount of labels. The best-performing model is once again **SciBERT**, followed by **BERT**, **BERT Extra**, **OpenAI-GPT**, **BOW**, **Word2Vec Extra**, and lastly **Word2Vec**. Once again, the **BOW** model performs surprisingly well, out-performing both **Word2Vec** models and coming very close to the accuracy of the transformer models. These results reinforce the observation that the domain prediction task does not benefit very much from the added complexity of the **transformer** models.

5.2.2 Keyword Prediction

The results for each model are presented in Table 5.15.

Model	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
3174 Labels						
Bag-of-words	0.0695	0.1511	0.0952	0.0508	0.0	0.0619
Word2Vec	0.0626	0.2022	0.0956	0.0511	0.0	0.0619
Word2Vec Extra	0.0627	0.2022	0.0957	0.0511	0.0	0.0619
BERT	0.3790	0.3342	0.3552	0.2208	0.0	0.2746
BERT Extra	0.4116	0.2977	0.3455	0.2133	0.0	0.2674
SciBERT	**0.4334**	**0.3662**	**0.3970**	**0.2536**	**0.0**	**0.3014**
OpenAI-GPT	0.4411	0.3403	0.3842	0.2425	0.0008	0.2940
805 Labels						
Bag-of-words	0.0793	0.1165	0.0943	0.0502	0.0	0.0619
Word2Vec	0.2490	0.2994	0.2719	0.1607	0.0	0.2144
Word2Vec Extra	0.2463	0.2894	0.2661	0.1576	0.0	0.2094
BERT	0.4635	0.3948	0.4264	0.2782	0.0008	0.3427
BERT Extra	0.3547	0.3234	0.3383	0.2087	0.0	0.2740
SciBERT	**0.4734**	**0.4189**	**0.4445**	**0.2916**	**0.0008**	**0.3604**
OpenAI-GPT	0.4535	0.3918	0.4204	0.2732	0.0004	0.3411
467 Labels						
Bag-of-words	0.2664	0.3188	0.2902	0.1750	0.0	0.2479
Word2Vec	0.2505	0.3353	0.2868	0.1704	0.0	0.2441
Word2Vec Extra	0.2563	0.3252	0.2867	0.1702	0.0	0.2429
BERT	0.4992	0.4247	0.4589	0.3045	0.0019	0.3910
BERT Extra	**0.5133**	**0.4454**	**0.4770**	**0.3236**	**0.0060**	**0.4104**
SciBERT	0.4999	0.4573	0.4776	0.3213	0.0026	0.4109
OpenAI-GPT	0.4497	0.4199	0.4343	0.2845	0.0022	0.3713

Table 5.15: Summary of results of the best-performing models in the keyword prediction experiments with the IEEE dataset with 3174, 805, and 467 labels

The **BOW** model did not produce sensible outputs for target sets of both 3174 labels and 805 labels, but performed slightly better than the **Word2Vec** and **Word2Vec Extra** models with the target set of 467 labels. Both of the **Word2Vec** models performed very similarly to one another and produced similar results with 805 labels and 467 labels.

Neither of the **Word2Vec** models was able to produce sensible outputs with 3174 labels. Once again, all **transformer** models performed better than both **Word2Vec** models and the **BOW** model. The best-performing model with both 3174 labels and 805 labels is **SciBERT**, with **BERT Extra** showing the best results for 467 labels. The **OpenAI-GPT** model performed second-best with 3174- and 805 labels, but worst of all the transformers with 467 labels. The **BERT Extra** model only showed improved performance over **BERT** with 467 labels, which may be due to the fact that the pre-training dataset (Section 3.3.2) did not include texts from the domains (scientific fields) represented in the IEEE dataset. The performance of the **transformer** models in the keyword prediction task demonstrates the value of the greater model complexity: for more complex tasks with sparser learning signals, the context-capturing embeddings produced by **transformer** models perform significantly better than the simpler **Word2Vec** embeddings and simplest representation of **BOW**.
The performance summary for each model on each target set are also represented in Figures 5.12, 5.13, and 5.14.

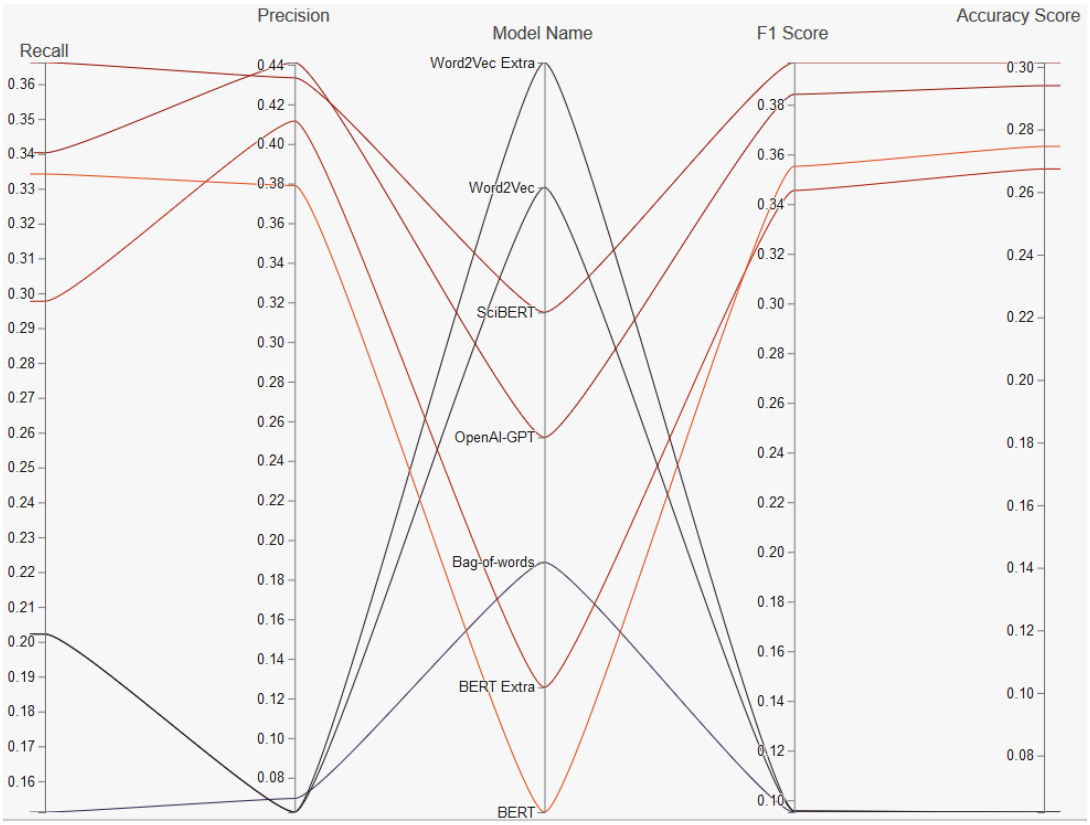


Figure 5.12: Summary of results of the best-performing models in the keyword prediction experiments with the IEEE dataset with 3174 labels

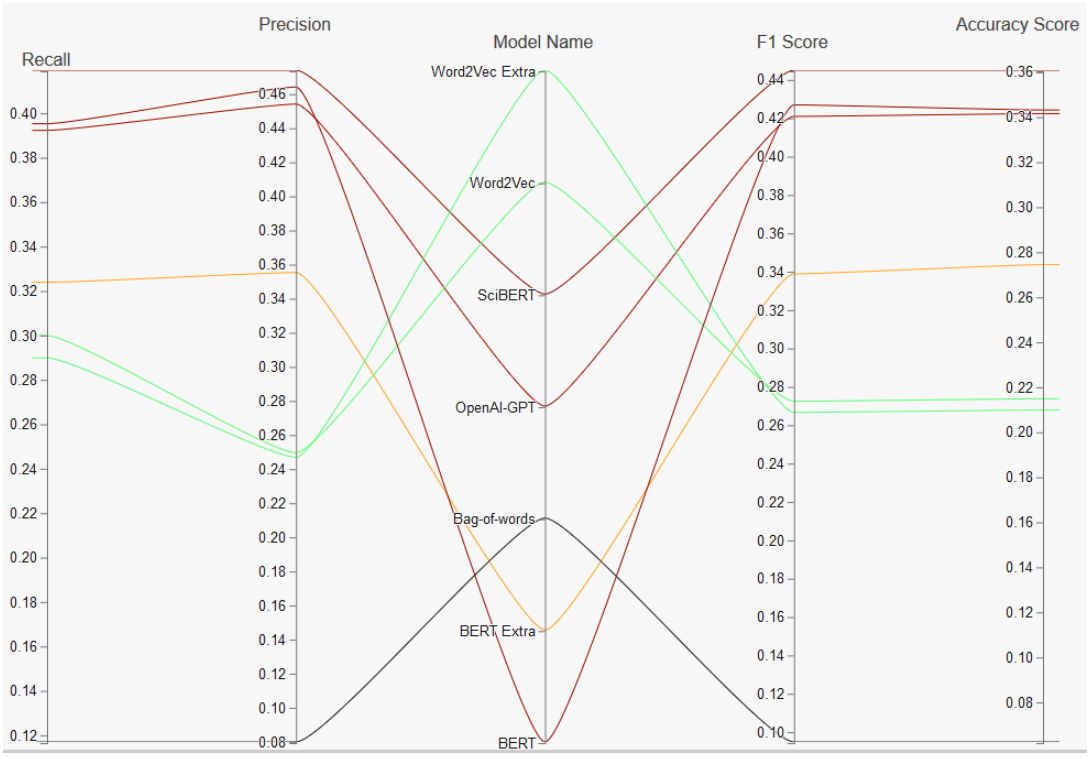


Figure 5.13: Summary of results of the best-performing models in the keyword prediction experiments with the IEEE dataset with 805 labels

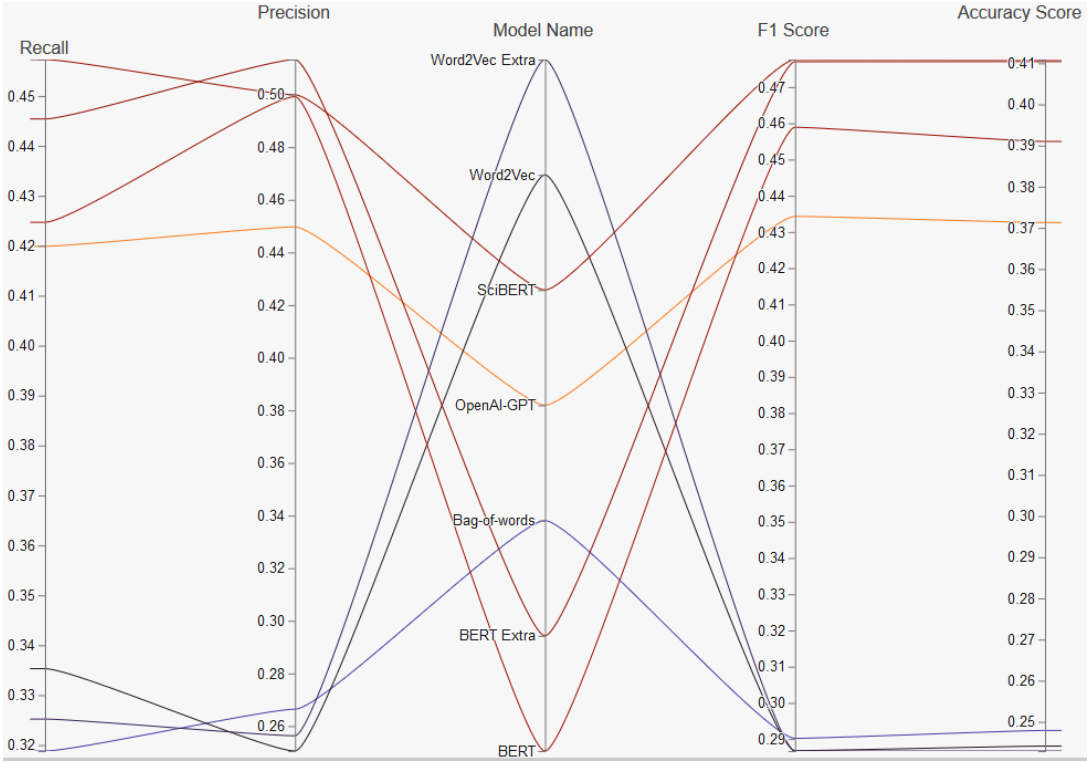
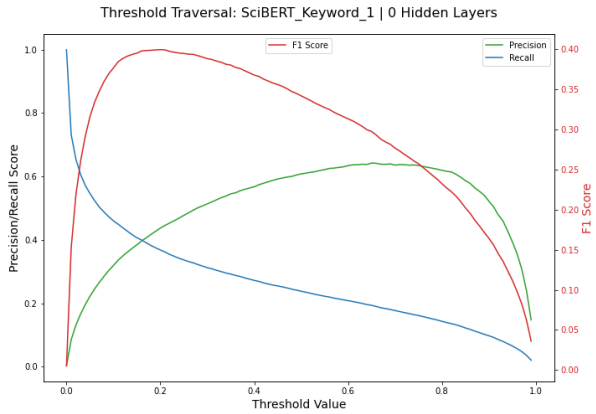
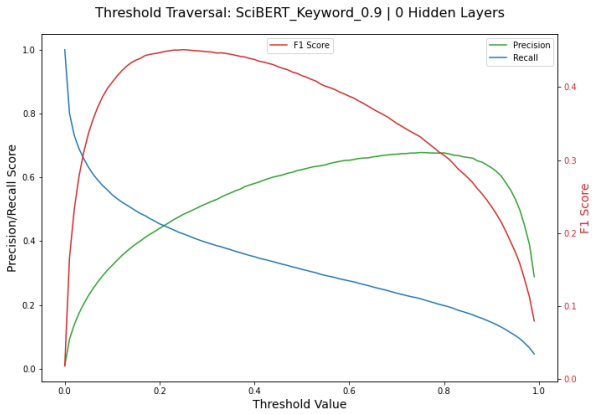


Figure 5.14: Summary of results of the best-performing models in the keyword prediction experiments with the IEEE dataset with 467 labels

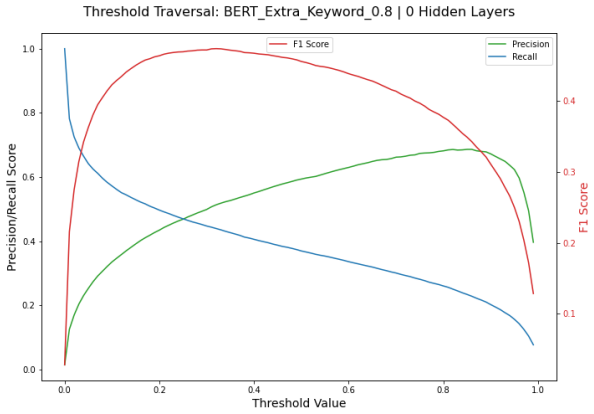
The threshold traversal graphs for the best-performing models on each target set are represented in Figure 5.15. The optimal thresholds for each of the models is presented in Table 5.16



(a) 3174 Labels: SciBERT



(b) 805 Labels: SciBERT



(c) 467 Labels: BERT Extra

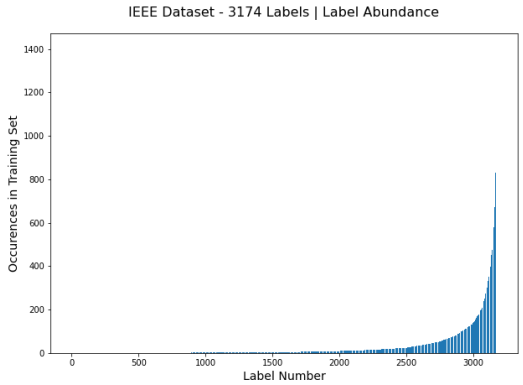
Figure 5.15: Precision, recall, and F_1 scores vs. threshold for the best-performing models in the keyword prediction experiments with the IEEE dataset with 3174, 805, and 467 labels

Model	N_{Labels}	Optimal Threshold
SciBERT	3174	0.20
SciBERT	805	0.25
BERT Extra	467	0.32

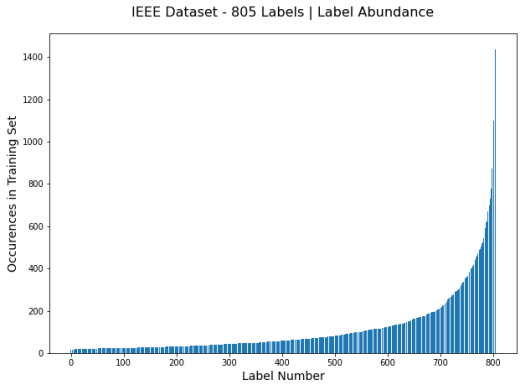
Table 5.16: Optimal threshold value for the best-performing model on the IEEE dataset with 3174, 805, and 467 labels

5.2.3 Analysis of Keyword Frequency on Classification Accuracy

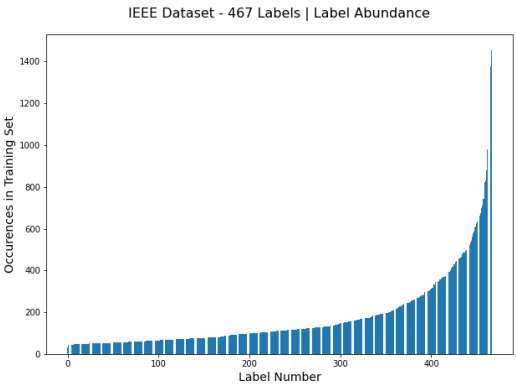
The abundance of labels in the training set of the IEEE dateset is presented in Figure 5.16.



(a) 3174 Labels



(b) 805 Labels

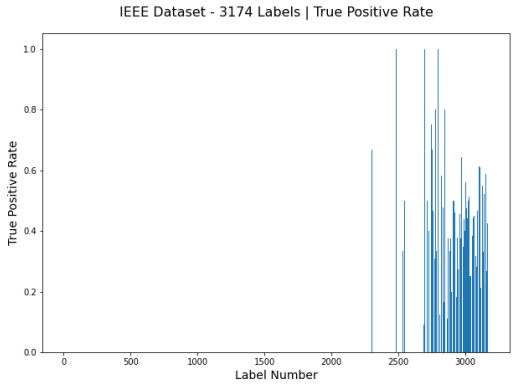


(c) 467 Labels

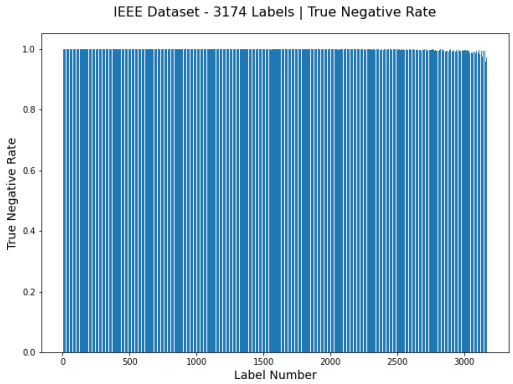
Figure 5.16: Distribution of unique keyword terms - IEEE dataset

The True Positive, True Negative, False Positive, and False Negative rates per label are presented in Figures 5.17, 5.18, 5.19, 5.20, 5.21, and 5.22.

The trends observed with the Nature dataset are observed again with the IEEE dataset. The *true positive* and *false positive* rates for most of the low-abundance labels are both 0. For all labels, the *true negative* rate is high and *false negative* rate is low. The *true positive* rate is generally lower than the *false positive* rate, but the *true positive* rate increases as the number of labels decreases from 3174 down to 467.

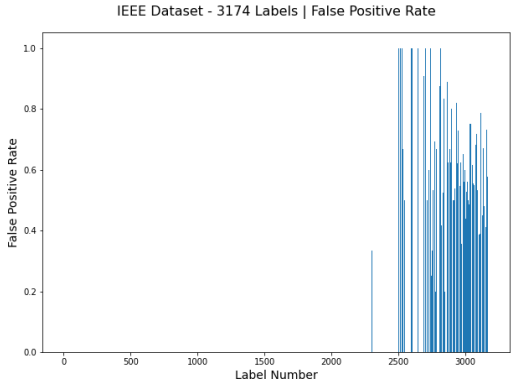


(a) True Positive Rate

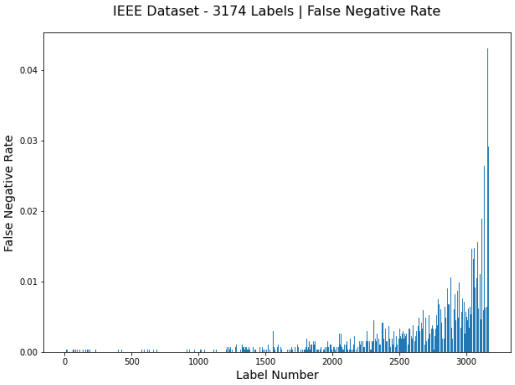


(b) True Negative Rate

Figure 5.17: True positive and true negative rates for the **SciBERT** model with the IEEE dataset with 3174 labels

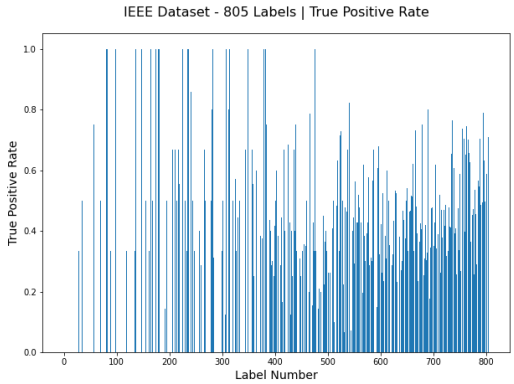


(a) False Positive Rate

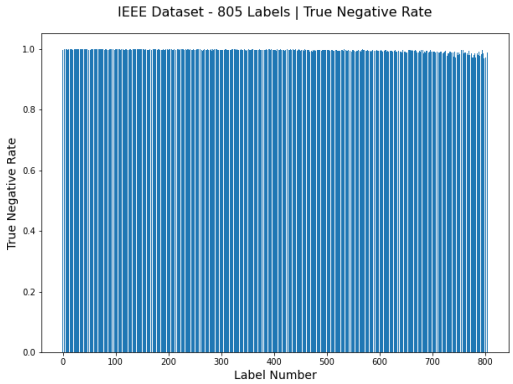


(b) False Negative Rate

Figure 5.18: False positive and false negative rates for the **SciBERT** model with the IEEE dataset with 3174 labels

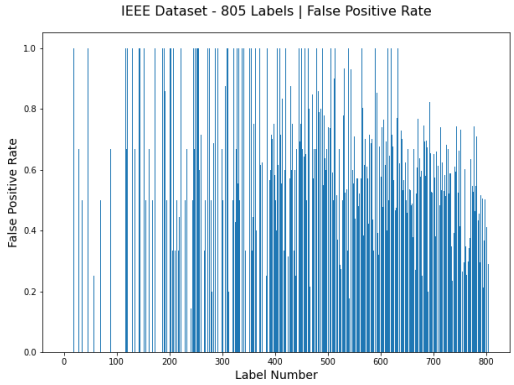


(a) True Positive Rate

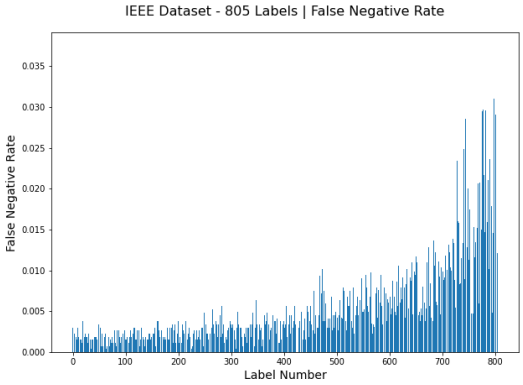


(b) True Negative Rate

Figure 5.19: True positive and true negative rates for the **SciBERT** model with the IEEE dataset with 805 labels

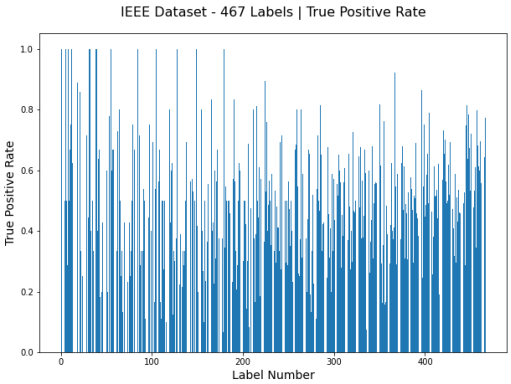


(a) False Positive Rate

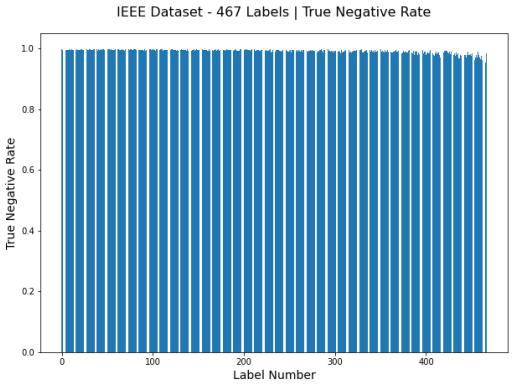


(b) False Negative Rate

Figure 5.20: False positive and false negative rates for the **SciBERT** model with the IEEE dataset with 805 labels

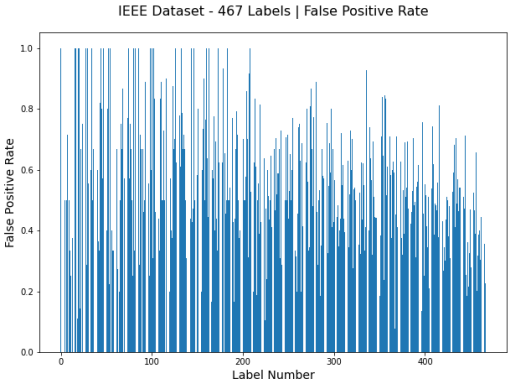


(a) True Positive Rate

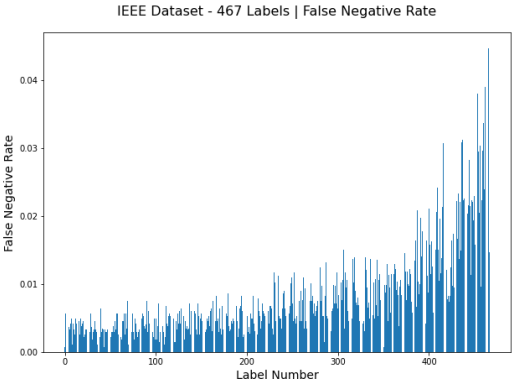


(b) True Negative Rate

Figure 5.21: True positive and true negative rates for the **BERT Extra** model with the IEEE dataset with 467 labels



(a) False Positive Rate



(b) False Negative Rate

Figure 5.22: False positive and false negative rates for the **BERT Extra** model with the IEEE dataset with 467 labels

5.3 Analysis of BERT Extra

This section presents the results of further experimentation with the **BERT Extra** model with 0 hidden layers.

5.3.1 Domain-level Keyword Prediction Results

Nature Dataset:

The results of training a separate model for each journal in the Nature dataset are summarised in Table 5.17, along with the corresponding results of the best-performing global model (trained over the whole dataset).

Domain	N_{Labels}	$N_{Samples}$	Precision	Recall	F1 Score	Accuracy
Domain-level Models						
Climate	137	1044	0.3646	0.4281	0.3938	0.5330
Medicine	754	1334	0.0542	0.0554	0.0548	0.0618
Materials	349	1480	0.3027	0.3129	0.3077	0.4098
Physics	229	1370	0.3488	0.4899	0.4075	0.5498
Global Model						
Climate	1234	1044	0.3353	0.4469	0.3832	0.4965
Medicine	1234	1334	0.1813	0.2006	0.1905	0.2497
Materials	1234	1480	0.2332	0.3107	0.2665	0.3761
Physics	1234	1370	0.4141	0.3916	0.4025	0.5132

Table 5.17: Performance comparison between domain-level classifiers and global BERT Extra classifiers with the Nature dataset

Training domain-level classifiers improved performance for all Nature journals except Nature Medicine. The domain-level model for the Medicine journal performed significantly lower than the global model and was not able to produce sensible outputs. This result indicates that the model is unable to be adequately trained if too few samples are provided, relative to the number of labels. This observation is supported by the fact that the global keyword classifier is able to perform significantly better on samples from the Medicine journal when trained along with samples from the other journals, even though the additional samples do not overlap greatly with the Medicine journal in terms of unique label occurrence (Figure 4.1a). If sufficient samples can be provided, then there is evidence to suggest that better performance can be obtained by training separate models for each domain, however this comes with a high overhead of computational cost.

IEEE Dataset:

The results of training a separate model for each journal in the IEEE dataset are summarised in Table 5.18, along with the corresponding results of the best-performing global model (trained over the whole dataset).

Domain	N_{Labels}	$N_{Samples}$	Precision	Recall	F1 Score	Accuracy
Domain-level Models						
Geo. Rem. Sen.	2546	11569	0.0968	0.1479	0.1170	0.0853
Pat. Ana. Mac. Int.	2253	6196	0.0995	0.1967	0.1321	0.0830
Global Model						
Geo. Rem. Sen.	2546	11569	0.4417	0.3270	0.3758	0.2923
Pat. Ana. Mac. Int.	2253	6196	0.3010	0.2888	0.2948	0.2209

Table 5.18: Performance comparison between domain-level and global-level BERT Extra classifiers with the IEEE dataset

The domain-level classifiers were not able to learn anything sensible, in contrast to the global model which achieved relatively good performance in both domains. It must be noted that there is significant overlap between the two domains in terms of the keywords represented in the dataset (4.2a). This means that separating the samples from each domain reduces the total number of samples for each of the overlapping keywords, which is likely why the domain-level classifiers were not successful. These findings indicate that it is preferable to combine data from different domains if there is sufficient overlap of unique labels, even though this increases the size (and therefore sparsity) of the target vectors.

5.3.2 IEEE Author Keyword Prediction Results

The results of training the **BERT Extra** model on the IEEE dataset using the Author keywords are presented in Table 5.19).

N_{Labels}	$N_{Samples}$	Precision	Recall	F1 Score	Accuracy
1802	10853	0.0848	0.0579	0.0688	0.0792

Table 5.19: Results of the keywords prediction experiment using BERT Extra with the IEEE dataset with 1802 Author keywords

The **BERT Extra** model is also not able to produce sensible outputs when trained on IEEE Author keywords. It is speculated that the author keywords focus on specifics of the research, to the point where the words in the abstract alone do not carry enough specific information to learn the mappings from text to keywords. These results suggest that automatic keyword assignment is much better suited to situations in which more generalised keyword categories must be learned, rather than keywords that capture the specific niches of the research.

5.3.3 Cross-Validation Results

The performance of the **BERT Extra** model is further assessed by 6-fold cross-validation using the Nature dataset with 300 labels. For each of the 6 models trained, the entire hold-out set is used to track validation loss during training. The hold-out set is then split randomly in half to generate validation and test sets for finding the optimal threshold,

and assessing model performance, respectively. The results of this cross-validation are presented in Table 5.20, along with the result obtained with the same model using the balanced training/validation/test splitting strategy of the main study (Section 4.3.1).

	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
Cross-Validation Models						
# 1	0.3180	0.3989	0.3539	0.2357	0.0367	0.5025
# 2	0.3228	0.4598	0.3793	0.2470	0.0342	0.5553
# 3	0.3307	0.3221	0.3263	0.2234	0.0562	0.4976
# 4	0.3040	0.4739	0.3704	0.2470	0.0318	0.5628
# 5	0.3284	0.4194	0.3684	0.2448	0.0342	0.5179
# 6	0.2767	0.4046	0.3286	0.2180	0.0196	0.5260
Average	**0.3134**	**0.4131**	**0.3564**	**0.2360**	**0.0355**	**0.5270**
Train/Validation/Test Model						
-	0.5535	0.5623	0.5579	0.4433	0.1848	0.7441

Table 5.20: Results of 6-fold cross validation of BERT Extra in keyword prediction with the Nature dataset with 300 labels

The **BERT Extra** model achieves consistent performance across the cross-validation experiments, but worse performance overall than the model trained with data split into training, validation, and test sets. These results indicate that the data splitting strategy, in which training/validation/test sets are created in such a way as to ensure proportionate representation from each journal (Section 4.3.1), produces better results than splitting the data randomly. Random splitting of the data can result in disproportionate representation from each journal in the respective datasets, which deteriorates model performance in this application. The deterioration in performance is observed in all the representative metrics, but is particularly apparent in the *Exact Match Ratio* (3.55% vs. 18.48%).

5.3.4 ADAM Optimiser Results

The comparative results obtained using both Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (ADAM) optimisation on the Nature dataset with 1234 labels are sumamrised in Table 5.21:

Optimiser	Precision	Recall	F1 Score	Jaccard	Exact Match	Accuracy
SGD	0.2726	0.3529	0.3076	0.1891	0.0128	0.4037
ADAM	0.3331	0.4405	0.3794	0.2493	0.0326	0.5440

Table 5.21: Comparison of performance of SGD vs. ADAM optimisers for BERT Extra in keyword prediction with the Nature dataset with 300 labels

These results show that ADAM significantly improves model performance in comparison to SGD. This performance boost comes at a greater computational cost per iteration *i.e.* fewer iterations per second. Convergence was reached in 68 epochs with

SGD and 105 epochs with ADAM. These factors combined result in ADAM requiring approximately double the time to train, but an appreciable boost in model performance is the reward for the extra expense.

5.4 Comparison of Word Embeddings

The results obtained in this study show the value of word embedding representations: in the complex task of keyword assignment (prediction), the **Word2Vec** and **transformer** models consistently out-perform the **bag-of-words** models, although the same is not always true for the simpler task of domain prediction. The value of context-capturing word embeddings is further demonstrated by the consistently superior performance of all tested **transformer** models relative to the **Word2Vec** models, in the keyword prediction task. The question of whether bi-directional context is indeed superior to left-to-right context in the application of **transformer** models is not as easy to answer. The **OpenAI-GPT** model (left-to-right) generally performed well in the keyword prediction task, and was the best-performing model in one instance. The models based on the **BERT** architecture do perform better than the **OpenAI-GPT** model in aggregate. However, this difference is not significant enough to draw definitive conclusions about the impact of the pre-training modelling objective on the keyword prediction task *i.e.* Causal Language Modelling (left-to-right) or Masked Language Modelling (bi-directional). There are other significant variables such as the choice of pre-training corpus for the transformers that could account for the discrepancies in performance between the various models. This point is observed in the different results obtained by **SciBERT** comared to **BERT**.

5.5 Comparison of Classifier Architectures

The **Bag-of-words** text representation is a very sparse vector that does not carry a lot of information. The classifier must therefore extract features from the input in such a way that complex subject matter can be inferred. The architecture with 5 hidden layers that steadily contract from one layer to the next, consistently out-performs the larger architecture with 10 hidden layers of fixed size. The larger number of parameters of the latter model must be trained with the same learning signal, and the results indicate that feature extraction is not as effective in this application with the larger model. It must also be noted that the learning signal is only strong enough to train the **Bag-of-words** model over the smallest set of target labels for both the Nature dataset and IEEE dataset.
The **Word2Vec** representations do carry more complex corpus-level information than **Bag-of-words** representations. However, when applied to the task of keyword prediction, it is still beneficial to use multi-layer classifiers to capture more complex interactions than those already contained in the **Word2Vec** embeddings [Ramasubramanian *et al.* 2020]. The best-performing **Word2Vec** architecture is consistently that with 4 hidden layers with steadily decreasing size, just as it is in the case of **Bag-of-words**. The architecture with 8 hidden layers of the same size does not perform nearly as well,

indicating that these models are not able to extract features as effectively, with the same learning signal as the smaller model.
The representations obtained from the **transformer** language models carry the most amount of information about the input text, of all representations studied. In the keyword prediction task, the best-performing architecture is consistently that with only 1 linear classification layer (0 hidden layers). It is evident that adding additional hidden layers does not assist in extracting more useful features from the transformer embeddings. It is speculated that this is because the transformer embedding is produced by a far more complex model that captures complex interactions in the input texts. Additional hidden layers only provide linear transformations and linear activation functions, which evidently do not add any value to the representation, for the purpose of keyword prediction. It must also be noted that the same learning signal that trains the classifier parameters, also fine-tunes the **transformer** parameters. This means that the word embeddings produced by the model change throughout training, which stands in contrast to both **Bag-of-words** and **Word2Vec** models. The non-static classifier input in the case of the **transformers** may cause instability in training the classifier parameters, which may contribute to the poorer performance when using larger classifier architectures.

5.6 Analysis of Parameters per Model

The **Word2Vec** model with 4 contracting hidden layers has the least trainable parameters of all models *i.e.* 8M parameters. The **Bag-of-words** model with 5 contracting hidden layers is slightly larger, with 9.6M parameters. The **transformer** models have significantly more parameters *i.e.* 109.8M for **BERT** and 116.8M for **OpenAI-GPT**. Although these **transformer** models consistently produce superior results in keyword prediction when compared to **Word2Vec** and **Bag-of-words** models, this performance increase is not proportional to the increase in model parameters. For example, comparing the best-performing models in the keyword prediction task using the Nature dataset with 300 labels (table 5.12), the **BERT Extra** and **Word2Vec** models achieve an F_1 Score of 0.4770 and 0.2868, respectively. This amounts to a performance increase of 66.3%, at a cost of 1272.5% more parameters. It must be noted that there is very little to separate the performance of all models in the domain prediction task. For example, comparing the best-performing models in the domain prediction task using the Nature dataset (table 5.4), the **SciBERT** and **Word2Vec** models achieve an accuracy of 89.01% and 91.07%, respectively. This observation indicates that **transformer** models might not be worth the significantly higher computational cost for simpler tasks, depending on the practical value of the marginal performance improvement.

5.7 Qualitative Analysis

In this section, one example from each journal is randomly selected from the corresponding test set for qualitative analysis. The lowest number of labels is used for both datasets (300 labels for Nature and 467 labels for IEEE), and the best-performing model is used to generate predictions (**BERT Extra** for both datasets). The assigned keywords

are compared qualitatively to both the labels predicted by applying the optimal threshold, as well as to the top *n* keywords (where *n* is equal to the number of assigned keywords for the sample).

5.7.1 Nature Climate

Abstract: *"The air–sea transfer of heat and fresh water plays a critical role in the global climate system (ref. 1). This is particularly true for the Greenland and Iceland seas, where these fluxes drive ocean convection that contributes to Denmark Strait overflow water, the densest component of the lower limb of the Atlantic Meridional Overturning Circulation (AMOC; ref. 2). Here we show that the wintertime retreat of sea ice in the region, combined with different rates of warming for the atmosphere and sea surface of the Greenland and Iceland seas, has resulted in statistically significant reductions of approximately 20% in the magnitude of the winter air–sea heat fluxes since 1979. We also show that modes of climate variability other than the North Atlantic Oscillation (NAO; refs 3, 4, 5, 6, 7) are required to fully characterize the regional air–sea interaction. Mixed-layer model simulations imply that further decreases in atmospheric forcing will exceed a threshold for the Greenland Sea whereby convection will become depth limited, reducing the ventilation of mid-depth waters in the Nordic seas. In the Iceland Sea, further reductions have the potential to decrease the supply of the densest overflow waters to the AMOC (ref. 8)."*

Keywords:

Assigned Keywords	**Predicted Keywords**	**Top *n* Keywords**
physical oceanography	physical oceanography	physical oceanography
-	climate change	climate change
cryospheric science	cryospheric science	cryospheric science
projection & prediction	projection & prediction	-

Table 5.22: Assigned keywords, predicted keywords and top *n* Keywords for a random sample from the Nature Climate journal

Comments: In this example, there are 3 assigned keywords and all are successfully predicted by the model. There is one additional keyword predicted ("climate change"), which is thematically relevant to the content of the abstract, which is indeed speaking of a particular climate change phenomenon. The top 3 predictions of the model exclude "projection prediction", because "climate change" scored more highly. These results do indicate that the model was able to successfully infer appropriate labels to assign to the text.

5.7.2 Nature Medicine

Abstract: *"The ubiquitously expressed non-receptor protein tyrosine phosphatase SHP2, encoded by PTPN11, is involved in signal transduction downstream of multiple growth factor, cytokine and integrin receptors (ref 1). Its requirement for complete RAS–MAPK activation and its role as a negative regulator of JAK–STAT signaling have established SHP2*

as an essential player in oncogenic signaling pathways (refs 1,2,3,4,5,6,7). Recently, a novel potent allosteric SHP2 inhibitor was presented as a viable therapeutic option for receptor tyrosine kinase-driven cancers, but was shown to be ineffective in KRAS-mutant tumor cell lines in vitro (ref 8). Here, we report a central and indispensable role for SHP2 in oncogenic KRAS-driven tumors. Genetic deletion of Ptpn11 profoundly inhibited tumor development in mutant KRAS-driven murine models of pancreatic ductal adenocarcinoma and non-small-cell lung cancer. We provide evidence for a critical dependence of mutant KRAS on SHP2 during carcinogenesis. Deletion or inhibition of SHP2 in established tumors delayed tumor progression but was not sufficient to achieve tumor regression. However, SHP2 was necessary for resistance mechanisms upon blockade of MEK. Synergy was observed when both SHP2 and MEK were targeted, resulting in sustained tumor growth control in murine and human patient-derived organoids and xenograft models of pancreatic ductal adenocarcinoma and non-small-cell lung cancer. Our data indicate the clinical utility of dual SHP2/MEK inhibition as a targeted therapy approach for KRAS-mutant cancers."

Keywords:

Assigned Keywords	**Predicted Keywords**	**Top *n* Keywords**
cancer models	cancer	cancer
translational research	translational research	translational research
preclinical research	drug development	drug development
experimental models of disease	cell signalling	cell signalling
cancer therapy	cancer metabolism	cancer metabolism

Table 5.23: Assigned keywords, predicted keywords and top *n* Keywords for a random sample from the Nature Medicine journal

Comments: In this example, there are 5 assigned keyword and only 1 was successfully predicted by the model. However, all keywords that were predicted are relevant to the subject matter of the text. For example, "cancer" and "cancer metabolism" were returned by the model, which are relevant to the text and close to the assigned keyword "cancer models". Furthermore, the text makes several mentions of signalling in the context of cancer cells, which is relevant to the returned keyword "cell signalling". The text makes mention of a patented "allosteric SHP2 inhibitor" drug, which is relevant to the returned keyword "drug development.

5.7.3 Nature Materials

Abstract: *"The Hall effect occurs only in systems with broken time-reversal symmetry, such as materials under an external magnetic field in the ordinary Hall effect and magnetic materials in the anomalous Hall effect (AHE) [ref 1]. Here we show a nonlinear AHE in a non-magnetic material under zero magnetic field, in which the Hall voltage depends quadratically on the longitudinal current (refs 2,3,4,5,6). We observe the effect in few-layer Td-WTe2, a two-dimensional semimetal with broken inversion symmetry and only one mirror line in the crystal plane. Our angle-resolved electrical measurements reveal*

that the Hall voltage maximizes (vanishes) when the bias current is perpendicular (parallel) to the mirror line. The observed effect can be understood as an AHE induced by the bias current, which generates an out-of-plane magnetization. The temperature dependence of the Hall conductivity further suggests that both the intrinsic Berry curvature dipole and extrinsic spin-dependent scatterings contribute to the observed nonlinear AHE."

Keywords:

Assigned Keywords	**Predicted Keywords**	**Top *n* Keywords**
electronic properties & materials	electronic properties & materials	electronic properties & materials
topological matter	topological matter	topological matter
magnetic properties & materials	magnetic properties & materials	magnetic properties & materials
-	condensed matter physics	-
-	quantum hall	-

Table 5.24: Assigned keywords, predicted keywords and top *n* Keywords for a random sample from the Nature Materials journal

Comments: In this example, there are 3 assigned keywords, which are all successfully returned by the model. The additional keywords returned are again relevant to the subject matter. The text makes mention of the hall effect, which is relevant to the returned keyword "quantum hall". The returned keyword "condensed matter physics" refers to "the field of physics that deals with the macroscopic and microscopic physical properties of matter, especially the solid and liquid phases which arise from electromagnetic forces between atoms" (https://en.wikipedia.org/wiki/Condensed_matter_physics), which also appears to be relevant to the subject matter of the text.

5.7.4 Nature Physics

Abstract: *"Hexagonal iron sulfide exhibits a fascinating coexistence of metal–insulator, structural and magnetic transitions, reflecting an intimate interplay of its spin, phonon and charge degrees of freedom. Here, we show how a subtle competition of energetic and entropic free-energy components governs its thermodynamics and the sequence of phase transitions it undergoes upon cooling. By means of comprehensive neutron and X-ray scattering measurements, and supported by first-principles electronic structure simulations, we identify the critical role of the coupling between antiferromagnetic ordering and instabilities of anharmonic phonons in the metallic phase in driving the metal–insulator transition. The antiferromagnetic ordering enables the emergence of two zone-boundary soft phonons, whose coupling to a zone-centre mode drives the lattice distortion opening the electronic bandgap. Simultaneously, spin–lattice coupling opens a gap in the magnon spectrum that controls the entropy component of the metal–insulator transition free energy. These results reveal the importance of spin–phonon coupling to tune anharmonic effects, thus opening new avenues to design novel technologically important materials harbouring the metal–insulator transition and magnetoelectric behaviours."*

Keywords:

Assigned Keywords	Predicted Keywords	Top *n* Keywords
magnetic properties & materials	magnetic properties & materials	magnetic properties & materials
-	electronic properties & materials	electronic properties & materials
phase transitions & critical phenomena	phase transitions & critical phenomena	phase transitions & critical phenomena
ferroelectrics & multiferroics	ferroelectrics & multiferroics	ferroelectrics & multiferroics
structure of solids & liquids	structure of solids & liquids	-
-	surfaces, interfaces & thin films	-

Table 5.25: Assigned keywords, predicted keywords and top *n* Keywords for a random sample from the Nature Physics journal

Comments: In this example, there are 4 assigned keywords, which are all successfully returned by the model. There are also two additional keywords returned i.e. *i.e.* "electronic properties & materials" and "surfaces, interfaces & thin films". The keyword "electronic properties & materials" is relevant to the central theme of the text, which makes mention of "electronic structure simulations", "electronic bandgap", and "magnetoelectric behaviours". These observations are central to the premise of this study *i.e.* that keywords can be assigned to texts based on thematic relevance, without the exact terms appearing in the text , which stands in contrast to keyword extraction techniques. It is not clear whether the returned keyword "surfaces, interfaces thin films" is relevant to the text.

5.7.5 IEEE Transactions on Geoscience and Remote Sensing

Abstract: *"The all-weather capability makes synthetic aperture radar (SAR) interesting with respect to glaciological studies in remote Arctic areas. The aim of this project was to investigate how the European Remote Sensing Satellite (ERS) SAR backscatter from certain glaciological and geomorphological structures varied with time and find out at which time of the year a SAR acquisition will give the best result when observing certain features. Five ERS-1 SAR images, one from the winter and four from the summer, were acquired over the northwestern part of Svalbard, Norway. Ground measurements and observations were made at the same time as one of the summer SAR acquisitions. The ground data as well as meteorological recordings were used to analyze the SAR backscatter changes in the multitemporal data set. A zonation on the glaciers representing snow, firn/superimposed ice, and glacier ice was detected in the winter image only. The equilibrium line could also be derived from this SAR image. Information on drainage patterns, distribution of wet and dry snow areas, and the occurrence of crevasses was derived from the summer images.*

Changes in snow cover distribution was identified in the summer images. The form and position of ice-cored moraines, fossil beach ridges, and river channels were identified in all images. Early summer images showed the highest potential for identification of landforms. Small-scale landforms, such as patterned ground and tundra polygons, could not be identified."

Keywords:

Assigned Keywords	**Predicted Keywords**	**Top *n* Keywords**
satellites	satellites	satellites
remote sensing	remote sensing	remote sensing
synthetic aperture radar	synthetic aperture radar	synthetic aperture radar
ice	ice	ice
backscatter	backscatter	backscatter
snow	snow	snow
arctic	arctic	arctic
meteorology	meteorology	meteorology
rivers	rivers	rivers
time measurement	-	clouds

Table 5.26: Assigned keywords, predicted keywords and top *n* Keywords for a random sample from the IEEE Transactions on Geoscience and Remote Sensing journal

Comments: In this example there are 10 assigned keywords. The model makes 9 predictions, all of which are correct. The last keyword, which the model fails to predict, is "time measurement". Considering the top *n* keywords, the next most-probable keyword is "clouds", which is not relevant to the text but is still within the appropriate domain. This example once again demonstrates the ability of the model to predict appropriate keywords for the input text.

5.7.6 IEEE Transactions on Pattern Analysis and Machine Intelligence

Abstract: *"Radar emitter classification is a special application of data clustering for classifying unknown radar emitters from received radar pulse samples. The main challenges of this task are the high dimensionality of radar pulse samples, small sample group size, and closely located radar pulse clusters. In this paper, two new online clustering algorithms are developed for radar emitter classification: One is model-based using the minimum description length (MDL) criterion and the other is based on competitive learning. Computational complexity is analyzed for each algorithm and then compared. Simulation results show the superior performance of the model-based algorithm over competitive learning in terms of better classification accuracy, flexibility, and stability."*

Keywords:

Assigned Keywords	**Predicted Keywords**	**Top *n* Keywords**
clustering algorithms	clustering algorithms	clustering algorithms
-	spaceborne radar	spaceborne radar
algorithm design & analysis	algorithm design & analysis	algorithm design & analysis
computational complexity	computational complexity	computational complexity
-	radar scattering	radar scattering
doppler radar	doppler radar	doppler radar
-	remote sensing	remote sensing
-	radar remote sensing	radar remote sensing
computational efficiency	computational efficiency	computational efficiency
classification algorithms	-	-
computational model	-	-
neural network	-	-
radar applications	-	-
radar detection	-	-

Table 5.27: Assigned keywords, predicted keywords and top *n* Keywords for a random sample from the IEEE Transactions on Pattern Analysis and Machine Intelligence journal

Comments: In this example there are 10 assigned keywords. The model makes 9 predictions, of which 5 are correct. Of the remaining 4 keywords predicted by the model, 3 of them (”spaceborne radar”, ”radar scattering”, ”radar remote sensing”) are thematically similar to the assigned keywords ”radar applications” and ”radar detection”, which were not predicted. It is interesting to note that the text is vague about the specific radar technology being studied, and the model is returning several keywords related to radar applications. The model is also unable to predict ”classification algorithms”, ”computational model”, and ”neural network”.

5.8 Conclusion

The results presented in this section clearly demonstrate the relative ordering of the various models in terms of performance in both domain- and keyword prediction. It is clear that the transformer models produce the most sensible outputs, which is particularly true in the case of the keyword prediction task. Performance is observed to improve when infrequently-occurring labels are pruned from the target variables. Most notably, it is observed that the keywords returned for test texts tend to be thematically relevant to the text, even if they are not in the list of assigned keywords for the text. This is an important feature of model performance because the performance metrics themselves do not create a sense of confidence that even the best models perform very well, with maximal F_1 sores in the region of 0.5. If cases of incorrect predictions were drastically dissimilar to the theme of the text e.g. ”climate change” being returned for a text about cancer, then these models would not be of much use in the application

of automatic keyword assignment. The qualitative analysis of keywords presented in Section 5.7 shows that this is not the case, and that incorrect keywords returned seem to be either variations of keywords that are expected, or at least within the appropriate domain. It is also observed that some of the expected keywords are missed, which would present as lower recall scores, but this is far less concerning than having drastic thematic mismatches.

Chapter 6

Conclusion

In his study, the complex keyword assignment task is studied rigorously, using datasets that reasonably represent the academic settings in which this task is applicable. A baseline method is implemented (**bag-of-words**) to form the foundation against which to compare results. The method proposed by Ramasubramanian *et al.* [2020], which relies on the **Word2Vec** architecture is implemented for reference. Lastly, several **transformer** models are implemented with the aim of advancing the performance frontier established by the preceding methods. All the models are also tested in the simpler task of journal (domain) prediction, to establish a benchmark for comparing model performance in tasks of varying complexity.

The results show that the best performance is indeed achieved using architectures that make use of the word embeddings produced by **transformer** models, which is particularly apparent in the complex task of keyword assignment. Furthermore, it is found that the best-performing models are able to produce outputs that are thematically coherent with respect to the input texts.

Future work may improve on the results presented in this paper by employing more sophisticated text pre-processing techniques, such as explicitly catering for phrase retention. Further improvements may be obtained by improving the optimisation strategy.

The coherence of the results indicate that the methods proposed may be applicable in a commercial setting as a tool to recommend keywords to authors and publishers, which could reduce the manual effort and specialist knowledge required to categorise publications effectively. For this commercial use case, there are also potential synergies with keyword extraction algorithms that can be utilised to enhance the relevance of returned keywords, as well as to expand the list of potential keywords to include terms that appear in the text but are not included in the controlled vocabulary established in the keyword assignment task.

References

[Augenstein *et al.* 2017] Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[Behrouzi *et al.* 2020] Saman Behrouzi, Zahra Shafaeipour Sarmoor, Khosrow Hajsadeghi, and Kaveh Kavousi. Predicting scientific research trends based on link prediction in keyword networks. *Journal of Informetrics*, 14(4):101079, 2020.

[Beliga 2014] Slobodan Beliga. 1 keyword extraction : a review of methods and approaches. 2014.

[Beltagy *et al.* 2019] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *EMNLP/IJCNLP*, 2019.

[Blei *et al.* 2003] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 05 2003.

[Bojanowski *et al.* 2016] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 07 2016.

[Devlin *et al.* 2019] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[Hasan *et al.* 2018] H. M. M. Hasan, F. Sanyal, and D. Chaki. A novel approach to extract important keywords from documents applying latent semantic analysis. In *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pages 117–122, 2018.

[Hu *et al.* 2018] Jie Hu, Shaobo Li, Yong Yao, Liya Yu, Yang Guanci, and Jianjun Hu. Patent keyword extraction algorithm based on distributed representation for patent classification. *Entropy*, 20:104, 02 2018.

[Hulth 2003] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. 06 2003.

[K Sarma *et al.* 2018] Prathusha K Sarma, Yingyu Liang, and Bill Sethares. Domain adapted word embeddings for improved sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 37–42, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Kamper *et al.* 2017] Herman Kamper, Shane Settle, Gregory Shakhnarovich, and Karen Livescu. Visually grounded learning of keyword prediction from untranscribed speech. pages 3677–3681, 08 2017.

[Kim *et al.* 2010] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[Langville and Meyer 2004] Amy Langville and Carl Meyer. Deeper inside pagerank. *Internet Mathematics*, 1, 01 2004.

[Li *et al.* 2019] Teng-Fei Li, Liang Hu, Jian-Feng Chu, Hong-Tu Li, and Ling Chi. An unsupervised approach for keyphrase extraction using within-collection resources. *IEEE Access*, PP:1–1, 08 2019.

[Liu *et al.* 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 07 2019.

[Mahata *et al.* 2018] Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. pages 634–639, 01 2018.

[Melamud *et al.* 2016] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Mikolov *et al.* 2013] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. pages 1–12, 01 2013.

[Mothe *et al.* 2018] Josiane Mothe, Faneva Ramiandrisoa, and Michael Rasolomanana. Automatic keyphrase extraction using graph-based methods. pages 728–730, 04 2018.

[Onan *et al.* 2016] Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57, 03 2016.

[Papagiannopoulou and Tsoumakas 2018] Eirini Papagiannopoulou and Grigorios Tsoumakas. Local word vectors guiding keyphrase extraction. *Information Processing and Management*, 54, 01 2018.

[Pennington *et al.* 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Peters *et al.* 2017] Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. 04 2017.

[Peters *et al.* 2018] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 02 2018.

[Radford 2018] A. Radford. Improving language understanding by generative pre-training. 2018.

[Raghavan and Krishnamurthy 2013] Kotikanyadanam Raghavan and Madaiah Krishnamurthy. An overview of classification technologies. *Desidoc*, 33:306–313, 07 2013.

[Ramasubramanian *et al.* 2020] M. Ramasubramanian, H. Muhammad, I. Gurung, M. Maskey, and R. Ramachandran. Es2vec: Earth science metadata keyword assignment using domain-specific word embeddings. In *2020 SoutheastCon*, pages 1–6, 2020.

[Schluter 2015] Natalie Schluter. A critical survey on measuring success in rank-based keyword assignment to documents. In *Proc of TALN*, United States, 2015. Association for Computational Linguistics.

[Singhal *et al.* 2017] Ayush Singhal, Ravindra Kasturi, Ankit Sharma, and Jaideep Srivastava. Leveraging web resources for keyword assignment to short text documents. 06 2017.

[Smith 2017] Leslie N. Smith. *Cyclical Learning Rates for Training Neural Networks*, 2017.

[Srivastava *et al.* 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[Sylva 2013] Lyne Da Sylva. Nlp and digital library management. 2013.

[Turc *et al.* 2019] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: The impact of student initialization on knowledge distillation. 08 2019.

[Vanyushkin and Graschenko 2020] Alexander Vanyushkin and Leonid Graschenko. Analysis of text collections for the purposes of keyword extraction task. *Journal of Information and Organizational Sciences*, 44(1), Jun. 2020.

[Vaswani *et al.* 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

[Wu *et al.* 2016] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc Le, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. 09 2016.

[Yin and Schütze 2014] Wenpeng Yin and Hinrich Schütze. An exploration of embeddings for generalized phrases. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 41–47, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[Ying *et al.* 2017] Yan Ying, Tan Qingping, Xie Qinzheng, Zeng Ping, and Li Panpan. A graph-based approach of automatic keyphrase extraction. *Procedia Computer Science*, 107:248–255, 12 2017.

[Yu and Dredze 2015] Mo Yu and Mark Dredze. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242, 12 2015.

[Yu *et al.* 2019] S. Yu, J. Su, and D. Luo. Improving bert-based text classification with auxiliary sentence and domain knowledge. *IEEE Access*, 7:176600–176612, 2019.

[Zhang and Xu 2009] C. Zhang and H. Xu. Using citation-knn for automatic keyword assignment. In *2009 International Conference on Electronic Commerce and Business Intelligence*, pages 131–134, 2009.