Hierarchical Multi-Label Classification of Scientific Documents

Mobashir Sadat **Cornelia Caragea**
Computer Science
University of Illinois Chicago
msadat3@uic.edu cornelia@uic.edu

Abstract

Automatic topic classification has been studied extensively to assist managing and indexing scientific documents in a digital collection. With the large number of topics being available in recent years, it has become necessary to arrange them in a hierarchy. Therefore, the automatic classification systems need to be able to classify the documents hierarchically. In addition, each paper is often assigned to more than one relevant topic. For example, a paper can be assigned to several topics in a hierarchy tree. In this paper, we introduce a new dataset for hierarchical multi-label text classification (HMLTC) of scientific papers called **SciHTC**, which contains $186,160$ papers and $1,233$ categories from the ACM CCS tree. We establish strong baselines for HMLTC and propose a multi-task learning approach for topic classification with keyword labeling as an auxiliary task. Our best model achieves a Macro-F1 score of 34.57% which shows that this dataset provides significant research opportunities on hierarchical scientific topic classification. We make our dataset and code available on Github.[1]

1 Introduction

With the exponential increase of scientific documents being published every year, the difficulty in managing and categorizing them in a digital collection is also increasing. While the enormity of the number of papers is the most important reason behind it, the problem can also be assigned to the large number of topics. It is a very difficult task to index a paper in a digital collection when there are thousands of topics to choose from. Fortunately, the large number of topics can be arranged in a hierarchy because, except for a few general topics, all topics can be seen as a sub-area of another topic. After arranging the topics in a hierarchy tree, the task of categorizing a paper becomes much simpler since now there are only a handful of topics to choose from at each level of the hierarchy. However, manual assignment of topics to a large number of papers is still very difficult and expensive, making an automatic system of hierarchical classification of scientific documents a necessity.

After arranging the topics in a hierarchy, the classification task no longer remains a multi-class classification because in multi-class classification, each paper is classified into exactly one of several mutually exclusive classes or topics. However, if the topics are arranged in a hierarchy, they no longer remain mutually exclusive: a paper assigned to a topic node a in the hierarchy also gets assigned to topic node b, where b is a node in the parental history of a. For example, if a paper is classified to the area of natural language processing (NLP), it is also assigned to the topic of artificial intelligence (AI) given that AI is in the parental history of NLP. This *non-mutual exclusivity* among the topics makes the task a multi-label classification task.

Despite being an important problem, hierarchical multi-label topic classification (HMLTC) has not been explored to a great extent in the context of scientific papers. Most works on hierarchical and/or multi-label topic classification focus on news articles (Banerjee et al., 2019; Peng et al., 2018) and use the RCV-1 dataset (Lewis et al., 2004) for evaluation. This is partly because of a lack of datasets for hierarchically classified scientific papers, which hinders progress in this domain. Precisely, the existing multi-label datasets of scientific papers are either comparatively small (Kowsari et al., 2017) or the label hierarchy is not deep (Yang et al., 2018).

Therefore, we address the scarcity of datasets for HMLTC on scientific papers by introducing a new large dataset called **SciHTC** in which the papers are hierarchically classified based on the ACM CCS tree.[2] Our dataset is large enough to allow deep learning exploration, comprising $186,160$ re-

[1] https://github.com/msadat3/SciHTC

[2] https://dl.acm.org/ccs

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 8923 - 8937

search papers that are organized into 1,233 topics, which are arranged in a six-level deep hierarchy. We establish several strong baselines for both hierarchical and flat multi-label classification for **SciHTC**. In addition, we conduct a thorough investigation on the usefulness of author specified keywords in topic classification. Furthermore, we show how multi-task learning with scientific document classification as the principal task and its keyword labeling as the auxiliary task can help improve the classification performance. However, our best models with SciBERT (Beltagy et al., 2019) achieve only 34.57% Macro-F1 score which shows that there is still plenty of room for improvement.

2 Related Work

To date, several datasets exist for topic classification of scientific papers. Kowsari et al. (2017) created a hierarchically classified dataset of scientific papers from the Web of Science (WoS).[3] However, their hierarchy is only two levels deep and the size of their dataset is 46,985, which is much smaller than its counterpart from news source data. In addition, there are only 141 topics in the entire hierarchy. The Cora[4] dataset introduced by McCallum et al. (2000) is also hierarchically classified with multiple labels per paper and contains about 50,000 papers. The hierarchy varies in depth from one to three and has 79 topics in total. However, the widely used version of Cora[5] contains only 2,708 papers (Lu and Getoor, 2003) and is not hierarchical. Similarly, the labeled dataset for topic classification of scientific papers from CiteSeer[6] (Giles et al., 1998) is also very small in size containing only 3,312 papers with no hierarchy over the labels. Yang et al. (2018) created a dataset of 55,840 arXiv papers where each paper is assigned multiple labels using a two-level deep topic hierarchy containing a total of 54 topics. Similar to us, Santos and Rodrigues (2009) proposed a multi-label hierarchical document classification dataset using the ACM category hierarchy. However, our dataset is much larger in size than this dataset (which has $\approx$ 15,000 documents in their experiment setting). Furthermore, the dataset by Santos and Rodrigues (2009) is not available online and cannot be reconstructed as the ACM paper IDs are not provided.

Recently, Cohan et al. (2020) released a dataset of 25,000 papers collected from the Microsoft Academic Graph[7] (MAG) as part of their proposed evaluation benchmark for document level research on scientific domains. Although the papers in MAG are arranged in a five level deep hierarchy (Sinha et al., 2015), only the level one categories (19 topics in total) are made available with the dataset. In contrast to the above datasets, **SciHTC** has 1,233 topics arranged in a six level deep hierarchy. The total number of papers in our dataset is 186,160 which is significantly larger than all other datasets mentioned above. The topic hierarchy of each paper is provided by their respective authors. Since each paper in our dataset is assigned to all the topics on the path from the root to a certain topic in the hierarchy tree, our dataset can be referred to as a multi-label dataset for topic classification.

For multi-label classification, there are two major approaches: a) training one model to predict all the topics to which each paper belongs (Peng et al., 2018; Baker and Korhonen, 2017b; Liu et al., 2017); and b) training one-vs-all binary classifiers for each of the topics (Banerjee et al., 2019; Read et al., 2009). The first approach learns to classify papers to all the relevant topics simultaneously, and hence, it is better suited to leverage the inter-label dependencies among the labels. However, despite that it is simpler and more time efficient, it struggles with data imbalance (Banerjee et al., 2019). On the other hand, the second approach gives enough flexibility to deal with the different levels of class imbalance but it is more complex and not as time efficient as the first one. In general, the second approach takes additional steps to encode the inter-label dependencies among the co-occurring labels. For example, in hierarchical classification, the parameters of the model for a child topic can be initialized with the parameters of the trained model for its parent topic (Kurata et al., 2016; Baker and Korhonen, 2017a; Banerjee et al., 2019). In this work, we take both approaches and compare their performance.

Besides these approaches, another approach for hierarchical and/or multi-label classification in recent years is based on sequence-to-sequence models (Yang et al., 2018, 2019), which we explored in this work. However, these models failed to show

[3] https://clarivate.com/webofsciencegroup/solutions/web-of-science/

[4] https://people.cs.umass.edu/ mccallum/data/cora-classify.tar.gz

[5] https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz

[6] https://linqs-data.soe.ucsc.edu/public/citeseer-mrdm05/

[7] https://academic.microsoft.com/home

satisfactory performance on our dataset. We also explored the hierarchical classification proposed by Kowsari et al. (2017) where a local classifier is trained at every node of the hierarchy, but this model also failed to gain satisfactory performance.

Onan et al. (2016) proposed the use of keywords together with traditional ensemble methods to classify scientific papers. However, since ground truth keywords were not available for the papers in their dataset, the authors explored a frequency based keyword selection, which gave the best performance. Therefore, their application of keyword extraction methods for the classification task can be seen as a feature selection method.

Our **SciHTC** dataset, in addition to being very large, multi-labeled, and hierarchical, contains the author-provided keywords for each paper. In this work, we present a thorough investigation of the usefulness of keywords for topic classification and propose a multi-task learning framework (Caruana, 1993; Liu et al., 2019) that uses keyword labeling as an auxiliary task to learn better representations for the main topic classification task.

3 The SciHTC Dataset

We constructed the **SciHTC** dataset from papers published by the ACM digital library, which we requested from ACM.

Precisely, the dataset provided by ACM has more than $300,000$ papers. However, some of these papers did not have their author-specified keywords, whereas others did not have any category information. Thus, we pruned all these papers from the dataset. Finally, there were $186,160$ papers which had all the necessary attributes and the category information. The final dataset was randomly divided into train, development and test sets in an $80:10:10$ ratio. The number of examples in each set can be seen in Table 1.

Dataset	Size
Train	148,928
Development	18,616
Test	18,616

Table 1: Dataset Splits.

The category information of the papers in our dataset were defined based on the category hierarchy tree created by ACM. This hierarchy tree is named CCS or Computing Classification System. The root of the hierarchy tree is denoted as 'CCS' and there are 13 nodes at level 1 which represent topics such as "Hardware," "Networks," "Security and Privacy," etc. Note that CCS itself does not represent any topic (or category). It is simply the root of the ACM hierarchy tree. There are 6 levels in the hierarchy tree apart from the root 'CCS'. That is, the maximum depth among the leaf nodes in the tree is 6. However, note that the depths of different sub-branches are not uniform and there are leaf nodes in the tree with a depth less than 6.

Category Hierarchy	Score
CCS → Software and its engineering → Software creation and management → Software verification and validation → Software defect analysis → Software testing and debugging	**500**
CCS → Software and its engineering → Software notations and tools → General programming languages → Language features	100
Author-specified Keywords	
Dependent enumeration, data generation, invariant, pairing function, algebra, exhaustive testing, random testing, lazy evaluation, program inversion, DSL, SciFe	

Table 2: Category hierarchies with different relevance scores and keywords for a paper — both specified by the authors.

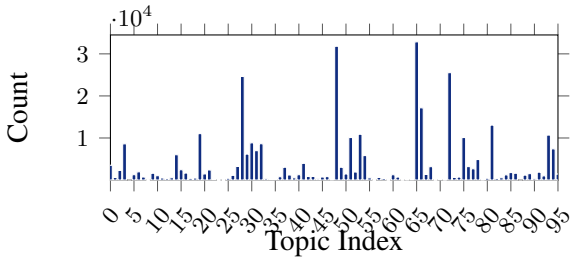

Figure 1: Number of papers in each topic up to level two of the ACM CCS hierarchy tree.

Each paper in our dataset is assigned to one or more sub-branches of the hierarchy tree by their respective authors with different depth levels and relevance scores among $\{100, 300, 500\}$ with 500 indicating the most relevant. The authors also provide a set of keywords relevant to their paper. Table 2 shows the assigned sub-branches and keywords of an example paper,[8] both of them being provided by the authors. Among the author-specified sub-branches, we only consider the sub-branch with the highest relevance score for each paper. Thus, the categories in the first sub-branch (bolded line) in Table 2 are selected as the labels for the paper. However, considering all relevant sub-branches can present a more interesting and challenging task

[8] http://dl.acm.org/citation.cfm?id=2814323

which can be explored in future work.

There are $1,233$ different topics in total in our final dataset. However, we find that the distribution of the number of papers over the topics is very imbalanced and a few topics (especially in the deeper levels of the hierarchy) had extremely low support (i.e., rare topics). Thus, for our experiments, we only consider the topics up to level 2 of the CCS hierarchy tree which had at least 100 examples in the training set. Figure 1 shows the number of papers in each of the 95 topics up to level 2 of the hierarchy tree in our dataset. We also report the explicit topic distribution (i.e., topic name vs. support) in Appendix A. Note that since there are 12 topics (among the 95 topics up to level 2 of the hierarchy) with less than 100 examples in the training set, we remove them and experiment with the remaining 83 topics. Although we do not use the topics with low support in our experiments, we believe that they can be potentially useful for hierarchical topic classification of rare topics. Therefore, we make available not only the two-level hierarchy dataset used in our experiments but also all relevant topics for each paper from the six-level hierarchy tree.

4 Methodology

This section describes the hierarchical and flat multi-label baselines used in our experiments (§4.1); after that, it introduces our simple incorporation of keywords into the models (§4.2); lastly, it presents our multi-task learning framework for topic classification (§4.3).

Problem Definition Let p be a paper, t be a topic from the set of all topics T, and n be the number of all topics in T; and let $\mathbf{x}^p$ denote the input text and $\mathbf{y}^p$ denote the label vector of size n corresponding to p. For our baseline models, $\mathbf{x}^p$ is a concatenation of the title and abstract of p. The goal is to predict the label vector $\mathbf{y}^p$ given $\mathbf{x}^p$ such that, if p belongs to topic t, $y_t^p = 1$, and $y_t^p = 0$ otherwise, i.e., identify all topics relevant to p.

4.1 Baseline Modeling

We establish both flat and hierarchical classification approaches as our baselines, as discussed below.

4.1.1 Flat Multi-Label Classification

We refer to the classifiers that predict all relevant topics of a paper with a single model as flat multi-label classifiers. Although these models leverage the inter-label dependencies by learning to predict all relevant labels simultaneously, they do not consider the label hierarchy structure. In the models, all layers are shared until the last layer during training. Instead of softmax, the output layer consists of n nodes, each with sigmoid activation. Each sigmoid output represents the probability of a topic t being relevant for a paper p, with $t = 1, \cdots, n$. The architecture is illustrated in Appendix B.

We use the following neural models to obtain representations of the input text: neural model Bi-LSTM (Hochreiter and Schmidhuber, 1997), and pre-trained language models—BERT (Devlin et al., 2019) and SciBERT (Beltagy et al., 2019).

Traditional Neural Models We use a BiLSTM based model similar to Banerjee et al. (2019) as our traditional neural baseline. Specifically, we take three approaches to obtain a single representation of the input text from the hidden states of the Bi-LSTM and concatenate them before they are sent to the fully connected layers. These approaches are: element-wise max pool, element-wise mean pool, and an attention weighted context vector. The attention mechanism is similar to the word level attention mechanism from Yang et al. (2016). After the Bi-LSTM, we use one fully connected layer with ReLU activation followed by the output layer with sigmoid activation. The obtained representations are projected with n weight matrices $\mathbf{W}_t \in \mathbb{R}^{d \times 1}$. We also explore a CNN based model as another neural baseline and report its performance and architectural design in Appendix C.

Pre-trained Language Models We fine-tune base BERT (Devlin et al., 2019) and SciBERT (Beltagy et al., 2019) using the HuggingFace[9] transformers library. We use the "bert-base-uncased" and "scibert-scivocab-uncased" variants of BERT and SciBERT, respectively. Both of these language models are pre-trained on huge amounts of text. While BERT is pre-trained on the BookCorpus (Zhu et al., 2015) and Wikipedia,[10] SciBERT is pre-trained exclusively on scientific documents. After getting the hidden state embedded in the `[CLS]` token from these models, we send them through a fully connected output layer to get the classification probability. That is, we project the `[CLS]` token with n weight matrices $\mathbf{W}_t \in \mathbb{R}^{d \times 1}$. The language model and classification parameters are jointly fine-tuned.

[9] https://github.com/huggingface/transformers

[10] https://www.wikipedia.org/

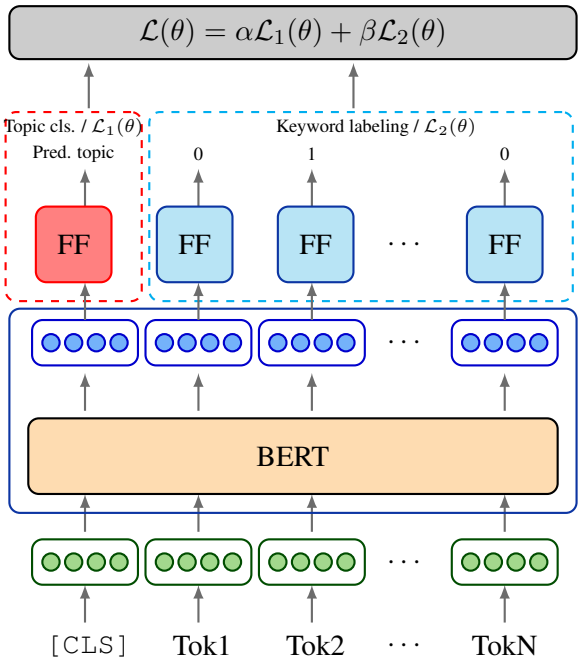

Figure 2: The architecture of our proposed multi-task learning model using BERT as the encoder. The model jointly learns two tasks: topic classification and keyword labeling. The shared layers are at the bottom whereas the task-specific layers are at the top.

4.1.2 Hierarchical Multi-Label Classification

In this approach, we train n one-vs-all binary classifiers. As with flat multi-label classification, we use both traditional neural models based architectures and pre-trained language models, which are similar to the flat architectures described in §4.1.1 with two key differences. First, the output layer no longer contains n number of nodes. Since we train binary classifiers, we change the architectures by having output layers with only one node with sigmoid activation. Second, to leverage the inter-label dependencies we initialize the model parameters of a child node in the topic hierarchy tree by its parent node's trained model parameters similar to Kurata et al. (2016); Baker and Korhonen (2017a); Banerjee et al. (2019). An illustration of this method of leveraging the topic hierarchy to learn inter-label dependencies can be seen in Appendix B.

4.2 Incorporating Keywords

We aim to improve upon the baseline models described above by incorporating the keywords specified by the authors of every paper into the model. The keywords of a paper can provide fine-grained topical information specific to a paper and at the same time are indicative about the general (coarse-grained) topics of the paper. Thus, the keywords can be seen as a bridge between the general topics of a paper and the fine details available in it (see Table 2 for examples of general topics and keywords of a paper for the fine nuances of each).

We incorporate the keywords by a simple concatenation approach. The input text $\mathbf{x}^p$ is extended with the keywords $\mathbf{k}^p$ specified by the authors of p.

$$\mathbf{x}^p := [\mathbf{x}^p, \mathbf{k}^p] \quad (1)$$

We use the same network architectures as in §4.1, in both flat and hierarchical settings.

Although this approach strikes by its simplicity and, as we will see in experiments, improves over the baselines in §4.1 that use only the title and abstract as input, it is often the case that at test time the keywords of a paper are not always available, which affects the results. Our aim is to build models that are robust enough even in the absence of keywords for papers at test time. Our proposal is to explicitly teach the model to learn to recognize the keywords in the paper that are indicative of its topics, using multi-task learning.

4.3 Multi-Task Learning with Keywords

We propose a neural multi-task learning framework for topic classification of scientific papers where the main task of topic classification is informed by the keyword labeling auxiliary task which aims to identify the keywords in the input text.

Keyword Labeling Given an input sequence $\mathbf{x}^p = \{\mathbf{x}_1^p, \cdots, \mathbf{x}_N^p\}$ (e.g., title and abstract), the objective is to predict a sequence of labels $\mathbf{z} = \{z_1, \cdots, z_N\}$, where each label z_i is 1 (a keyword) or 0 (not a keyword), i.e., predict whether a word in the sequence is a keyword or not. During training, we do an exact match of the tokenized author-specified keywords in the tokenized input text (title+abstract) and set the keyword label z_i as 1 for the positions where we find a match in the input text and 0 otherwise.

Multi-Task Learning Model The architecture of our multi-task learning model is shown in Figure 2. It jointly learns two tasks: topic classification and keyword labeling. As can be seen from the figure, the model has shared layers across both tasks at the bottom and task-specific layers at the top. In the figure, we show BERT as the encoder to avoid clutter, but in experiments we use all encoders described in §4.1.

Shared layers. The input of the model is the sequence $\mathbf{x}^p$ of N words. These words are first mapped into word embedding vectors (e.g., by summing word and positional embeddings in BERT), which are then fed into the encoder block that produces a sequence of contextual embeddings (one for each input token, including the `[CLS]` token in the transformer-based models).

Task-specific layers. There are two task-specific output layers. The topic classification output layer works the same as discussed in §4.1. On the other hand, the output layer for keywords labeling consists of a fully connected layer with sigmoid activation which predicts whether a word is a keyword or not by using each token contextual embeddding.

Training The model is optimized based on both tasks. During training, two losses are calculated for the two tasks (main and auxiliary) and they are combined together as a sum. This summed loss is used to optimize the model parameters. The overall loss $\mathcal{L}(\theta)$ in our model is as follows:

$$\mathcal{L}(\theta) = \alpha\mathcal{L}_1(\theta) + \beta\mathcal{L}_2(\theta) \quad (2)$$

where $\mathcal{L}_1(\theta)$ is the loss for topic classification and $\mathcal{L}_2(\theta)$ is the loss for keyword labeling. α and β are hyperparameters to scale the losses with different weights.

5 Experiments and Results

We perform the following experiments. First, we study the difficulty of classifying topics for scientific papers from our dataset in comparison with related datasets (§5.1). Second, we show the impact of the hierarchy of topics on models' performance and how incorporating keywords can help improve the performance further (§5.2). Third, we evaluate the performance of our proposed multi-task learning approach (§5.3). Implementation details are reported in Appendix D.

5.1 SciHTC vs. Related Datasets

We contrast **SciHTC** with three related datasets: Cora Research Paper Classification (McCallum et al., 2000), WoS-46985 (Kowsari et al., 2017) and the Microsoft Academic Graph (MAG) dataset released by Cohan et al. (2020).

The WoS dataset does not have the titles of the papers. Therefore, only the abstracts are used as the input sequence. For the Cora and MAG datasets as well as our **SciHTC** dataset, we use both title and abstract as the input sequence. We use the train, test and validation splits released by Cohan et al. (2020) for the MAG dataset but we split the other two datasets (Cora and WoS) in a 80:10:10 ratio similar to ours because they did not have author defined splits. For this experiment, our goal was to compare the degree of difficulty of our dataset with respect to the other related datasets. We thus choose to experiment with both flat and hierarchical baseline Bi-LSTM models with 300D Glove embeddings. On the MAG dataset, we only report the Flat Bi-LSTM performance since only level 1 categories are made available by the authors (with no hierarchy information). We experiment with the categories up to level 2 of the hierarchy tree for the other datasets. Table 3 shows the Macro F1 scores of these models on the four datasets along with the number of topics in each dataset and the size of each dataset. We find that:

Dataset	n	Size	F-BiLSTM	HR-BiLSTM
MAG	19	25,000	70.52%	–
Cora	79	50,000	45.71%	51.79%
WoS	141	46,985	51.99%	60.12%
SciHTC	83	186,160	24.78%	28.54%

Table 3: Number of topics n, dataset size, and Macro F1 of flat (F) and hierarchical (HR) Bi-LSTM.

SciHTC is consistently more challenging compared with related datasets. As we can see from Table 3, both models (flat and hierarchical) show a much lower performance on **SciHTC** compared with the other datasets. It is thus evident that the degree of difficulty is much higher on our dataset, making it a more challenging benchmark for evaluation.

An inspection into the categories of the related datasets revealed that these categories are more easily distinguishable from each other. For example, the categories in WoS and MAG cover broad fields of science with small overlaps between them. They range from Psychology, Medical Science, Biochemistry to Mechanical Engineering, Civil Engineering, and Computer Science. The vocabularies used in these categories/fields of science are quite different from each other and thus, the models learn to differentiate between them more easily. On the other hand, in our dataset, all papers are from the ACM digital library which are related to Computer Science and are classified to more fine-grained topics than the ones from the above datasets. Examples of topics from our dataset include Network Architectures, Network Protocols,

Approach		Precision	Recall	Micro-F1	Macro-F1
Flat-BiLSTM	w/o KW	24.02 ± 0.37	28.84 ± 0.90	46.00 ± 0.37	25.36 ± 0.50
Flat-BERT	w/o KW	28.15 ± 0.32	34.65 ± 0.47	50.03 ± 0.14	30.01 ± 0.11
Flat-SciBERT	w/o KW	29.38 ± 0.20	35.92 ± 0.51	51.23 ± 0.30	31.30 ± 0.27
HR-BiLSTM	w/o KW	26.69 ± 0.41	33.38 ± 0.60	47.36* ± 0.10	28.73* ± 0.25
HR-BERT	w/o KW	31.06 ± 0.36	36.44 ± 1.94	51.23* ± 0.24	32.20* ± 0.77
HR-SciBERT	w/o KW	31.19 ± 0.63	38.27 ± 0.19	52.16* ± 0.11	33.13* ± 0.26
Flat-BiLSTM	with KW	26.28 ± 0.87	32.02 ± 0.56	48.01$^{\#}$ ± 0.43	27.53$^{\#}$ ± 0.32
Flat-BERT	with KW	28.85 ± 0.67	35.41 ± 0.12	50.97$^{\#}$ ± 0.26	30.91 ± 0.38
Flat-SciBERT	with KW	30.85 ± 0.21	36.27 ± 0.21	52.01 ± 0.72	32.47$^{\#}$ ± 0.06
HR-BiLSTM	with KW	28.39 ± 0.07	35.18 ± 1.34	49.07$^{*\#}$ ± 0.02	30.59$^{*\#}$ ± 0.44
HR-BERT	with KW	31.99 ± 0.09	37.82 ± 0.07	52.26$^{*\#}$ ± 0.15	33.64* ± 0.26
HR-SciBERT	with KW	32.88 ± 0.47	39.37 ± 0.50	**53.17**$^{*\#}$ ± **0.18**	**34.57**$^{*\#}$ ± **0.05**

Table 4: Performance comparison between models which use keywords vs. models which do not use keywords. HR - hierarchical, KW - keywords. Best results (Micro-F1 and Macro-F1) are shown in bold. Here, * and $^{\#}$ indicate statistically significant improvements by the HR models over their flat counterparts (*) and with KW models over their w/o KW counterparts ($^{\#}$), respectively, according to a paired T-test with significance level $\alpha = 0.05$.

Approach		Precision	Recall	Micro-F1	Macro-F1
Flat-BiLSTM	with KW^{tr}	24.10 ± 1.01	27.99 ± 0.79	45.48 ± 0.54	25.04 ± 0.67
Flat-BERT	with KW^{tr}	28.01 ± 1.17	34.47 ± 1.50	49.26 ± 0.40	29.44 ± 0.38
Flat-SciBERT	with KW^{tr}	29.73 ± 0.36	34.12 ± 0.31	50.42 ± 0.60	30.96 ± 0.09
HR-BiLSTM	with KW^{tr}	27.91 ± 0.08	30.20 ± 1.03	47.16 ± 0.04	28.18 ± 0.32
HR-BERT	with KW^{tr}	31.76 ± 0.27	34.65 ± 0.04	51.04 ± 0.01	32.06 ± 0.14
HR-SciBERT	with KW^{tr}	32.13 ± 0.37	35.91 ± 0.73	51.93 ± 0.23	32.67 ± 0.25
Flat-BiLSTM	Multi-Task	23.90 ± 0.39	29.88 ± 0.57	46.18 ± 0.91	25.68 ± 0.24
Flat-BERT	Multi-Task	28.16 ± 0.94	34.49 ± 0.56	51.03* ± 0.12	30.06 ± 0.06
Flat-SciBERT	Multi-Task	31.21 ± 1.18	36.63 ± 0.79	52.24 ± 0.07	32.31* ± 0.28
HR-BiLSTM	Multi-Task	27.04 ± 0.32	33.66 ± 0.35	47.57* ± 0.23	29.11* ± 0.26
HR-BERT	Multi-Task	31.06 ± 0.07	37.20 ± 1.55	51.42* ± 0.01	32.32 ± 0.59
HR-SciBERT	Multi-Task	32.52 ± 0.89	38.01 ± 0.33	**52.48** ± **0.02**	**33.78*** ± **0.18**

Table 5: Performance comparison between models which use keywords during training by concatenating them using Eq. 1 but not during testing vs. models trained using multi-task learning which also do not use keywords at test time. The superscript ^{tr} on KW^{tr} indicates that the keywords were concatenated only during training. Best results (Micro-F1 and Macro-F1) are shown in bold. Here, * indicates statistically significant improvements of the multi-task models over the KW^{tr} models according to a paired T-test with significance level $\alpha = 0.05$.

Software Organization and Properties, Software Creation and Management, Cryptography, Systems Security, etc. Therefore, it is more difficult for the models to learn and recognize the fine differences in order to classify the topics correctly resulting in lower performance compared to the other datasets.

5.2 Impact of Hierarchy and Keywords

Next, we explore the usefulness of the hierarchy of topics and keywords for topic classification on **SciHTC**. We experiment with all of our baseline models (flat and hierarchical) described in §4.1 and with the incorporation of keywords described in §4.2. Precisely, each model is evaluated twice: first using only the input sequence (title+abstract) without the keywords and second by concatenating the input sequence with the keywords as in Eq. 1. We run each experiment three times and report their average and standard deviation in Table 4. As we can see, the standard deviations of the performance scores shown by the models are very low. This illustrates that the models are stable and easily reproducible. We make the following observations:

The hierarchy of topics improves topic classification. We can observe from Table 4 that all hierarchical models show a substantially higher performance than their flat counterparts regardless of using keywords or not. Given that the flat models learn to predict all relevant labels for each document simultaneously, it is possible for them to learn inter-label dependencies to some extent. However, due to the unavailability of the label hierarchy, the nature of the inter-label dependencies is not specified for the flat models. As a result, they can learn some spurious patterns among the labels which are harmful for overall performance. In contrast, for the hierarchical models we can specify how the inter-label dependencies should be learned (by initializing a child's model with its parent's model) which helps improve the performance as we can see in our results.

Incorporating keywords brings further improvements. From Table 4, we can also see that the performance of all of our baseline models increases when keywords are incorporated in the input sequence. These results illustrate that indeed, the fine-grained topical information provided by the keywords of each paper is beneficial for predicting its categorical labels (and thus capture an add-up effect for identifying the relevant coarser topics). Moreover, keywords can provide additional information which is unavailable in the title and the abstract but is relevant of the rest of the content and indicative of the topic of the paper. This additional information also helps the models to make better predictions.

Transformer-based models consistently outperform Bi-LSTM models and SciBERT performs best. BERT and SciBERT show strong performance across all settings (hierarchical vs. flat and with keywords vs. without) in comparison with the BiLSTM models. Interestingly, even the *flat* transformer based models outperform all BiLSTM based models (including hierarchical). We believe that this is because BERT and SciBERT are pre-trained on a large amount of text. Therefore, they are able to learn better representations of the words in the input text. Comparing the two transformer based models (BERT and SciBERT), SciBERT shows the better performance. We hypothesize that this is because SciBERT's vocabulary is more relevant to the scientific domain and it is pre-trained exclusively on scientific documents. Hence, it has a better knowledge about the language used in scientific documents.

5.3 Multi-task Learning Performance

The results in Table 4 show that the keywords are useful for topic classification but it is assumed that these keywords are available for papers not only during training but also at test time. However, often at test time the keywords of a paper are not available. We turn now to the evaluation of models when keywords are not available at test time. We compare our multi-task approach (§4.3) with the models trained with concatenating the keywords in the input sequence (during training) but tested only on the input sequence without keywords. The motivation behind this comparison is to understand the difference in performance of the models which leverage keywords during training in a manner different from our multi-task models but not at test time (same as the models trained with our multi-task approach). These results are shown in Table 5. We found that:

Multi-task learning effectively makes use of keywords for topic classification. A first observation is that *not making use* of gold (author-specified) keywords at test time (but only during training KW^{tr} through concatenation using Eq. 1) decreases performance (see Table 4 bottom half and Table 5 top half). Remarkably, the multi-tasking models (which also do not use gold keywords at test time) are better at classifying the topics than the models that use keywords only during training through concatenation. In addition, comparing the models that do not use keywords at all and the multi-task models (top half of Table 4 and bottom half of Table 5), we can see that the multi-task models perform better. Furthermore, the performance of the multi-tasking models is only slightly worse compared with that of the models that use gold (author-specified) keywords both during train and test (see bottom halves of Tables 4 and 5). These results indicate that the models trained with our multi-task learning approach learn better representations of the input text which help improve the classification performance, thereby harnessing the usefulness of author-specified keywords even in their absence at test time.

6 Analysis and Discussion

From our experiments, it is evident that all of our hierarchical baselines can outperform their flat counterparts. But it is not clear whether the performance gain comes from using the hierarchy to better learn the parent-child dependency or it is because we allow the models to focus on each class individually by training one-vs-all binary classifiers in our hierarchical setting as opposed to one flat model for all the classes. In addition, our experiments also show that keywords can be used in multiple ways to improve topic classification performance. However, it is unclear whether or not keywords *by themselves* can achieve the optimal performance. Thus, we analyze our models in these aspects with the following experiments.

Hierarchical vs. n-Binary We conduct an experiment with SciBERT where we train a binary classifier for each class similar to the hierarchical SciBERT model but do not initialize it with its parent's model parameters, i.e., we do not make use of the topic hierarchy. We compare the performance

Approach	Micro F1	Macro F1
HR-SciBERT with KW	53.04%	34.61%
n-Binary-SciBERT with KW	53.01%	32.44%

Table 6: Comparison of performance between hierarchical SciBERT and n-binary SciBERT models which do not learn the parent-child relationships.

of this *n-binary-SciBERT* model with *HR-SciBERT* model in Table 6.

We can see that the non-hierarchical approach with n binary models has more than 2 percentage points lower Macro F1. The performance of deep learning models depends partly on how their parameters are initialized (Bengio et al., 2017). For the *n-binary* approach, since we initialize the model parameters for each class with a SciBERT model pre-trained on unsupervised data, it is forced to learn to distinguish between the examples belonging to this class and the examples from all other classes from scratch. In contrast, when the model parameters for a node in the topic hierarchy are initialized with its parent node's trained model (for HR models), we start with a model which already knows a superset of the distinct characteristics of the documents belonging to this node (i.e., the characteristics of the papers which belong to its parent node). In other words, the model does not need to be trained to classify from scratch. Therefore, the hierarchical classification setup acts as a better parameter initialization strategy which leads to a better performance.

With Keywords vs. Only Keywords We experiment with flat BiLSTM, BERT and SciBERT models with only keywords as the input. A comparison of these *only keywords models* with the models which use title, abstract and keywords can be seen in Table 7. We can see a decline of $\approx 12\%$, $\approx 8\%$ and $\approx 5\%$ in Macro F1 for BiLSTM, BERT and SciBERT, respectively, when only keywords are used as the input. Therefore, we can conclude that keywords are useful in topic classification but that usefulness is evident when other sources of input are also available.

7 Conclusion

In this paper, we introduce **SciHTC**, a new dataset for hierarchical multi-label classification of scientific papers and establish several strong baselines. Our experiments show that **SciHTC** presents a challenging benchmark and that keywords can play a vital role in improving the classification performance. Moreover, we propose a multi-task learning framework for topic classification and keyword labeling which improves the performance over the models that do not have keywords available at test time. We believe that **SciHTC** is large enough for fostering research on designing efficient models and will be a valuable resource for hierarchical multi-label classification. In our future work, we will explore novel approaches to further exploit the topic hierarchy and adopt few-shot and zero-shot learning methods to handle the extremely rare categories. We will also work on creating datasets from other domains of science with similar characteristics as SciHTC to allow further explorations.

Approach	Micro F1	Macro F1
Flat-BiLSTM with KW	47.78%	27.40%
Flat-BiLSTM only KW	33.63%	15.87%
Flat-BERT with KW	51.84%	30.64%
Flat-BERT only KW	48.14%	22.95%
Flat-SciBERT with KW	52.22%	32.41%
Flat-SciBERT only KW	49.38%	27.18%

Table 7: Comparison of performance of flat BiLSTM, BERT and SciBERT trained and tested with only keywords (no title and abstract) vs. trained and tested with title+abstract+keywords.

8 Limitations

One limitation of our proposed dataset could potentially be that all of our papers are from the computer science domain and therefore, it does not provide coverage to papers from other scientific areas. However, we see this as a strength of our dataset rather than a weakness. There are other datasets already available which cover a diverse range of scientific areas (e.g., WoS). In contrast, we address the lack of a resource which can be used to study hierarchical classification among fine-grained topics, with potential confusable classes. SciHTC can be used as a benchmark for judging the models' ability in distinguishing very subtle differences among documents belonging to closely related but different topics which will lead to development of more sophisticated models.

Acknowledgements

This research is supported in part by NSF CAREER award #1802358, NSF CRI award #1823292, NSF IIS award #2107518, and UIC Discovery Partners Institute (DPI) award. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF and DPI. We thank AWS for computational resources. We also thank our anonymous reviewers for their constructive feedback.

References

Simon Baker and Anna Korhonen. 2017a. Initializing neural networks for hierarchical multi-label text classification. In *BioNLP 2017*, pages 307–315, Vancouver, Canada,. Association for Computational Linguistics.

Simon Baker and Anna-Leena Korhonen. 2017b. Initializing neural networks for hierarchical multi-label text classification. Association for Computational Linguistics.

Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6295–6300, Florence, Italy. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Yoshua Bengio, Ian Goodfellow, and Aaron Courville. 2017. *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *International Conference on Machine Learning*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, page 89–98, New York, NY, USA. Association for Computing Machinery.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, , Matthew S Gerber, and Laura E Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. IEEE.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, San Diego, California. Association for Computational Linguistics.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3(2):127–163.

Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. 2016. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57:232–247.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1063–1072, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.

António Paulo Santos and Fátima Rodrigues. 2009. Multi-label hierarchical text classification using the acm taxonomy. In *14th Portuguese Conference on Artificial Intelligence (EPIA)*, volume 5, pages 553–564. Springer Berlin.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 243–246, New York, NY, USA. Association for Computing Machinery.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and CNN-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 592–598, Melbourne, Australia. Association for Computational Linguistics.

Pengcheng Yang, Fuli Luo, Shuming Ma, Junyang Lin, and Xu Sun. 2019. A deep reinforced sequence-to-set model for multi-label classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5252–5258, Florence, Italy. Association for Computational Linguistics.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

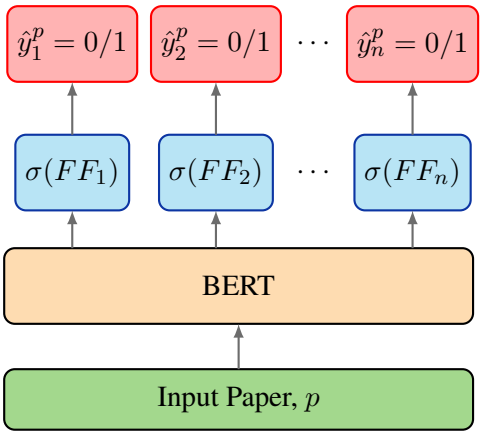

Figure 3: Architecture of our flat multi-label classification baseline using BERT. Here, BERT is the only shared layer for all topics. FF_t, σ and $\hat{y}_t^p$ denote the feed forward layer for topic t, sigmoid activation function and prediction indicating whether topic t is relevant for the input paper p or not (1 or 0), respectively.

A Label Distribution

We can see the explicit label distribution showing the number of topics belonging to each topic up to level 2 of the category hierarchy tree in Table 9.

B Flat & Hierarchical Model Architectures

Figure 3 illustrates the architecture of our flat multi-label classification baselines. Here, we show BERT as the encoder to avoid clutter but we also use BiLSTM, XML-CNN and SciBERT as encoders as we describe in Section 4. We can see that the encoder is shared by all topics and there is one feed-forward layer for each topic, $t = 1, 2, ..., n$. A sigmoid activation is applied to the feed-forward layers' output to predict whether each corresponding topic is relevant to an input paper or not (1 or 0).

We can also see an example of leveraging topic hierarchy to learn inter-label dependencies in Figure 4. Here, all models are binary classifiers for a single label from the topic hierarchy. θ_a, θ_b and θ_c represent the model parameters of topics a, b and c, respectively where a is the parent topic of b and c in the hierarchy tree. Both θ_b and θ_c are initialized with θ_a to encode inter-label dependencies and then fine-tuned to predict whether topic b and topic c are relevant to an input paper or not.

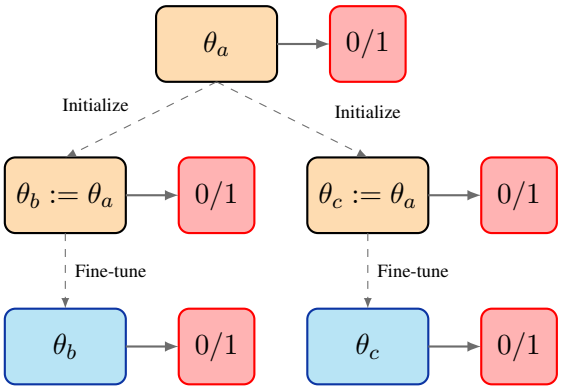

Figure 4: Leveraging topic hierarchy to learn inter-label dependencies. Here, θ_a, θ_b and θ_c are the model parameters of a parent class a and its children b and c, respectively. The child models θ_b and θ_c are initialized with the parent model θ_a and then fine-tuned on the data at child nodes b and c, respectively.

C CNN Models and Results

We follow the XML-CNN architecture proposed by Liu et al. (2017), which consists of three convolution filters on the word embeddings of the input text. The outputs from the convolution filters are pooled with a certain window. The pooled output then goes trough a bottleneck layer where the dimensionality of the output is reduced to make it computationally efficient. The output from the bottleneck layer is then sent to the output layer for topic prediction.

Note that Bi-LSTM, BERT and SciBERT give a contextual representation for every word in the input text which can be used for sequence labeling. This is not necessarily true for CNN. To ensure we have a representation of every word in the input text from CNN filters, the filter sizes are selected in such a way that the number of output vectors match the length of the input text, as presented in (Xu et al., 2018). Having a corresponding representation for each token is necessary for our multi-task objective. We can see the results of this model in Table 8.

D Implementation Details

We started pre-processing our data by converting title, abstract and keywords to lower case letters. Then, we removed the punctuation marks for the LSTM and CNN models. The text was tokenized using the NLTK tokenizer.[11] After tokenizing the

[11] https://www.nltk.org/api/nltk.tokenize.html

Approach		Precision	Recall	Micro-F1	Macro-F1
Flat-XML-CNN	w/o KW	23.75 ± 1.08	28.77 ± 0.50	45.18 ± 0.07	24.73 ± 0.21
Flat-XML-CNN	with KW	25.73 ± 0.93	31.66 ± 1.32	48.22 ± 0.29	27.31 ± 0.56
Flat-XML-CNN	with KW^{tr}	23.97 ± 0.34	27.68 ± 1.47	45.54 ± 0.44	24.84 ± 0.58
Flat-XML-CNN	Multi-Task	22.13 ± 1.04	26.97 ± 0.76	44.91 ± 0.40	23.13 ± 0.17
HR-XML-CNN	w/o KW	24.11 ± 0.53	30.78 ± 1.55	45.12 ± 0.04	26.25 ± 0.71
HR-XML-CNN	with KW	24.81 ± 0.37	33.01 ± 0.22	46.15 ± 0.17	27.58 ± 0.14
HR-XML-CNN	with KW^{tr}	24.75 ± 0.31	27.57 ± 0.59	44.01 ± 0.54	25.24 ± 0.02
HR-XML-CNN	Multi-Task	23.70 ± 0.67	31.97 ± 0.18	44.79 ± 0.37	26.40 ± 0.42

Table 8: Comparison of performance among different CNN based models. Here, HR - hierarchical, KW - keywords. The superscript ^{tr} on KW^{tr} indicates that the keywords were used only during training.

text, we stemmed the tokens using Porter stemmer.[12] Finally, we masked the words which occur less than two times in the training set with an $< unk >$ tag. The rest of the unique words were used as our vocabulary.

We address the imbalance of classes in our data by assigning the following weights to the examples of the positive class while training our CNN and LSTM based hierarchical classifiers: 1,3,5,10,15...40. The best weight was chosen based on the model's F1 score on the validation set. However, for the flat multi-label classifiers, we could not try this method because finding the optimal combination of weights would take exponential time. We also did not try this approach for hierarchical BERT and Sci-BERT because they are already very time consuming and expensive. We tuned the sigmoid thresholds from $0.1 - 0.9$ on the validation data and the thresholds with the highest performance scores were chosen for every class separately. We tune the loss scaling parameters $[\alpha, \beta]$ for our multi-task objective with the following values: $[0.3, 0.7], [0.4, 0.6], [0.5, 0.5], [0.6, 0.4], [0.7, 0.3], [1, 1]$ on the development set and found that the models show the best performance with $[1, 1]$.

For all our experiments, the maximum lengths for input text (title+abstract) sequence and keywords sequence was set to 100 and 15 respectively. We used pre-trained 300 dimensional Glove[13] embeddings to represent the words for LSTM and CNN based models. The hidden state size for the bidirectional LSTMs were kept at 72 across all our models. The fully connected layer after the bi-LSTM layer has size 16 for the hierarchical models and 72 for the flat models. We tried to keep them both at size 16. However, the flat LSTM model showed very unstable performance with a hidden layer of size 16. The filter sizes for the XML-CNN was chosen as 3, 5 and 9 and the number of filters for each of the sizes were set at 64. The input text was padded by 1, 2 and 4 units for each of the filter windows, respectively. The pooling window was set at 32 and the bottleneck layer converted the pooled output to a vector of length 512.

We used binary cross-entropy as our loss functions for both classification and keyword labeling tasks in all our models. Adam optimizer (Kingma and Ba, 2014) was used to train the models with mini-batch size 128. Except the transformer based models, the initial learning rate for all other models was kept at 0.001. For BERT and Sci-BERT, the learning rate was set to $2e^{-5}$. The hierarchical LSTM and CNN based models were trained for 10 epochs each. We employed early stopping with patience equal to 3. On the other hand, flat-LSTM and flat-XML-CNN models were trained for 50 epochs with patience 10. The flat and hierarchical transformer based models were fine tuned for 5 and 3 epochs, repectively. We ran our experiments on NVIDIA Tesla K80 GPUs. The average training time was 2 days for the hierarchical LSTM models and additional ∼24 hours with the multi-task approach. Hierarchical CNN models took ∼24 hours to train with additional 4/5 hours more with the multi-task approach. The flat models took less than 1 hour to train for both LSTM and CNN. The flat transformer based models took ∼14 hours to train on one GPU. We used 8 of the same NVIDIA Tesla K80 GPUs to train the hierarchical transformer based models. It took ∼ 6 days to train all 83 binary models.

[12] https://www.nltk.org/howto/stem.html

[13] https://nlp.stanford.edu/projects/glove/

Category	Level	Count		
		TRAIN	TEST	DEV
General and reference	1	2807	365	346
Hardware	1	6889	853	866
Computer systems organization	1	4758	630	629
Networks	1	8853	1102	1081
Software and its engineering	1	19687	2418	2518
Theory of computation	1	6948	824	858
Mathematics of computing	1	3147	410	415
Information systems	1	25445	3167	3198
Security and privacy	1	4640	567	609
Human-centered computing	1	26309	3303	3242
Computing methodologies	1	20434	2602	2493
Applied computing	1	10491	1320	1247
Social and professional topics	1	8520	1055	1114
Document types	2	488	57	59
Cross-computing tools and techniques	2	1823	239	223
Communication hardware, interfaces and storage	2	1016	123	131
Integrated circuits	2	1571	197	179
Very large scale integration design	2	561	82	73
Power and energy	2	11	0	1
Electronic design automation	2	1279	161	167
Hardware validation	2	826	107	98
Hardware test	2	392	37	48
Robustness	2	289	31	35
Emerging technologies	2	420	56	65
Architectures	2	1941	251	251
Embedded and cyber-physical systems	2	1305	173	180
Real-time systems	2	307	48	45
Dependable and fault-tolerant systems and networks	2	361	47	59
Network architectures	2	1179	142	132
Network protocols	2	1938	243	213
Network components	2	140	14	17
Network algorithms	2	16	3	4
Network performance evaluation	2	26	5	3
Network properties	2	337	41	39
Network services	2	873	111	126
Network types	2	2561	340	326
Software organization and properties	2	4964	614	586
Software notations and tools	2	7018	901	923
Software creation and management	2	5601	649	735
Models of computation	2	270	32	21
Formal languages and automata theory	2	186	25	28
Computational complexity and cryptography	2	189	14	23
Logic	2	642	72	74
Design and analysis of algorithms	2	2444	304	291
Randomness, geometry and discrete structures	2	960	111	131
Theory and algorithms for application domains	2	426	45	48
Semantics and reasoning	2	1001	129	131
Discrete mathematics	2	670	89	82
Probability and statistics	2	665	93	89
Mathematical software	2	214	23	35
Information theory	2	576	77	64
Mathematical analysis	2	653	77	84
Continuous mathematics	2	87	8	14
Data management systems	2	2408	319	310
Information storage systems	2	1159	149	132
Information systems applications	2	8139	995	972
World Wide Web	2	1514	191	201
Information retrieval	2	8697	1060	1124
Cryptography	2	435	37	42
Formal methods and theory of security	2	10	2	3
Security services	2	505	46	65
Intrusion/anomaly detection and malware mitigation	2	293	43	37
Security in hardware	2	30	9	5
Systems security	2	1021	120	115
Network security	2	499	68	74
Database and storage security	2	46	3	4

Software and application security	2	78	8	7
Human and societal aspects of security and privacy	2	94	11	12
Human computer interaction (HCI)	2	13714	1777	1675
Interaction design	2	1079	107	140
Collaborative and social computing	2	2543	312	324
Ubiquitous and mobile computing	2	40	4	6
Visualization	2	127	15	14
Accessibility	2	62	7	10
Symbolic and algebraic manipulation	2	478	70	75
Parallel computing methodologies	2	554	69	75
Artificial intelligence	2	8117	1002	986
Machine learning	2	2567	312	314
Modeling and simulation	2	2123	273	274
Computer graphics	2	3903	513	459
Distributed computing methodologies	2	65	14	10
Concurrent computing methodologies	2	319	46	56
Electronic commerce	2	287	38	42
Enterprise computing	2	438	49	48
Physical sciences and engineering	2	979	123	121
Life and medical sciences	2	1419	189	171
Law, social and behavioral sciences	2	1308	159	145
Arts and humanities	2	943	113	106
Computers in other domains	2	1219	161	139
Operations research	2	264	38	28
Education	2	1495	191	180
Document management and text processing	2	795	97	95
Professional topics	2	5898	736	775
Computing / technology policy	2	1114	161	153
User characteristics	2	316	34	37

Table 9: Category name vs paper count up to level 2 of the CCS hierarchy tree in SciHTC.