MTLV: A LIBRARY FOR BUILDING DEEP MULTI-TASK LEARNING ARCHITECTURES

by

Fatemeh Rahimi

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2021

I dedicate this to all women in the computer science community who believed in themselves, worked hard, and fought, to be in the place they are today. You all are true inspirations.
I would also like to dedicate this to the 176 victims of flight PS752, especially the 46 students who were looking forward to their future when they perished. May they rest in peace.

Table of Contents

List of Tables

List of Figures

List of Abbreviations Used

Acronyms

BERT Bidirectional Encoder Representations from Transformers

GB Gigabyte

GMHL Grouped Multi-Head Learning

GMTL Grouped Multi-Task Learning

MLM Masked Language Modeling

MTLV Multi-Task Learning Visualizer

MTL Multi-Task Learning

NLP Natural Language Processing

NSP Next Sentence Prediction

STL Single Task Learning

UI User Interface

Abstract

Multi-Task Learning (MTL) for text classification takes advantage of the data to train a single shared model with multiple task-specific layers on multiple related classification tasks to improve its generalization performance. We choose pre-trained language models (BERT-family) as the shared part of this architecture. Although they have achieved noticeable performance in different downstream NLP tasks, their performance in an MTL setting for the biomedical domain is not thoroughly investigated. In this work, we investigate the performance of BERT-family models in different MTL settings with Open-I (radiology reports) and OHSUMED (PubMed abstracts) datasets. We introduce the MTLV (Multi-task learning visualizer) library to facilitate building Multi-task learning-related architectures. This library uses existing infrastructure (e.g., Hugging Face Transformers and MLflow Tracking) to allow users to build and compare multi-task, multi-head, and single-task learning designs using available models of the Transformers library. Following previous work in the computer vision domain, we clustered tasks in few groups and trained each group separately on separate models (Grouped Multi-Task Learning (GMTL)) and a single model with different heads (Grouped multi-Head Learning (GMHL)) where each head includes a group of tasks. Contextual representation of class labels (Tasks) and their descriptions was used by the library as features to cluster the tasks. The set of models from GMTL are trained separately, each for the set of tasks(groups) that is arrived from task clustering. We experimented with a set of binary classification tasks that share the same dataset (multi-label classification). The contributions of this research are: (a) We observed that grouping tasks for training with few models (GMTL) outperforms both the multi-task (MTL) and multi-head learning settings (GMHL); (b) We proposed an approach to use task (label) names and their description embeddings as clustering feature of tasks; (c) Although GMTL have similar performance compared to Single Task Learning (STL), GMTL is computationally less expensive than the STL setting where a separate model is trained for each task; (d) Code of the MTLV library is available as open-source on GitHub(`https://github.com/fatemerhmi/MTLV`)

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Evangelous Milios, for his mentorship and continuous guidance during my master's study and thesis. Furthermore, I am grateful to my co-supervisor Dr. Stan Matwin for his great support and guidance.

Many thanks to my colleagues in the MALNIS group, whom I learned so much from and their inspiring research environment. I would like to especially thank Dr. Martha Dais Ferreira, who has guided me throughout my thesis research with her great comments and the time she dedicated to meeting me discussing my challenges. Many thanks to Jeniffer David, Maksym Taranukhin, and Farshid Verno whom I had the great pleasure of working with moreover receiving guidance and support during my research projects.

Thanks to all of my family and friends for their continuous support.

Finally, special thanks to my brother Reza, who always believed in me and encouraged me to pursue science.

Chapter 1

Introduction

Multi-task Learning (MTL) is a learning paradigm where multiple tasks are being learned jointly, usually with a single network. Learning a set of tasks together is effective if the alignment between the tasks is high such that it is improving the overall learning process (positive transfer). However, it is often observed that in MTL settings, task objectives interfere, and the performance suffers (negative transfer).

Classification is one of the most well-explored supervised machine learning techniques over the past several years. One of its forms is the multi-label classification, where each instance may be associated with one or multiple classes. One of the Multi-label classification challenges [21] is the volume of a dataset which refers to the number of labels. The two challenges it brings during an experiment are; first, the space of output labels grows, and the second one is the label imbalance issue. Multi-label classification is an example of multi-task learning in which a set of binary classification tasks share the same dataset.

One of the Multi-task learning architectures often used is hard parameter sharing which means that some layers of the neural network is shared between all the tasks. Since pre-trained language models (BERT-family models) have proven to be effective for a wide variety of natural language processing (NLP) tasks, they are considered a suitable architecture for the shared part of multi-task learning architectures. Although BERT was trained on general domain text, other variations of it such as BioBERT[9] and BlueBERT[14] were trained on more specialized context such as biomedical and clinical domain.

One of the typical architectures of multi-task learning for hard-parameter sharing is multi-head architecture [15] which includes a set of shared layers and a couple of heads on top of the model that aims to learn different tasks. This architecture has been widely used in different research domains, especially in natural language processing where the shared layers are replaced with pre-trained language models

(i.e., BERT) [13, 11, 16]. Observing the effectiveness of using pre-trained language models as the shared layers of multi-task and multi-head architectures, we also follow the same footsteps.

Our motivation behind investigating Multi-task learning architectures is that not only are they more computationally efficient comparing to single-task learning (especially when the number of tasks grows), but they also lead to positive transfer when the correlated set of tasks share their features. The study of how to group tasks can be viewed as an extension of clustering algorithms to the task level[22].

Previous work in computer vision [17] assigns tasks to few neural networks such that each model learns a set of cooperating tasks. Inspired by that work, we developed a python library called MTLV (Multi-task learning visualizer) that implements a set of multi-task learning architectures and can visualize their learning process throughout their run. The visualization component is taking advantage of an existing infrastructure called MLflow Tracking to keep track of learning process of different runs and let user compare them at ease.

We developed four main architectures in the MTLV framework, which are as follows: (1) Single-task learning (STL), where each binary classification is being learned with separate models ending up with the same number of models as the tasks; (2) Multi-Task Learning (MTLV) where all the tasks are being learned jointly together with a single model; (3) Grouped Multi-Head Learning (GMHL) architecture where we cluster tasks to few groups and place each group in a head-on top of a single model similar to multi-head learning architecture; (4) Grouped Multi-Task Learning (GMTL) which cluster the tasks similar to GMHL but learn them with few separate models where each model aims to learn a group of tasks that are most related to each other.

1.1 Contributions

This thesis investigates the positive or negative transfer of learning a set of binary classification tasks jointly in different multi-task learning architectures using the developed framework (MTLV). MTLV uses existing infrastructure (e.g., Hugging Face

Transformers[1] and MLFlow tracking[2]) to facilitate the process of training multi-task learning architectures and also being able to query, investigate their performance and compare their learning.

The contribution of this work can be summarized as follows.

- Evaluate BERT-family language models learning in different domains such as biomedical and clinical context.
- Compared the learning process of three deep multi-task learning architectures (MTL, GMTL, and GMHL) to single-task learning(STL).
- Analyse the computation cost of GMTL to STL models in addition to their overall performance.
- proposed a clustering approach for binary text classification tasks that uses tasks context features (embeddings) of label names and their descriptions as task features.
- Code and documentation of the MTLV library is publicly available to let users build their variation of proposed models and also designed to easily integrate other models, datasets, optimization functions, and task clustering algorithms.

[1] `https://github.com/huggingface/transformers`
[2] `https://github.com/mlflow/mlflow`

Chapter 2

Related work

The related work is divided into three parts. In the first one, we focus on multi-task learning and its similar architectures, while the second part discusses the effectiveness of pre-trained language models. Lastly, we discuss using pre-trained language models as their shared part of multi-task learning architectures, which is the main focus of this thesis.

2.1 Multi-Task Learning

Multi-task Learning (MTL) helps to improve the generalization of a model by leveraging the information in the training sets of the related datasets [22]. MTL has proven to be helpful in various applications such as computer vision, health informatics, speech, and especially Natural Language Processing (NLP) [3] for a long time.

MTL has also proven to help the model to generalize more. A multi-task learning framework for sentence representations was presented and showed that learning-related tasks (including skip-thought training, machine translation, entailment classification, and constituent parsing) jointly results in good generalization [19].

MTL contains mainly two techniques [15] of hard parameter sharing and soft parameter sharing [10]. The former is when the shared part of the model is completely shared between all the tasks, while the latter is more flexible and partially shared across tasks.

MTL default setting is homogeneous-feature MTL [16] where it is considered to consist of only one type of task [22]. The opposite to homogeneous-feature MTL is heterogeneous-feature, consisting of different types of supervised tasks such as classification and regression problems [11, 13, 19].

It is often observed that tasks interfere with each other, and the model's learning leads to a negative transfer phenomenon. One way to prevent this behaviour is to find correlated tasks using task clustering algorithms which is just an extension of

clustering algorithms to the task level rather than the data level. Which tasks should and should not be trained together in one network when employing multi-task learning has been investigated in [17]. That work also provides an empirical study of several factors that influence multi-task learning, including network size, dataset size, and how tasks influence one another when learned together. They also explore which tasks should be learned together by proposing a framework in which they assign tasks to a few neural networks such that cooperating tasks being learned by the same network.

2.2 Pretrained language models

Pre-trained language models such as BERT (Bidirectional Encoder Representations from Transformer) have shown effectiveness in various NLP tasks such as classification, sentiment analysis, question-answering, and named entity recognition. BERT [5] was trained in a self-supervised multi-task objective (two supervised objectives), which included a masked language modelling (MLM) head and next sentence prediction (NSP). For the MLM task, they randomly hide 15% of words in a sentence and train BERT to predict the missing words, while for NSP, the task aims to predict whether two sentences are consecutive or not (binary classification).

Although the BERT language model was trained on Wikipedia and Book Corpus (total of 16GB text), its performance suffers in biomedical and clinical context due to the word distribution shift from the general domain to the more technical corpus. BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining) [9] is a domain-specific language representation model pre-trained on large-scale biomedical corpora that outperforms BERT on a variety of biomedical text mining tasks such as biomedical named entity recognition, biomedical relation extraction, and biomedical question answering. BioBERT includes five versions that are pre-trained on either PubMed, PMC, or both of them.

Another BERT-family model is BlueBERT[14] which uses the pre-trained BERT and continues to pre-train the model on PubMed abstracts and MIMIC-III clinical notes.

2.3 Multi-task learning meets pre-trained language models

BERT's effectiveness on the GLUE benchmark compared to its previous SOTA hugely impacted transfer learning research in the NLP domain. Recent work on multi-task learning has also taken advantage of the BERT-family models and has successfully surpassed other models on the GLUE benchmark. BERT and PALS [18] outperformed the BERT model by training all the GLUE benchmark tasks jointly. BAM [2] used the BERT model in MTL settings differently and used knowledge distillation combined with teacher annealing technique. Multi-Task Deep Neural Network (MT-DNN) argued that language model pretraining and MTL are complementary technologies and can learn better representation when combined [11]. MT-DNN also boosted the performance of various NLU tasks using multi-task objectives.

BERT-family models have shown effectiveness in MTL architectures when used in the biomedical domain. BERT was used as the shared part of a multi-task learning design [13] to learn multiple biomedical and clinical NLP tasks jointly. That work outperformed BERT on text similarity and relation extraction in the biomedical and clinical context. CheXbert [16] is another example that uses BERT in their MTL setting to predict pathologies for a large corpus of radiology reports (classification task) with 14 heads on top of models of the BERT family.

Following the above work, we also used BERT-family models to evaluate our proposed approaches, namely GMHL and GMTL. We evaluated our work with a biomedical dataset, OHSUMED [6], and a clinical one Open-I [4] which are both publicly available. OHSUMED is a subset of the MEDLINE database, a bibliographic database of peer-reviewed medical papers, while Open-I is a free-text radiology report dataset along with a set of pathologies as their labels. The instances of these two datasets might be associated with one or multiple classes (multi-label), which can be interpreted as a set of binary classifications that share the same dataset. We evaluate how our grouping labels(binary classification tasks) are impacted when they are trained all together (MTL), trained in a set of groups (GMTL, GMHL), or individually (STL).

Chapter 3

Methodology

In this research, we developed a library that lets users build and train different multi-task learning architectures while also comparing their learning process. The four architectures that the MTLV library supports are: Single-Task Learning (STL), Multi-task learning (MTL), Grouped Multi-Head Learning (GMHL) and Grouped Multi-Task Learning (GMTL) learning. Single task learning aims to learn each task separately using a single model, while multi-task learning aims to learn a set of tasks with one neural network. GMHL and GMTL are two architectures based on multi-task learning, the goals of which are to cluster tasks from similar groups to learn in the same single model (GMHL) or with few models per group (GMTL). In the following sections, we define these models in detail and explain their architectures.

3.1 Single-Task Learning (STL)

In order to describe the problem, we first define the task and single-task learning.

Definition 1. Task. A supervised task $\mathcal{T}_i$ is defined as $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathscr{L}_i\}$, such that $\mathbf{x}$ is the inputs and $\mathbf{y}$ is the corresponding targets (labels). Different tasks may vary based on input distribution ($p_i(\mathbf{x})$), the label distribution given the input ($p_i(\mathbf{y} \mid \mathbf{x})$), or the loss function ($\mathscr{L}_i$).

Definition 2. STL (Single-Task Learning). Given $\mathcal{D}$ a supervised dataset that has n input and output pairs of x and y defined as $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})_n\}$. The single-task learning goal is to minimize a loss function of with respect to the neural network parameters θ ($\min_\theta \mathscr{L}(\theta, \mathcal{D})$).

Solving a binary or multi-class text classification problem with a pre-trained language model is an example of single-Task learning (STL) model. In Fig 3.1, an example of such is depicted where a pre-trained language model is used to learn representations of Sentence (Sequence) input S and classify the label with an additional layer on top of a model (classification layer).

In the setting, where there are m tasks to learn, m separate models are trained separately to learn these tasks. This design decision can become computationally very expensive as the number of tasks grows.

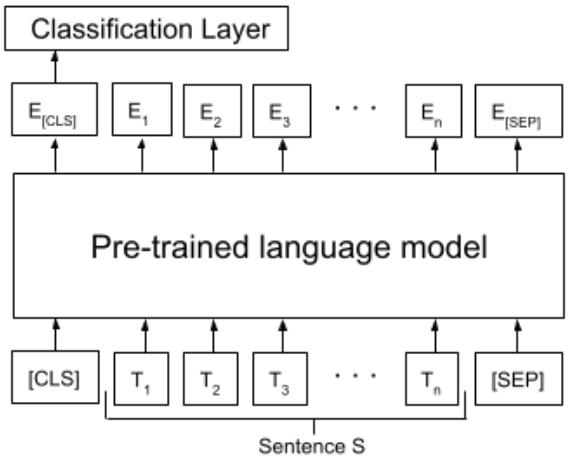

Figure 3.1: This design is a typical single task learning architecture (inspired by BERT [5]) for solving a binary or multi-class classification problem, where a single layer is added on top of the pre-trained language model to perform as a classification layer. A set of tokens (T1 to Tn) from the input sentence (sequence of words) is fed into a pre-trained language model. In the case of BERT-family models, the CLS and SEP tokens are added at the beginning and end of the input, respectively. The CLS token is a classification token representing the whole sentence, while SEP marks the end of the sentence. The output of the pre-trained model is the sequence of token embeddings (E[CLS], E1, E2, ..., En, E[SEP]). The first output, i.e. the CLS embedding, is then given to the classification layer on top of the model to perform binary/multi-class classification.

3.2 Multi-Task Learning (MTL)

Definition 3. MTL(Multi-Task Learning) Given m supervised tasks $\{\mathcal{T}_i\}_{i=1}^{m}$ multi-task learning aims to improve the learning of a model for each task $\mathcal{T}_i$, by using the knowledge learned jointly from all the m tasks.

According to the MTL definition, we can learn either different types of supervised tasks (e.g. classification and regression problems) or only one type of task. The first is called heterogeneous MTL, while the latter is homogeneous MTL(default multi-task setting). This work focuses on homogeneous MTL for classification tasks where all these tasks share the same dataset. The setting where some binary classification tasks share the same data is often referred to as Multi-label classification.

The multi-label problem is an example of a multi-task learning problem, where, given a sentence or sequence S, each label represents a different binary classification task. Usually, a single model (MTL) or series of models (STL) predicts the positive or negative label of each task. One of the Multi-label dataset challenges is the volume problem which refers to the number and imbalance of output labels. It is often the case to encounter an imbalanced dataset in real-world scenarios, which means that the number of instances associated with some labels is much smaller (greater) than the number of instances assigned to others.

Unfortunately, the design decision for a multi-task architecture can become challenging due to the variety of design choices. Because of the dependency of MTL on the relatedness of tasks, architecture, and size of the network, it is often observed that the same architecture might work for one set of tasks but not necessary for another. This work proposes an open-source framework (MTLV) to easily let researchers experiment with a set of MTL architectures variations by simply providing a configuration file of their design choices. Furthermore, we develop MTLV in a way the researchers can easily integrate their models, optimization function, and dataset preprocessing (More details on MTLV library in Appendix B).

One of the most commonly used design choice in MTL is hard parameter sharing, where a set of hidden layers (in this case, a pre-trained language models) is shared between all tasks while having some separate layers (heads) on top of the model to learn m different tasks jointly. This architecture is referred to as multi-head architecture [15] that is depicted in Fig 3.2. This architecture design divides the neural network into two parts of shared layers and task-specific layers.

Shared layers of multi-head architecture play an essential part in this design because they enable positive transfer among correlated tasks.

A common approach to solve a multi-label classification problem (with L labels) using pre-trained language models is to add a single classification layer on top of a model (Fig 3.3), where the activation and loss Function of the last layer is sigmoid and Binary Cross entropy respectively. This architecture can be interpreted as a multi-task learning (MTL) architecture where all the binary classification datasets are being learned jointly with a single model. An example of such a model is depicted on the right-hand side of Fig 3.3.

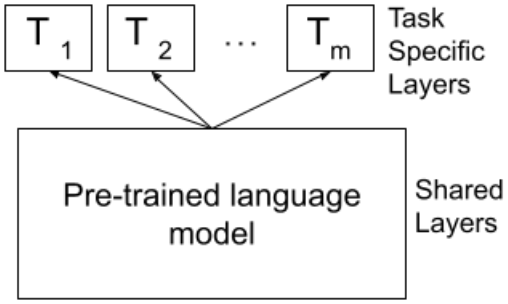

Figure 3.2: The multi-head architecture [15] is a type of MTL architecture in which a set of layers are shared between all the tasks (hard parameter sharing). The effectiveness of pre-trained language models makes them a great fit for the shared part of this network. On top of this architecture, separate heads are jointly learning m different tasks. The number of layers for the shared and task-specific heads are hyperparameters of this design.

This typical approach works as follows: (1) The input of a pre-trained BERT-family model which is a tokenized sequence of words is formed; (2) [CLS] and [SEP] are the two special tokens used to form the inputs. The classification token named [CLS] is added at the beginning of each input sequence, while [SEP] is added at the end of the input sequence. [CLS] token is the classification token which represents the context of the whole input sequence. This token is often used as a representation (embedding) of all the input tokens (sentence or sequence embedding).

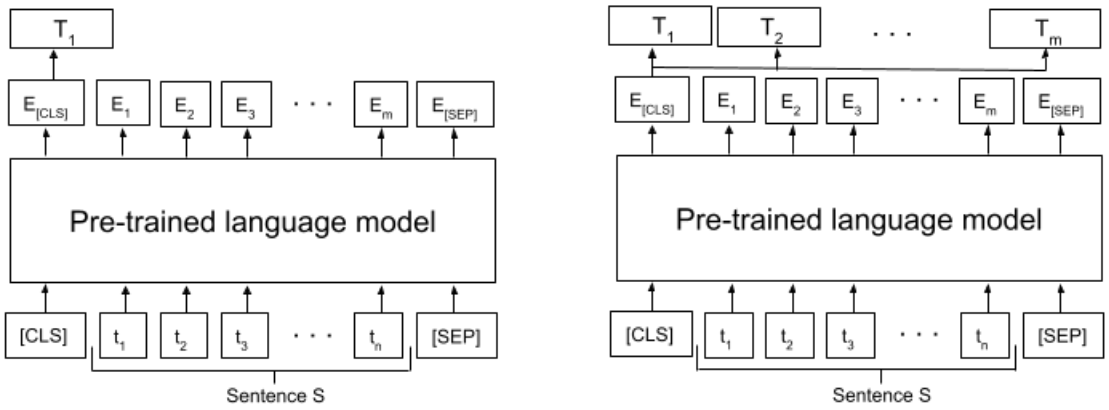

Figure 3.3: On the left, the architecture design of Single-Task Learning (STL) is depicted. The same architecture is fine-tuned separately for each task, such that m models are trained separately for m tasks. A multi-task learning (MTL) architecture is shown on the right, including a single model that learns m different tasks jointly. In the binary classification task, the architecture on the right is designed to solve a multi-label classification (MLC) task.

3.3 Grouping Multi-head Learning (GMHL)

Following the design of multi-head architecture, we introduce an architecture where each head learns a set of tasks. Grouped Multi-Head Learning (GMHL) architecture aims to learn a set of groups of tasks jointly. Our hypothesis for this architecture design is that each head will specialize in some related tasks and achieve positive transfer. An example of GMHL architecture is depicted on right-hand side of Fig 3.4.

The hyperparameters of this design are loss function, the number of layers of each head, the task groupings (which task in which head), learning rate, and which pre-trained model to use as shared layers.

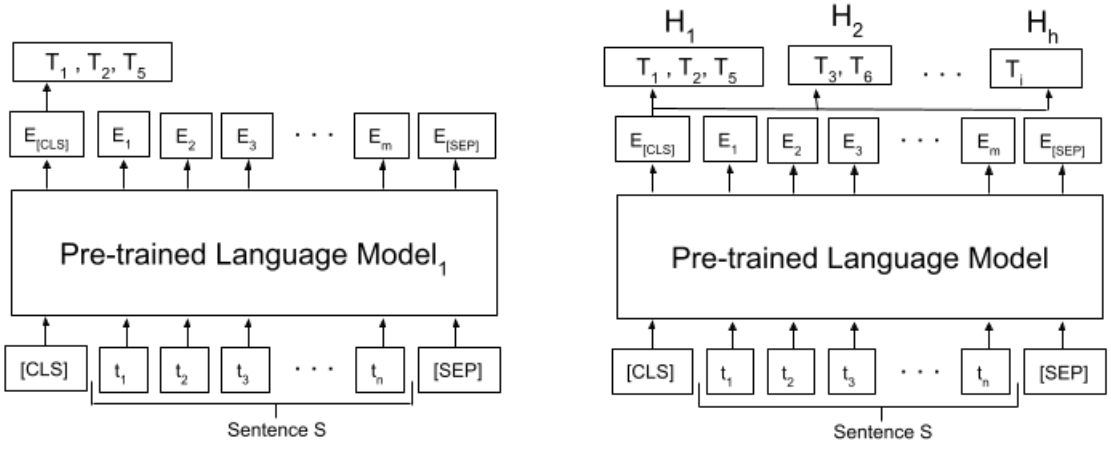

Figure 3.4: The Grouped Multi-Task Learning(GMTL) is located on the left, while the Grouped Multi-Head Learning(GMHL) appears on the right. Both of these architectures rely on a task clustering algorithm. The GMTL design is inspired from previous work in computer vision [17], where a few models (similar to the model on the left) are fine-tuned to learn a group of tasks (i.e. in this figure, the model is learning Task T1, T2 and T5). We use the same group of tasks to build a GMHL architecture, where a group of related tasks are located in each head. In GMHL, a single model is used to train all the tasks. The input of both these two designs is the CLS token embedding from the last layer of the pre-trained language model.

3.4 Grouping Multi-Task Learning (GMTL)

Following computer vision domain work [17], we design Grouping Multi-Task Learning (GMTL) architecture, in which we train few models separately for a group of tasks. The hypothesis behind designing this architecture is that each model specializes in

a set of related tasks, where learning one task can positively affect the other one and finally leads to positive transfer. A sample of this architecture with pre-trained models as the shared layers is shown in Fig 3.4.

3.5 MTLV workflow

As already described, we focus on four main architectures in this research. The two architecture of STL and MTL are the ones that are designed (depicted in Fig 3.3) to train per task or train all tasks jointly together. The hyperparameters of these two architectures are the following: number of epochs, batch size, the tokenizer parameters (maximum length, truncation, and padding), optimizer, and learning rate.

The other two architecture (GMTL and GMHL) follow a similar workflow as depicted in Fig 3.5. The MTLV library workflow includes four steps to train GMHL and GMTL models, which are as follows: (1) Clustering the tasks (2) Design, training, and optimization decision (3) Fine-tunning the model (4) Visualize learning of heads, labels, and model.

This framework simplifies the comparison of different architecture with an easy-to-use user interface visualization. The visualization that MTLV provides is the MLflow tracking system, a framework that supports the machine learning life cycle and helps the user monitor the model during training and running. During training in the backend, we log different values such as loss, F1-score micro, F1-score macro, and subset accuracy using MLflow library and later visualization it with MLflow tracking UI that shows the performance of different designs while also allowing the user to compares how each label, head, and the model is learning.

3.5.1 Clustering the tasks (labels)

Before describing the approach that this research is proposing for clustering the tasks, we first define Task Features and Task clustering.

Definition 4. Task Features. Given a supervised task $\mathcal{T}_i$, task feature vector V_i is formed to represent the task context and characteristics. Task features may be labels, data distribution or classification model properties. The task feature vector V_i is a d-dimensional vector representation ($V_i \in \mathbb{R}^d$).

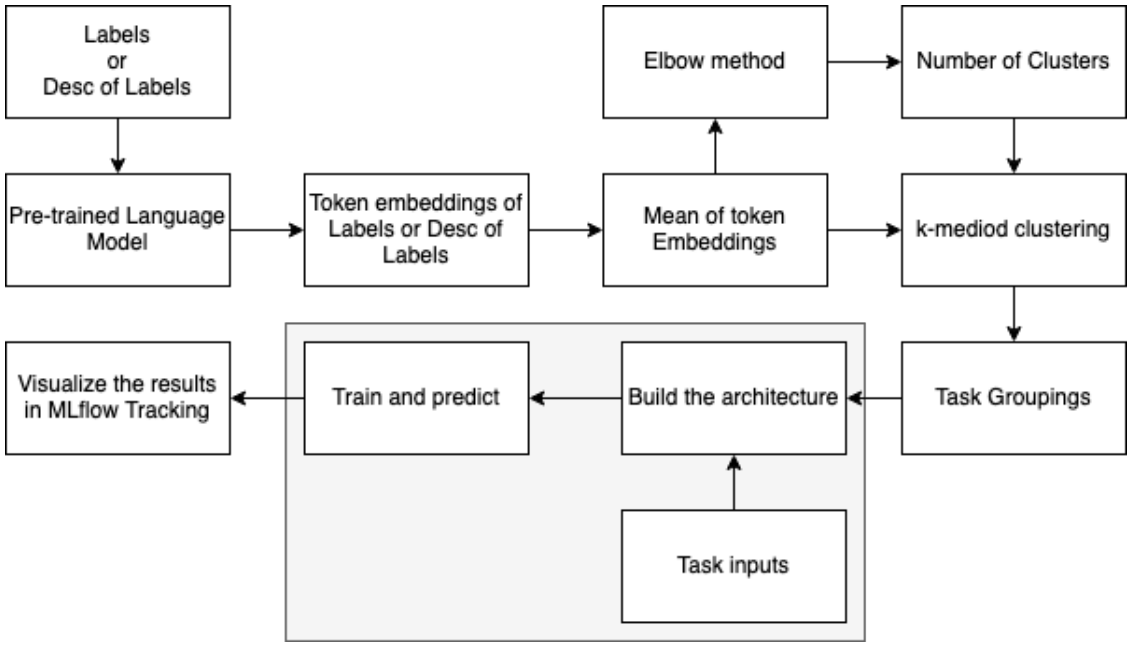

Figure 3.5: MTLV workflow for GMTL and GMHL is depicted in this flowchart. First, the label names or the description of the labels is given to a pre-trained language model (BERT-family) to extract its embeddings. Then the mean of the token embeddings is calculated and the elbow curve is plotted to help the user identify the number of clusters. The embedding of each task and the number of clusters is passed to the k-medoid clustering algorithm to cluster (group) tasks. Using the groups from the clustering algorithm to build either GMTL or GMHL architecture is depicted in gray. Once the model and its architecture are decided, it is trained on data and all the model parameters are fine-tuned. While training and predicting, we log the parameters and metrics of these architectures and later visualize them in MLflow Tracking UI.

In the context of our multi-label classification task, we address the problem as a set of binary classification tasks, where a binary classification task is defined by a single label representing the presence or absence of a specific pathology or diagnostic. We represent each task by the contextual embedding of the name of its label. The label names are first passed to a BERT-family model to extract their contextual embedding. These embeddings are the task feature vectors used for task clustering.

Our hypothesis behind using contextual embeddings is that the model learning the representation of one task can positively influence the learning of the other task in the same model while learning all the tasks with the same model can cause negative transfer if the tasks are not correlated.

The embedding of a label name consisting of multiple tokens is computed as the

average of the embeddings of all the tokens.

In general, task features for task clustering can consist of the parameters of a model (gradients [17]), an embedding vector that represents an entire task text (i.e. task2vec [1]).

One of the hyperparameters of the GMTL and GMHL architectures is which set of tasks should be learned together. According to the task clustering definition, we propose an approach to cluster the dataset labels based on their contextual embeddings as task features.

Definition 5. Task Clustering. Given m supervised tasks $\{\mathcal{T}_i\}_{i=1}^{m}$, where $\mathcal{T}_i$ is a supervised text classification task (i.e. binary classification), task clustering forms several clusters based on task feature vectors (V_i) and a given distance metric (e.g. euclidean distance) such that each cluster consists of similar tasks.

For clustering the labels(tasks), we used k-medoid clustering algorithm [8] (with euclidean distance as the distance measure), where it attempts to minimize the sum of dissimilarities of the other points in the cluster, given that in each cluster, one of the points is the medoid of the cluster. In the k-medoid algorithm, the number of clusters is a prior, making it a hyperparameter for our design decision. MTLV framework plots the elbow method for a given dataset to help users identify the number of clusters.

In natural language processing (NLP) tasks, it is sometimes clearer to the user which tasks are related by reading the corresponding documents of a task to find correlated tasks based on related documents. MTLV also allows users to cluster the labels with any given set of task groupings. We believe this feature is necessary as a human insight (expert) to this approach can help the model learn better representations and help them evaluate their hypothesis.

3.5.2 Design decisions

According to the four architectures explained in this work (STL, MTL, GMTL, and GMHL), users can choose from various configuration options. For GMHL and GMTL designs, users must choose how the framework should cluster the tasks(labels) and how many clusters. The first can be either using the label name or the label description of the binary classification task, while the latter can be decided with the elbow plot. It is also possible to provide the framework with any given set of task groupings

instead of clustering approaches.

For all the architecture decisions, the user can choose from the pre-trained language models in Table 4.2 as the shared layer of the network, and values for hyperparameters, including the number of epochs for training, batch size, tokenizer maximum length and whether to truncate the inputs.

3.5.3 Fine-tunning the model

After grouping the tasks, making design decisions and choosing the hyperparameter, the MTLV framework builds the model with corresponding heads and fine-tune all the model parameters for the downstream tasks. The last layer embedding of [CLS] token is passed as input to heads as a feature vector for classification. Both GMHL and GMTL heads have the same activation and loss function. It is sigmoid for the former and binary cross-entropy for the latter. The main difference is an extra step for GMHL to calculate the overall loss of the model. Since each head of this architecture has a separate loss, we use the overall sum loss and leave exploring the different loss functions for future work. The network gets updated with the AdamW optimizer, during training.

3.5.4 Visualize the learning

During the workflow of MTLV, we use MLflow to log parameters (key-value string types for storing hyperparameters and design decisions), metrics(key-value where the value is numeric and gets updated throughout the run (i.e. loss, F1-scores)), artifacts (i.e. the trained weights, the classification report). After the experiments are over, in the MLflow tracking UI, the user can query and compare different runs (More details on Mlflow Tracking UI in Appendix E).

Chapter 4

Experimental Results

We experimented with the four main architectures STL, MTL, GMTL, and GMHL that the MTLV framework supports and evaluated them with biomedical and clinical domain datasets. Open-I and OHSUMED are the two datasets used to compare the set of architectures from the MTLV framework to evaluate the effectiveness of learning tasks jointly in three multi-task learning architectures.

4.1 Data

We chose Open-I and OHSUMED datasets for this research experiment. OHSUMED includes peer-reviewed medical literature from the MEDLINE database, while Open-I consists of free-text radiology reports from radiologists' observations. The number of labels and number of documents for each dataset is listed in Table 4.1

dataset	#labels	Total	Train	Test
Open-I	19	3,159	2,527	632
OHSUMED	23	34,389	27,556	6,833

Table 4.1: The details of the two datasets used in this research. The split for the OHSUMED dataset is already provided with the dataset, while Open-I dataset splitting was performed using multi-label stratification method [20]. During the experiments, the validation set is always 15% of the training set. We used 20% of OHSUMED dataset, which was extracted using stratified sampling [20] to save computational cost. The total, train and test are the count of unique documents in each set.

OHSUMED includes medical abstracts from the MeSH categories of the year 1991. The task was to categorize the 23 cardiovascular disease categories. Although some subset collection of the OHSUMED dataset exists (i.e. OHSUMED-10 [7]), we used all the 23 categories from the original dataset. This collection consists of 34,389 unique documents of the medical abstracts. In this collection, there are 27,556 unique

abstracts for training and 6,833 for testing. The detailed list of how many documents exists per category and what is the category themselves are in listed in Appendix A.

The original Open-I dataset includes 3,955 radiology reports. The two most important parts of these reports, which we used as input of the models, are FINDINGS and IMPRESSION. Findings is a radiologist's note on the observation, whether it is normal or abnormal, while Impression summarizes the findings, which often includes the diagnosis; therefore, it is the most important part of the radiology report. According to their radiology reports, this dataset has 1,719 different labels, which indicate one or more pathologies for patients. We only selected the 19 top classes from this list. Three preprocessing steps for using this dataset are as follows: 1. removing the empty reports; 2. removing the duplicate reports; 3. removing the strings used for de-identification, such as "XXXX". We ended up with a total of 3,159 reports to work with. This set of reports is split with 80/20 training/test split and stratified multi-label sampling [20]. The splitting leads to 2,527 reports for the train set and 632 reports for the test set. The count and percentage of each class of the Open-I dataset are mentioned in Appendix A.

4.2 Pretrained Language Models

We chose three pre-trained language models for this research experiment listed in Table 4.2. All the models listed in the table are BERT-family models trained on different domain corpus. BERT-base is trained on the general domain, while BioBERT and BlueBERT are trained on biomedical and clinical domains. These models help us to evaluate how MTLV architectures perform using a dataset from similar contexts. The details of these three models are listed in Table 4.2

4.3 Implementation details

Our implementation is based on PyTorch and Huggingface Transformers library. We used AdamW optimizer [12] which is a variant of the Adam algorithm and learning rate of 2e-5 by following [5]. The batch size for both datasets was set to 16. All the experiments used an NVIDIA V100 SXM2 (16GB memory).

Open-I was trained with ten epochs while OHSUMED with 12 (We found out that

Model	Trained corpus	# p	case
BERT-Base	Wikipedia and book corpus	110M	uncased
BioBERT-Base v1.1	PubMed 1M	110M	cased
BlueBERT-Base	pretrained on PubMed abstracts and clinical notes (MIMIC-III)	110M	uncase

Table 4.2: A list of pre-trained models along with their trained corpus used in this research. All three models are the base version (their encoder have 12 layers) of BERT and also have the same number of parameters (110 Milion) to provide a fairer comparison. BERT-Base and BlueBERT-Base are trained on uncased corpus while BioBERT was trained on cased corpus. Although different variations of BioBERT and BlueBERT exist, we refer by these short names to the ones with settings shown in this table.

most of the architectures converged with this setting). We also truncated documents to 128 tokens for Open-I and 256 for the OHSUMED dataset based on their document token length distribution. All the experiments for a given dataset used the same train and test set for their fold for a fairer comparison. We used four-fold cross-validation for Open-I and threefold for the OHSUMED dataset.

For statistical significance testing, we used the sample of size 10 for both OHSUMED and Open-I datasets.

4.4 Open-I Results

We set up experiments for the four architectures mentioned with BERT, BioBERT and BlueBERT language models as their shared layers. We experimented with two different feature vectors for clustering the labels(binary classification tasks), namely the label name and a short description of what each label represents, embedding extracted from a BERT-family model. We also evaluate the number of clusters (3, 4, and 5) for the k-medoid clustering algorithm to cluster(group) the labels(tasks). The number of clusters are concluded from plotting the elbow method. The results are reported in Table 4.3. We only list the best performance of GMHL and GMTL settings. The full results for the Open-I dataset are in Appendix C.

We use three measurements to evaluate our approaches, which are F1-score micro(F1 micro), F1-score macro(F1 macro) and subset accuracy. F1 micro calculates

the metric globally while F1 macro calculates metric for each class and finds their unweighted mean. F1 macro and f1 micro are two measurements we use to compare the experiments as they represent the measurement better than subset accuracy, which is just an Exact Match Ratio (EMR) that does not take into account the partially correct labels and consider them as incorrect. The subset accuracy is reported in our result tables for completeness.

Architecture	Model	clustering	F1-macro	F1-micro	subsetAcc
STL	BERT	-	91.40 ± 0.79	93.30 ± 0.48	86.16 ± 0.84
	BioBERT	-	91.49 ± 0.13	93.16 ± 0.21	85.52 ± 0.74
	BlueBERT	-	93.38 ± 0.41	94.58 ± 0.67	88.06 ± 1.43
MTL	BERT	-	85.72 ± 1.66	90.43 ± 0.77	81.41 ± 1.69
	BioBERT	-	90.00 ± 1.07	92.69 ± 0.63	85.17 ± 0.90
	BlueBERT	-	90.64 ± 1.32	93.41 ± 0.82	86.63 ± 1.29
GMHL	BERT	5 (LabelDesc)	63.73 ± 5.38	82.23 ± 2.38	82.95 ± 1.95
	BioBERT	5 (LabelDesc)	79.86 ± 4.16	88.73 ± 1.16	88.97 ± 0.94
	BlueBERT	5 (LabelDesc)	76.36 ± 1.66	89.11 ± 0.11	89.16 ± 0.08
GMTL	BERT	5 (LabelDesc)	91.45 ± 0.24	93.22 ± 0.27	92.34 ± 0.36
	BioBERT	4 (LabelDesc)	92.56 ± 0.40	94.25 ± 0.31	93.76 ± 0.46
	BlueBERT	5 (LabelDesc)	93.86 ± 0.34	95.32 ± 0.15	94.40 ± 0.24

Table 4.3: We compare the four main Architectures explained in this work for the Open-I dataset. We conclude the following: (1) Both BioBERT and BlueBERT outperform BERT in different settings; (2) BlueBERT outperforms the other two as it was trained on both PubMed abstract and clinical notes which makes its context more closer to radiology reports; (3) STL and GMTL appear to have similar performance and both outperform the MTL architecture. Although GMHL reports the poorest results, it is important to note that we only experimented with a fixed learning rate, number of epochs and the overall loss function. The number of clusters and the features used (either label name or label description) for clustering the tasks are also indicated in the clustering column. To compare these architectures, we only use F1-macro and F1-micro results, while the subsetAcc (Exact match ratio) is only reported for completeness, as it does not consider partial correctness of labels. The complete table of GMHL and GMTL experimental results is in Appendix C.

We observed that BlueBERT usually outperforms the BERT and BioBERT model since it is trained on both PubMed abstracts and MIMIC-III clinical notes, which is closer to the context of Open-I reports. Among the architectures that binary classification tasks are being learned jointly, GMTL performed the best while having similar results compared to learning tasks individually (STL). Although GMHL had

poor performance compared to other architectures, it is important to note that the architecture includes different hyperparameters to tune, and we only used a fixed set of hyperparameters to experiment with. The GMHL results can be considered a fair baseline for future work.

The standard way to solve a multi-label classification problem is to learn all the labels in a single model (MTL Architecture). Table 4.3 shows that GMTL is outperforming the MTL setting when using any of the BERT, BioBERT and BlueBERT models. We used the Wilcoxon signed-rank test (a non-parametric version of the paired T-test) for statistical significance tests of all three language models in GMTL and MTL settings. The p-value of GMTL and MTL significant test is calculated and listed in Table 4.4. A p-value less than 0.05 shows a statistical significance which is the case for most of the models. The BlueBERT language model has the most related context to the Open-I dataset, which its p-value successfully shows that GMTL-BlueBERT is statistically significant compared to MTL-BlueBERT for both F1-micro and F1-macro metric. It is the same case for BERT, although it has been trained on general domain text. Lastly, we found out BioBERT for GMTL is not statistically significant from MTL, which is because although BioBERT was trained on PubMed abstract articles, it has a different domain compared to radiology reports.

Model A	Model B	F1-macro p-value	F1-micro p-value
GMTL-BERT	MTL-BERT	0.002	0.004
GMTL-BioBERT	MTL-BioBERT	0.02	0.77
GMTL-BlueBERT	MTL-BlueBERT	0.002	0.004

Table 4.4: P-values from the Wilcoxon signed-rank test (a non-parametric version of the paired T-test) for the Open-I dataset. To check whether GMTL and MTL's observed results are significantly different, we performed statistical tests and reported their p-value in this table. The statistical significance test for BlueBERT, which is the model that has the most related context to the Open-I dataset p-value, shows that GMTL is statistically significant compare to MTL (as the p-value is less than 0.05, the null hypothesis gets rejected). Although BERT and BioBERT do not have a similar context to radiology reports, we still observe that GMTL is statistically significant when using BERT, but it fails to reject the null hypothesis for the statistical significance test for BioBERT.

The p-value of F1-macro is always lower than the F1-micro, and the reason is that the Open-I dataset is unbalanced. Although F1-micro helps to compare models,

F1-macro takes each class into account, making it a fairer comparison. We list all the 19 labels of the Open-I dataset and compare their F1-score per label to see how the GMTL and MTL models perform per label(binary classification task) in table 4.5.

class(Pathology)	F1-score(GMTL)	F1-score(MTL)	count(%)
Airspace Disease	$\mathbf{95.8 \pm 1.14}$	94.4 ± 2.07	25(3.2)
Atelectasis	$\mathbf{98.1 \pm 1.60}$	97.7 ± 0.48	66(8.3)
Calcified granuloma	96.4 ± 1.35	96.4 ± 1.35	55(6.9)
Calcinosis	96.5 ± 3.54	$\mathbf{98.0 \pm 0.94}$	61(7.7)
Cardiomegaly	$\mathbf{96.5 \pm 0.85}$	95.3 ± 0.82	75(9.5)
Cicatrix	$\mathbf{93.4 \pm 2.01}$	93.0 ± 2.58	42(5.3)
Deformity	$\mathbf{78.6 \pm 3.50}$	72.5 ± 5.25	23(2.9)
Effusion	$\mathbf{93.4 \pm 2.41}$	90.7 ± 2.58	33(4.2)
Emphysema	$\mathbf{91.8 \pm 1.55}$	89.7 ± 2.91	21(2.6)
Medical Device	$\mathbf{85.9 \pm 1.37}$	83.6 ± 2.91	26(3.3)
Nodule	$\mathbf{93.9 \pm 3.51}$	88.6 ± 4.33	22(2.8)
granulomatous disease	$\mathbf{96.7 \pm 1.83}$	94.9 ± 2.28	22(2.8)
indwelling catheters	$\mathbf{89.0 \pm 2.36}$	86.8 ± 3.97	25(3.2)
lung hyperdistention	$\mathbf{97.3 \pm 1.42}$	95.0 ± 0.67	41(5.2)
lung hypoinflation	$\mathbf{97.8 \pm 0.79}$	96.8 ± 0.79	56(7.1)
opacity	$\mathbf{97.9 \pm 0.32}$	97.4 ± 0.70	91(11.5)
spine degenerative	98.5 ± 2.12	98.5 ± 1.58	35(4.4)
surgical instrument	$\mathbf{92.3 \pm 1.16}$	90.0 ± 6.48	21(2.6)
thoracic vertebrae degenerative	$\mathbf{97.8 \pm 1.23}$	97.7 ± 1.06	53(6.7)
all	$\mathbf{93.86 \pm 0.34}$	90.64 ± 1.32	793(100)

Table 4.5: To further compare how Blue-BERT performs when comparing MTL and GMTL architecture for the Open-I dataset, we listed per label F1-score for each architecture and showed the count and percentage of each label (class). The count shows how many documents had the given class. It is observed that for most of the classes, GMTL is performing better compared to MTL. The higher scores are shown in bold.

Although GMTL successfully outperformed the MTL architecture, its performance is still close to STL. We recorded the clock time of each experiment (Table 4.9) and observed that GMTL is about three times faster than STL when trained on the Open-I dataset.

4.5 OHSUMED Results

Similar to Open-I, the same set of experiments was performed for the OHSUMED dataset. As the original OHSUMED dataset was computationally expensive for most of models we were training, we only used a 20% sample of the dataset. We experimented with the four architectures that MTLV supports and three BERT-family models (BERT, BioBERT, BlueBERT) and reported its results in Table 4.6.

Architecture	Model	clustering	F1-macro	F1-micro	subsetAcc
STL	BERT	-	59.56 ± 0.64	64.21 ± 1.06	32.86 ± 1.01
	BioBERT	-	70.36 ± 1.27	71.79 ± 0.47	41.22 ± 1.38
	BlueBERT	-	65.61 ± 0.62	69.20 ± 0.47	38.88 ± 1.57
MTL	BERT	-	59.35 ± 1.57	64.82 ± 1.21	37.67 ± 0.24
	BioBERT	-	69.10 ± 0.72	72.41 ± 0.71	44.57 ± 2.16
	BlueBERT	-	62.70 ± 1.34	66.46 ± 0.81	39.81 ± 0.27
GMHL	BERT	4 (Label)	46.83 ± 1.98	60.24 ± 2.84	52.09 ± 2.94
	BioBERT	3 (Label)	67.81 ± 0.80	72.33 ± 0.81	66.43 ± 0.73
	BlueBERT	5 (Label)	54.43 ± 2.17	65.26 ± 0.70	58.05 ± 1.22
GMTL	BERT	5 (LabelDesc)	63.13 ± 0.64	66.30 ± 0.35	57.28 ± 0.88
	BioBERT	5 (LabelDesc)	70.89 ± 0.63	72.43 ± 0.33	64.80 ± 0.35
	BlueBERT	5 (Label)	65.92 ± 0.44	68.63 ± 0.66	59.76 ± 1.02

Table 4.6: The results obtained by only training on 20% of the original OHSUMED dataset. BioBERT outperforms the other two as it was trained on 1 million PubMed abstracts, which makes it close to the context of the OHSUMED dataset. GMTL outperforms MTL, while STL and GMTL performance is similar (except for using BERT as shared layers). GMHL achieves the lowest results among the other ones. The full table of GMHL and GMTL experiments is in Appendix D)

As shown in Table 4.6, GMTL and STL performance achieve similar results and outperform the other two architecture (MTL and GMHL) results. The table shows that simply jointly training all the tasks brings negative transfer when the volume of the dataset (number of labels) grows. Even so, the GMHL results reflected in the table are disappointing. It is important to note that this architecture is only trained on a fixed set of hyperparameters and not fully explored. Therefore we leave further improving this architecture for future work. BioBERT language model surpasses the other models since it was trained on the most related context as the OHSUMED dataset, which consists of PubMed abstracts. BlueBERT also receives better performance compared to BERT, which is only trained on the general domain.

We performed a statistical significance test (Wilcoxon signed-rank test) for the three language models (BERT, BioBERT, BlueBERT) on the OHSUMED dataset and reported their F1-macro and F1-micro p-values in Table 4.7. A p-value less than 0.05 shows statistical significance, which is the case for all models when tested with their F1-macro. Although considering F1-macro and F1-micro together gives good insights about the model's overall performance, for an imbalanced dataset, model performance is primarily reflected in F1-macro.

Model A	Model B	F1-macro p-value	F1-micro p-value
GMTL-BERT	MTL-BERT	0.002	0.002
GMTL-BioBERT	MTL-BioBERT	0.002	0.375
GMTL-BlueBERT	MTL-BlueBERT	0.027	0.193

Table 4.7: P-values of statistical significance tests to compare GMTL and MTL models when trained on the OHSUMED dataset. We report the p-value of the Wilcoxon signed-rank test for both F1-macro and F1-micro. If a p-value is less than 0.05, the comparison of GMTL and MTL is considered statistically significant. The GMTL-BioBERT is statistically significant compared to MTL-BioBERT when they are tested with F1-macro. Although we could not reject the null hypothesis for BioBERT when tested with F1-micro, we argue that F1-macro can be trusted more when a dataset is imbalanced. The F1-score of each class of both GMTL and MTL models when BioBERT is their shared part is listed in Table 4.8. It is also observed that BERT and BlueBERT models are statistically significant when tested with F1-macro.

BioBERT was trained on 1 million PubMed abstracts, which makes its context the closest one to OHSUMED. The F1-macro for BioBERT is significantly significant, while F1-micro is not. To further investigate why F1-macro can play a more important role when comparing GMTL and MTL results, we listed the per-class F1-score of 10 fold cross-validation for the OHSUMED dataset in Table 4.8. Although the performance of each class in GMTL and MTL architectures are usually similar, some of the classes with fewer data samples have significantly improved when using GMTL, such as C02 by 11.9%, C22 by 36.6%, C03 by 4.1%. The overall F1-macro of GMTL surpasses MTL by 1.76%. The last row of the table shows the unique number of documents in the OHSUMED dataset.

GMTL and STL achieved comparable performance when trained on the BERT-family models, except for the BERT model that GMTL-BERT outperformed the STL-BERT. We argue that GMTL is a better design and approach compared to STL

that learns each task individually. As detailed in Table 4.9, we recorded the clock time for STL and GMTL when trained on BERT, BioBERT and BlueBERT to compare their computational cost.

4.6 Limitations

Although the MTLV library allows the user to set any set of hyperparameters for her network, such as the number of epochs, learning rate, optimizer, we used the same set of fixed values during our experiments for each dataset.

The GMTL and GMHL results rely heavily on the task clustering algorithm that decides which tasks are getting learned jointly. The label name and its description embeddings were used as a task feature, while no features of the tasks' documents were explored in this work. Also, the label description was found by searching for a definition online while it would be more proper for an expert to deliver those explanations for the model to learn embeddings and later for task clustering.

Only a single clustering algorithm(k-medoid) was used for all the experiments of GMHL and GMTL.

The GMHL architecture had the most hyperparameters (number of layers per head, overall loss, head initialization) and was only experimented with a set of fixed hyperparameters that led to the poorest performance overall other architectures.

class	F1-score(GMTL)	F1-score(MTL)	count(%)
C01	79.5 ± 1.35	79.5 ± 2.42	102(4.5)
C02	$\mathbf{62.6 \pm 5.4}$	53.7 ± 4.0	47(2.1)
C03	$\mathbf{92.0 \pm 2.94}$	87.9 ± 5.11	17(0.7)
C04	82.4 ± 1.51	$\mathbf{84.9 \pm 1.1}$	253(11.1)
C05	64.1 ± 2.92	$\mathbf{65.0 \pm 2.75}$	67(2.9)
C06	78.2 ± 1.14	$\mathbf{78.9 \pm 1.66}$	120(5.3)
C07	$\mathbf{75.8 \pm 3.79}$	70.3 ± 4.03	21(0.9)
C08	$\mathbf{73.3 \pm 3.02}$	72.7 ± 2.11	104(4.6)
C09	72.4 ± 2.67	$\mathbf{73.7 \pm 3.53}$	29(1.3)
C10	68.2 ± 2.04	68.2 ± 1.32	154(6.7)
C11	74.4 ± 1.65	$\mathbf{77.2 \pm 1.75}$	40(1.8)
C12	80.7 ± 1.06	$\mathbf{82.4 \pm 1.65}$	101(4.4)
C13	78.4 ± 2.22	$\mathbf{79.4 \pm 2.32}$	65(2.8)
C14	82.7 ± 1.06	$\mathbf{84.0 \pm 1.33}$	244(10.7)
C15	63.6 ± 1.71	$\mathbf{65.1 \pm 3.31}$	51(2.2)
C16	58.0 ± 2.75	$\mathbf{59.7 \pm 2.58}$	43(1.9)
C17	68.7 ± 1.83	$\mathbf{70.3 \pm 2.95}$	65(2.8)
C18	77.0 ± 2.31	$\mathbf{78.70 \pm 3.2}$	77(3.4)
C19	58.3 ± 3.56	$\mathbf{63.4 \pm 3.57}$	35(1.5)
C20	75.5 ± 2.01	$\mathbf{75.8 \pm 1.55}$	125(5.5)
C21	75.1 ± 2.02	$\mathbf{76.4 \pm 1.96}$	117(5.1)
C22	$\mathbf{37.4 \pm 6.47}$	0.8 ± 2.53	23(1.0)
C23	$\mathbf{55.6 \pm 4.5}$	46.1 ± 6.24	384(16.8)
all	$\mathbf{71.08 \pm 0.56}$	69.32 ± 0.95	2284(100)

Table 4.8: The F1-score (of each class) of GMTL-BioBERT and MTL-BioBERT when trained on OHSUMED is listed in this table. The last column (count%) is simply showing how many times a given class had documents corresponding to it. Although most of the classes are outperformed in the MTL setting, the classes with fewer documents in GMTL considerably outperformed MTL, such as C22, C03, C02 and C07, which shows how GMTL is affecting the performance of the OHSUMED dataset. The GMTL architecture of this table is using label descriptions as features for clustering and clustering the labels into five groups. The higher scores are shown in bold.

(a) Open-I dataset

Architecture	Clock time
STL-BERT	3.5 h
STL-BioBERT	3.5 h
STL-BlueBERT	3.6 h
GMTL-BERT	56.6 min
GMTL-BioBERT	45.0 min
GMTL-BlueBERT	1.1 h

(b) OHSUMED dataset

Architecture	Clock time
STL-BERT	15.6 h
STL-BioBERT	15.0 h
STL-BlueBERT	15.8 h
GMTL-BERT	3.5 h
GMTL-BioBERT	3.1 h
GMTL-BlueBERT	3.3 h

Table 4.9: The clock time for running Open-I and OHSUMED dataset experiments is listed in the above tables. The clock time for running Grouped multi-task learning (GMTL) architecture is about three times less for the Open-I dataset and almost five times less for the OHSUMED dataset than single task Learning (STL) ones. Both Open-I and OHSUMED results (for GMTL) are based on clustering to five clusters (group of tasks) except GMTL-BioBERT trained on Open-I dataset, which is trained with four clusters, so it has a lower clock-time compared to the other Open-I results.

Chapter 5

Conclusion

In this work, we introduced a framework called MTLV (Multi-Task Learning Visualizer) which supports four main architectures of Single-Task Learning (STL), Multi-task Learning (MTL), Grouped Multi-Head Learning (GMHL), and Grouped Multi-Task Learning (GMTL). This framework evaluated the effectiveness of using multi-task learning architectures compared to single-task learning. In the STL architecture, each task is learned individually using the same number of models as tasks, while in MTL, all the tasks are learned jointly. We introduced GMHL and GMTL architectures where tasks are clustered using label name and descriptions features and later assigned to different heads of a model or few models, respectively. We reported clustering results with the K-medoid algorithm with only a few clusters based on the elbow method.

Although learning all tasks together (MTL) has become a go-to approach to solve a multi-label classification problem(a set of binary classifications that share the same dataset), our experiments showed that STL and GMTL both outperform MTL. Our statistical tests confirmed that GMTL is statistically significant compared to MTL when testing their F1-macro scores. Although STL and GMTL produce similar results, we detected that STL is 3 to 5 times more computationally expensive when comparing their clock-time. Finally, GMHL received the poorest results among all the settings.

We used BERT-family models for training MTLV models and found out the corpus context that a model is trained on strongly affects the results of our approaches. Our experiments confirmed that the performance on biomedical and clinical datasets improved with BioBERT and BlueBERT as the shared layer of the multi-task learning design. Significantly, the BlueBERT-MTL achieved improvements over BERT-MTL by 4.92% and 2.98% for F1-macro and F1-micro, respectively, when trained on the Open-I dataset. BioBERT-MTL also surpasses BERT-MTL by 9.75% F1-macro and

by 7.59% F1-micro for the OHSUMED results.

Furthermore, we logged the parameters and metrics while the models were training and visualize them using MLflow to evaluate their learning process.

5.1 Future Work

There are several directions that we can follow to explore the proposed approaches of this work further. One such is the Grouped Multi-Head Learning (GMHL) architecture, which can be further enhanced by exploring other heads' initialization, optimizing algorithms, or lowering its learning rates. The overall loss of GMHL can include weighting algorithms to learn much each head should contribute to the overall loss.

Another direction is to consider other clustering algorithms such as K-means or Mean Shift to cluster the tasks based on the label name or label description embedding. The label (task) clustering approach can also go beyond just using the mean of the token embeddings of label names and use the count of labels as features to cluster the tasks. This approach might be more useful in GMHL rather than GMTL, where the label with a lower count gets assigned to the same head so that during training, we force the model to learn them more by giving them a higher weight compared to the other heads.

Another path to further explore could be the interpretability of GMHL, GMTL compared to MTL network to investigate how heads affect the learning process and the quality of the predictions. The visualization component of MTLV is taking advantage of the existing infrastructure which is the MLFlow tracking system. A more advanced system to both interpret and compare the runs is a suitable direction to follow.

Curriculum learning in combination with the proposed architecture is another candidate for the future work of this research. Using curriculum learning, which is a type of learning in which the model first learns easy samples of a task and then gradually increases the task difficulty, is a proper direction to follow in combination with the GMHL architecture.

Bibliography

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6430–6439, 2019.

[2] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc V Le. Bam! born-again multi-task networks for natural language understanding. *arXiv preprint arXiv:1907.04829*, 2019.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[4] Dina Demner-Fushman, Marc D Kohli, Marc B Rosenman, Sonya E Shooshan, Laritza Rodriguez, Sameer Antani, George R Thoma, and Clement J McDonald. Preparing a collection of radiology examinations for distribution and retrieval. *Journal of the American Medical Informatics Association*, 23(2):304–310, 2016.

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

[6] William Hersh, Chris Buckley, TJ Leone, and David Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer, 1994.

[7] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*. Springer Berlin Heidelberg, 1998.

[8] Leonard Kaufmann and Peter Rousseeuw. Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416, 01 1987.

[9] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[10] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017.

[11] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *ArXiv*, abs/1901.11504, 2019.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[13] Yifan Peng, Qingyu Chen, and Zhiyong Lu. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*, 2020.

[14] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)*, pages 58–65, 2019.

[15] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[16] Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Y. Ng, and Matthew P. Lungren. CheXbert: Combining Automatic Labelers and Expert Annotations for Accurate Radiology Report Labeling Using BERT. *arXiv e-prints*, page arXiv:2004.09167, April 2020.

[17] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[18] Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR, 2019.

[19] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*, 2018.

[20] Piotr Szymański and Tomasz Kajdanowicz. A scikit-based python environment for performing multi-label classification. *arXiv preprint arXiv:1702.01460*, 2017.

[21] D. Xu, Y. Shi, I. W. Tsang, Y. Ong, C. Gong, and X. Shen. Survey on multi-output learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7):2409–2429, 2020.

[22] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114v2*, 2017.

Appendix A

Datasets Details

The details of the Open-I and OHSUMED dataset that we choose for this research are discussed here. The distribution of each class for Open I and OHSUMED are depicted in Table A.1 and Table A.2 respectively.

Although The preprocessing steps for Open-I dataset is mentioned in this work, note that the "major" tag of the XML files are the Manual annotations while the "automatic" tag is the MTL annotations [4].

The OHSUMED dataset is a subset of the MEDLINE database maintained by the National Library of Medicine. MEDLINE database is a bibliographic database of peer-reviewed medical literature that contains 23 categories. These 23 categories counts, which are the classes of this multi-label dataset are listed in the table A.2.

The OHSUMED table classes are Medical Subject Headings (MeSH) categories of cardiovascular diseases group which are listed in the table A.3.

Pathology	total(%)	train(%)	test(%)
Airspace Disease	125(4.0)	100(4.0)	25(4.0)
Atelectasis	332(10.5)	266(10.5)	66(10.4)
Calcified granuloma	274(8.7)	219(8.7)	55(8.7)
Calcinosis	304(9.6)	243(9.6)	61(9.7)
Cardiomegaly	375(11.9)	300(11.9)	75(11.9)
Cicatrix	196(6.2)	154(6.1)	42(6.6)
Deformity	117(3.7)	94(3.7)	23(3.6)
Effusion	164(5.2)	131(5.2)	33(5.2)
Emphysema	105(3.3)	84(3.3)	21(3.3)
Medical Device	132(4.2)	106(4.2)	26(4.1)
Nodule	111(3.5)	89(3.5)	22(3.5)
granulomatous disease	111(3.5)	89(3.5)	22(3.5)
indwelling catheters	125(4.0)	100(4.0)	25(4.0)
lung hyperdistention	206(6.5)	165(6.5)	41(6.5)
lung hypoinflation	282(8.9)	226(8.9)	56(8.9)
opacity	455(14.4)	364(14.4)	91(14.4)
spine degenerative	175(5.5)	140(5.5)	35(5.5)
surgical instrument	105(3.3)	84(3.3)	21(3.3)
thoracic vertebrae deg.	265(8.4)	212(8.4)	53(8.4)
all	3159(100)	2527(100)	632(100)

Table A.1: The Open-I dataset is a multi-label classification problem that assigns a set of pathologies to radiology reports. The listed pathologies are the classes of this dataset. Each class's count and percentage are indicated for the train and test set and also their total. The last row of the label that shows the count for all the labels is showing the unique number of reports.

category	total(%)	train(%)	test(%)
C01	508(7.4)	406(7.3)	102(7.5)
C02	234(3.4)	187(3.4)	47(3.5)
C03	85(1.2)	68(1.2)	17(1.3)
C04	1265(18.4)	1012(18.3)	253(18.7)
C05	336(4.9)	269(4.9)	67(5.0)
C06	598(8.7)	478(8.6)	120(8.9)
C07	105(1.5)	84(1.5)	21(1.6)
C08	518(7.5)	414(7.5)	104(7.7)
C09	143(2.1)	114(2.1)	29(2.1)
C10	770(11.2)	616(11.1)	154(11.4)
C11	200(2.9)	160(2.9)	40(3.0)
C12	504(7.3)	403(7.3)	101(7.5)
C13	325(4.7)	260(4.7)	65(4.8)
C14	1220(17.7)	976(17.6)	244(18.0)
C15	255(3.7)	204(3.7)	51(3.8)
C16	217(3.1)	174(3.1)	43(3.2)
C17	323(4.7)	258(4.7)	65(4.8)
C18	384(5.6)	307(5.5)	77(5.7)
C19	173(2.5)	138(2.5)	35(2.6)
C20	623(9.0)	498(9.0)	125(9.2)
C21	587(8.5)	470(8.5)	117(8.6)
C22	101(1.5)	78(1.4)	23(1.7)
C23	1922(27.9)	1538(27.8)	384(28.4)
all	6891(100)	5538(100)	1353(100)

Table A.2: The count per class(category) for the OHSUMED dataset is reflected in this table. The category count and percentage in parenthesize show how the data is distributed over all the categories. The last row of the table is showing the unique number of documents in the dataset.

Categories	class
Bacterial Infections and Mycoses	C01
Virus Diseases	C02
Parasitic Diseases	C03
Neoplasms	C04
Musculoskeletal Diseases	C05
Digestive System Diseases	C06
Stomatognathic Diseases	C07
Respiratory Tract Diseases	C08
Otorhinolaryngologic Diseases	C09
Nervous System Diseases	C10
Eye Diseases	C11
Urologic and Male Genital Diseases	C12
Female Genital Diseases and Pregnancy Complicat...	C13
Cardiovascular Diseases	C14
Hemic and Lymphatic Diseases	C15
Neonatal Diseases and Abnormalities	C16
Skin and Connective Tissue Diseases	C17
Nutritional and Metabolic Diseases	C18
Endocrine Diseases	C19
Immunologic Diseases	C20
Disorders of Environmental Origin	C21
Animal Diseases	C22
Pathological Conditions, Signs and Symptoms	C23

Table A.3: The categories or classes of OHSUMED dataset and what they stand for are reflected in this table.

Appendix B

Library Structure

In this work, we developed a python package (library) that let user explore multi-task learning architectures. The structure of the library associated with this thesis is depicted in Fig B.1. The src directory contains all the source code of the library where each subpackage focuses on a separate part of building and training the model. For instance, the heads sub-package focus on how to cluster and build the grouping of heads for Grouped Multi-task learning and Grouped Multi-head Learning architectures, while Sample configuration files of the hyperparameters which are given as inputs are provided in the config directory. This library is available on GitHub [1].

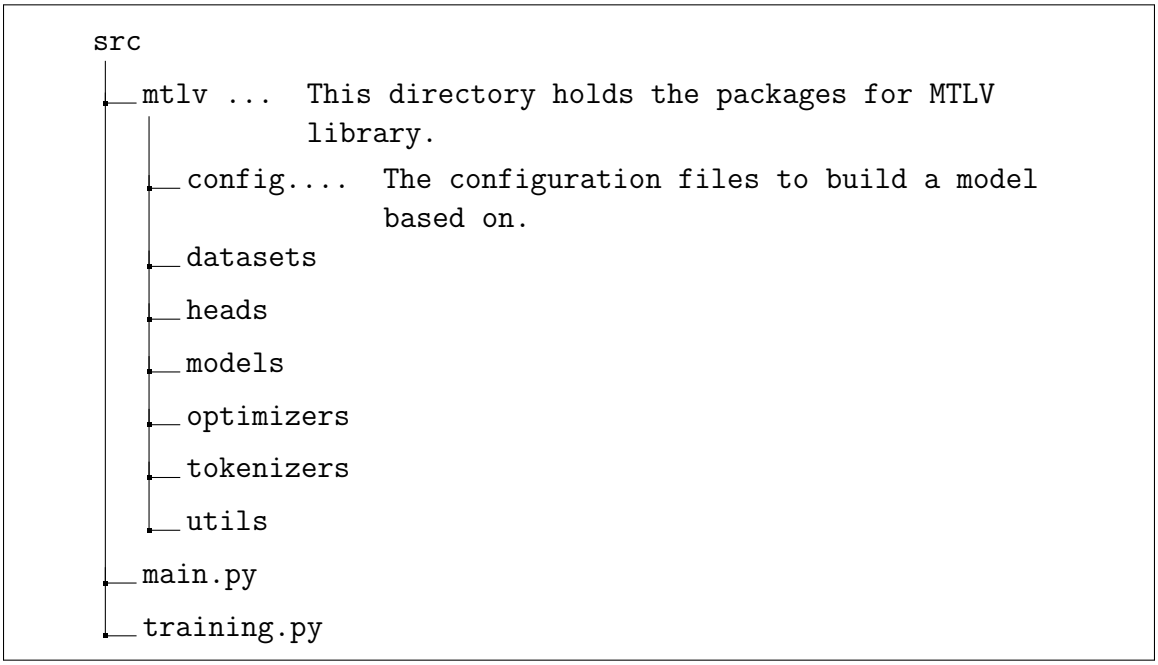

Figure B.1: The structure of the MTLV library is depicted in this figure. The mtl directory in the src includes all of the architecture and implementations of this thesis.

[1] https://github.com/fatemerhmi/MTLV

Appendix C

Open-I detailed results

We experimented with a set of hyperparameters for clustering the labels(tasks) of the OHSUMED dataset, such as Label/Label description and the number of clusters. The GMHL and GMTL detailed result of the Open-I dataset is listed in Table C.1, and Table C.2 respectively.

Model	# clusters	Clustering	F1-macro	F1-micro	subsetAcc
BERT	3	Label	46.69 ± 2.88	65.83 ± 3.65	68.06 ± 2.89
		Label Desc	59.34 ± 2.56	77.02 ± 2.52	78.90 ± 2.32
	4	Label	40.63 ± 2.03	55.66 ± 2.24	58.43 ± 1.67
		Label Desc	61.81 ± 5.66	80.48 ± 3.16	81.67 ± 2.63
	5	Label	50.57 ± 2.36	66.00 ± 2.08	66.94 ± 2.19
		Label Desc	63.73 ± 5.38	82.23 ± 2.38	82.95 ± 1.95
BioBERT	3	Label	74.96 ± 3.14	88.16 ± 0.92	88.25 ± 0.77
		Label Desc	73.49 ± 0.43	85.40 ± 0.41	86.00 ± 0.25
	4	Label	75.28 ± 5.12	87.04 ± 2.07	87.43 ± 1.80
		Label Desc	66.77 ± 4.30	82.51 ± 2.27	83.45 ± 2.06
	5	Label	65.30 ± 3.28	83.30 ± 1.09	84.00 ± 1.19
		Label Desc	79.86 ± 4.16	88.73 ± 1.16	88.97 ± 0.94
BlueBERT	3	Label	75.82 ± 4.96	87.78 ± 2.01	87.75 ± 1.81
		Label Desc	71.85 ± 2.99	86.88 ± 0.99	87.18 ± 0.76
	4	Label	69.74 ± 1.20	85.85 ± 0.37	85.39 ± 0.58
		Label Desc	74.02 ± 2.99	87.27 ± 1.44	87.54 ± 1.32
	5	Label	67.54 ± 0.96	83.19 ± 1.00	82.91 ± 1.25
		Label Desc	76.36 ± 1.66	89.11 ± 0.11	89.16 ± 0.08

Table C.1: Detailed results of GMHL architecture for Open-I dataset is listed in this table. We experimented with clustering label names and their description and a range of clusters (from 3 to 5). To distinguish the best number of clusters we plotted the elbow method and figured that this range is best suited for this dataset. The best performing hyperparameters for the GMHL setting are using 5 clusters with Label description embeddings as features, which are highlighted in the table.

Model	# clusters	Clustering	F1-macro	F1-micro	subsetAcc
BERT	3	Label	90.64 ± 0.99	92.93 ± 0.73	92.25 ± 0.88
		Label Desc	91.15 ± 0.51	93.35 ± 0.42	92.44 ± 0.53
	4	Label	91.06 ± 0.53	92.97 ± 0.33	92.11 ± 0.75
		Label Desc	91.39 ± 0.31	93.37 ± 0.40	92.52 ± 0.48
	5	Label	90.82 ± 0.39	92.82 ± 0.28	91.76 ± 0.33
		Label Desc	91.45 ± 0.24	93.22 ± 0.27	92.34 ± 0.36
BioBERT	3	Label	92.17 ± 0.68	94.15 ± 0.54	93.41 ± 0.49
		Label Desc	91.79 ± 0.93	93.81 ± 0.87	93.23 ± 0.66
	4	Label	92.20 ± 0.30	93.98 ± 0.32	93.18 ± 0.39
		Label Desc	92.56 ± 0.39	94.25 ± 0.30	93.76 ± 0.46
	5	Label	91.60 ± 0.56	93.40 ± 0.72	92.59 ± 0.94
		Label Desc	91.49 ± 0.83	93.43 ± 0.53	92.38 ± 0.73
BlueBERT	3	Label	93.01 ± 0.54	94.65 ± 0.38	93.86 ± 0.37
		Label Desc	93.20 ± 0.30	94.79 ± 0.34	94.37 ± 0.28
	4	Label	93.53 ± 0.43	95.07 ± 0.47	94.35 ± 0.50
		Label Desc	93.55 ± 0.62	95.03 ± 0.57	94.48 ± 0.55
	5	Label	93.38 ± 0.39	94.83 ± 0.31	94.19 ± 0.31
		Label Desc	93.86 ± 0.34	95.32 ± 0.15	94.40 ± 0.24

Table C.2: A comprehensive performance of GMTL architecture for Open-I dataset is listed in this table. The results show that using Label Description achieves the best performing results for all the 3 language models. The BlueBERT results outperform the other two as it has the closest context to the Open-I dataset. The best set of results per each model is highlighted in the table.

Appendix D

OHSUMED detailed results

Similar to Open-I we also performed the same set of experiments for the OHSUMED dataset. The two main hyperparameters of GMHL and GMTL are the clustering features and the number of clusters. We experimented with both the label names and label descriptions of the OHSUMED dataset along with a range of numbers for clusters (from 3 to 5) which were chosen from plotting the elbow method. The full results for GMHL and GMTL architectures are depicted in Table D.1 and D.2 respectively.

Model	# clusters	Clustering	F1-macro	F1-micro	subsetAcc
BERT	3	Label	43.76 ± 0.38	62.83 ± 0.84	55.09 ± 0.83
		Label Desc	44.77 ± 0.26	62.70 ± 0.81	55.17 ± 0.51
	4	Label	46.83 ± 1.98	60.24 ± 2.84	52.09 ± 2.94
		Label Desc	31.04 ± 0.49	51.99 ± 1.01	42.49 ± 0.47
	5	Label	38.40 ± 2.85	55.71 ± 1.12	46.33 ± 1.57
		Label Desc	40.84 ± 0.98	58.98 ± 0.87	33.85 ± 0.27
BioBERT	3	Label	67.81 ± 0.80	72.33 ± 0.81	66.43 ± 0.73
		Label Desc	51.96 ± 1.27	66.44 ± 0.31	59.62 ± 0.37
	4	Label	59.73 ± 1.87	69.38 ± 0.19	63.17 ± 0.39
		Label Desc	60.22 ± 2.24	70.03 ± 1.16	63.97 ± 1.32
	5	Label	68.10 ± 0.45	70.75 ± 0.40	64.95 ± 0.29
		Label Desc	54.84 ± 0.49	67.72 ± 0.54	61.07 ± 0.85
BlueBERT	3	Label	52.66 ± 0.85	63.81 ± 0.25	57.19 ± 0.60
		Label Desc	51.21 ± 0.62	62.71 ± 0.73	56.14 ± 1.14
	4	Label	47.39 ± 1.52	62.97 ± 1.10	55.70 ± 0.92
		Label Desc	49.67 ± 3.32	62.89 ± 1.10	56.40 ± 1.59
	5	Label	54.43 ± 2.17	65.26 ± 0.70	58.05 ± 1.22
		Label Desc	50.80 ± 1.68	62.64 ± 1.55	55.73 ± 1.79

Table D.1: The full result of the OHSUMED dataset in GMHL architecture is listed in this table. The number of clusters for best-performing hyperparameters varies per language mode since BERT, BioBERT and BlueBERT are achieve the best results when using 4, 3 and 5 clusters respectively. For all the models, using the label embeddings as features are outperforming label description embeddings. The best performing configuration per model is highlighted in the table.

Model	# clusters	Clustering	F1-macro	F1-micro	subsetAcc
BERT	3	Label	62.96 ± 0.47	66.95 ± 0.49	58.29 ± 0.72
		Label Desc	62.85 ± 0.31	66.54 ± 0.85	57.21 ± 1.46
	4	Label	63.14 ± 0.67	66.33 ± 0.74	56.82 ± 1.04
		Label Desc	62.66 ± 0.69	66.14 ± 0.61	57.16 ± 1.14
	5	Label	62.50 ± 0.28	66.69 ± 0.17	57.73 ± 0.12
		Label Desc	63.13 ± 0.64	66.30 ± 0.35	57.28 ± 0.88
BioBERT	3	Label	70.89 ± 0.63	72.43 ± 0.33	64.80 ± 0.35
		Label Desc	70.51 ± 0.67	71.79 ± 0.69	63.99 ± 0.77
	4	Label	70.13 ± 1.18	72.54 ± 0.80	64.84 ± 0.94
		Label Desc	70.46 ± 0.80	72.10 ± 0.57	63.84 ± 0.94
	5	Label	71.01 ± 0.69	72.97 ± 0.21	65.18 ± 0.44
		Label Desc	71.23 ± 0.75	72.52 ± 0.16	64.37 ± 0.39
BlueBERT	3	Label	65.19 ± 1.24	68.44 ± 0.70	59.89 ± 1.42
		Label Desc	64.94 ± 0.55	69.03 ± 0.34	61.02 ± 0.43
	4	Label	66.04 ± 1.26	69.10 ± 0.99	60.29 ± 1.21
		Label Desc	65.89 ± 0.71	68.57 ± 0.19	59.96 ± 0.31
	5	Label	65.92 ± 0.44	68.63 ± 0.66	59.76 ± 1.02
		Label Desc	65.31 ± 1.26	68.35 ± 0.27	60.18 ± 0.32

Table D.2: This table lists the GMTL detailed performance for the OHSUMED dataset. All the language models are performing best when their number of clusters is 5. The BioBERT surpasses the other models as it has a closer context to the OHSUMED dataset. Although BERT and BioBERT's best-performed configuration is using Label description, BluBERT uses Label embedding as the clustering features. Highlighted cells are the best performing settings of GMTL architecture per language model.

Appendix E

MLflow Tracking UI

An example of the MLflow Tracking UI is depicted in Fig E.2.

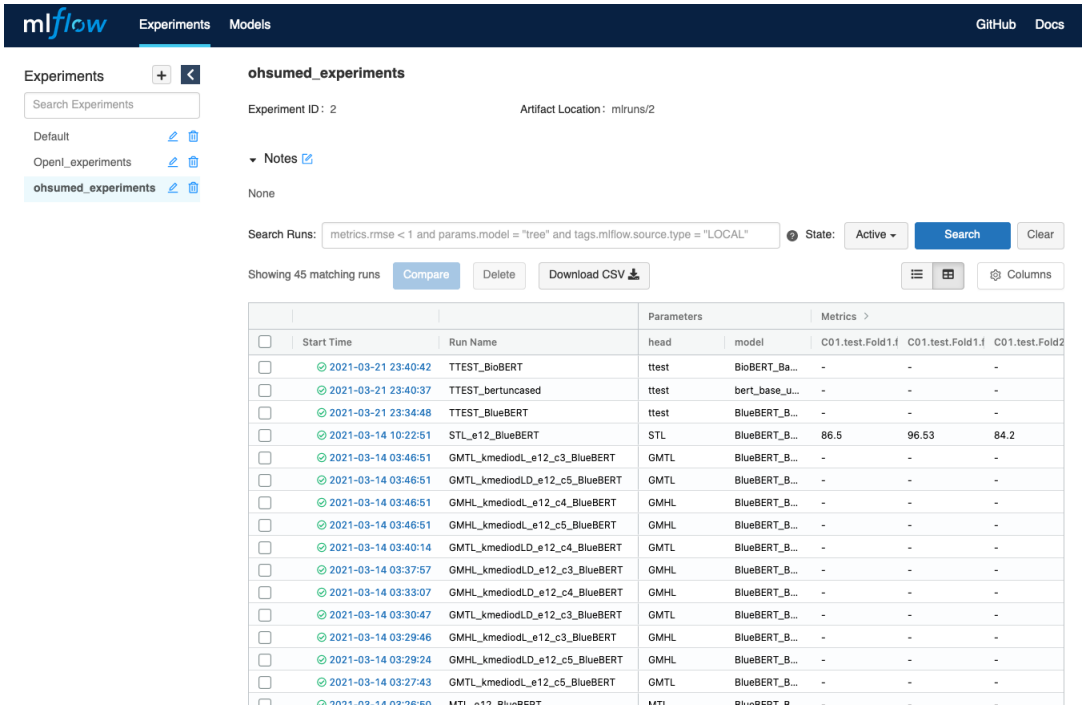

Figure E.1: In this figure the UI of the MTLV framework is depicted. After training different models with a different set of design decisions, it becomes necessary to keep track of their performance and how they are learning in different settings. We use MLflow Tracking to depict and compare these results and learning charts.

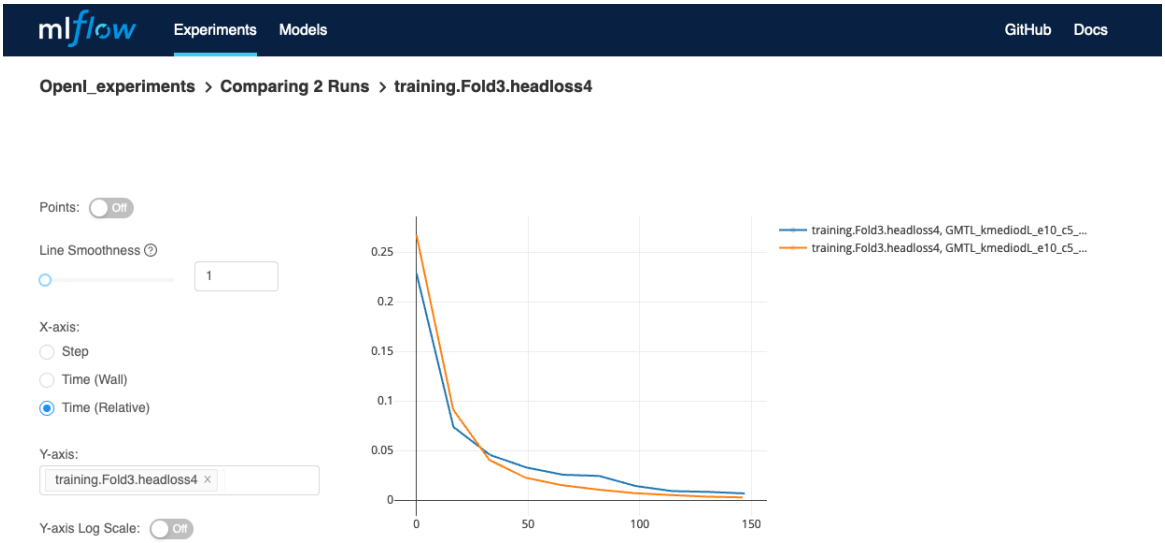

Figure E.2: In this figure, the comparison of learning from 2 GMTL architecture using BlueBERT and BioBERT as their shared layers is depicted. MLflow Tracking UI allows users to compare the learning of different heads of same run or architectures of different runs.