Reproducibility Project for CS598 DL4H in Spring 2023

Miles Grogger and Jeremy Ahn
{milesjg2, jcahn2}@illinois.edu

Group ID: 90
Paper ID: 104
Presentation link: https://www.youtube.com/watch?v=7uFIdRXgoyo
Code link: https://github.com/jcahn2/DLH_2023_keyclass

1 Introduction

Traditionally, EHR data, or Electronic Health Records, is an essential form of healthcare data containing nearly real-time patient records of medications, diagnoses, procedures, etc. However, the unstructured form of many clinical texts that come with EHR data is often a difficulty when it comes to creating classification models. The method of assigning codes for supervised learning to unstructured clinical notes from EHR data has proven costly, difficult, and overall inefficient, given that the job of assigning codes related to procedures or diagnostics for example inevitably fall to human coders.

KeyClass (Gao et al., 2022) is a solution to the problem of classifying unstructured clinical notes by being a weakly supervised framework for text classification. This model combines Data Programming (Ratner et al., 2016) with a novel voting method for obtaining weak supervision sources for later classification. In the context of the unstructured text from EHR data, these sources are keywords and phrases interpreted from class-label descriptions that allow the model to assign ICD-9 codes to patient notes. The novelty of KeyClass comes in the fact that this is done without labeled documents, giving its weakly supervised nature. Furthermore, the model contributes to the multi-class multilabel field of classification by being the first to use Data Programming as part of its process. Successfully applying KeyClass opens the door to efficient and affordable deployments of text classifiers without needing human coders to process thousands of unstructured text documents.

2 Scope of reproducibility

This paper (Gao et al., 2022) introduces the general weakly supervised text classification model **KeyClass**, which finds weak supervision sources from class descriptions that are used for downstream classifiers without labeled documents. KeyClass performs on par with another text classification model, **FasTag** (Venkataraman et al., 2020), on MIMIC-III ICD-9 code assignments, and outperforms other weakly supervised models such as Dataless, WeSTClass, and LOTClass on other text datasets.

The scope of reproducibility is determined largely by the complexity of the classification problem the KeyClass model is trying to solve. While we were able to manipulate the code provided on the KeyClass Github to allow for the original multi-label functionality the paper had, the authors of the paper did not have, nor did they remember, the code that made all components train with multi-label capabilities. This impacted our ability to replicate the exact process in full.

Because one of the main goals of KeyClass is to apply weak supervision as a solution to a common text classification problem, a large focus of our reproduction was placed on the model's ability to produce embedded data and probabilistic labels that can be used interchangably with downstream classifiers. To get this right, we needed to pre-process the MIMIC data in a way that maximized the amount of signal the different embeddings would be able to capture. However, the original paper and code did not provide this. We used the FasTag process, but were unsure if that was all the authors of KeyClass did.

Finally, the original authors did not provide information on the downstream classifier that they used to tag ICD-9 codes. Therefore, our architecture was built mostly based off of what they had for other applications, as well as a TCN architecture we found online for the ablations (Bai et al., 2018).

2.1 Addressed claims from the original paper

- KeyClass performs on par with FasTag on MIMIC-III ICD-9 code assignment.
- KeyClass performs better with domain-specific encoders on Medical data.
- KeyClass can effectively inform any downstream models for text classification.

3 Methodology

Our methodology for implementing KeyClass largely followed the steps outlined in the paper. At a high level description, our approach involved using class descriptions to find relevant keywords to encode the dataset for downstream classification.

3.1 Model descriptions

As mentioned before, the KeyClass methodology relies on forming keywords and phrases from class descriptions. In the case of ICD-9 code assignment, we are able to mine a list of the most frequent words in the text descriptions of the codes [1]. KeyClass then finds keywords and phrases to serve as sources of weak supervision by using the CountVectorizer function from scikit-learn (Buitinck et al., 2013) and applies these to their respective ICD-9 code categories using pre-trained language models such as BERT (Devlin et al., 2019). After, KeyClass can then generate the labeling functions needed to produce probabilistic labels that will help train the downstream classifier. Keywords from unlabeled text data can then be mined and passed through a downstream classifier for code assignment. The existing classifier model was a simple Feed Forward network with several connected layers. But, based on the paper's claims mentioned above, the structure and type of classifier are interchangeable.

3.2 Data descriptions

The main dataset used to train and test KeyClass was the MIMIC-III dataset (Johnson et al., 2023), which consists of deidentified health data of nearly 50,000 patients and around 52,000 visits. This data was recorded over the years of 2001 to 2012 at the Beth Israel Deaconess Medical Center, and includes data such as clinical event notes, diagnoses, procedure codes, etc. The information most relevant for this study were the diagnosis codes as well as the event notes. The database was accessed through the PhysioNet repository (Goldberger et al., 2000). For initial testing, the IMDb movie description dataset was used, as this was the data the KeyClass code was originally set up to process.

3.3 Hyperparameters

Most hyperparameters were set using the prespecified parameters in the confgiuration files provided by the KeyClass authors. They included a batch size of 128, learning rate of 1e-6 for the downstream classifier, a LeakyReLU activation function, and a dropout rate of 0.5 for the hidden layers. For the upstream labeling model, we set the learning rate to 1e-4.

3.4 Implementation

The majority of our implementation was taken from the freely-available KeyClass github, where a basic description of the method, tutorials, and datasets can be found. However, as mentioned before, since the available code was set up to process the IMDb dataset, we of course had to make small adjustments to components such as the configuration files that aided in parsing the data sources. The preprocessing of the MIMIC-III data itself was also inspired by FasTag [2] (Venkataraman et al., 2020), a related study that prepared the data in a similar way to the KeyClass method.

In addition, the original paper did not provide much clarity on the architecture of the downstream classifier used in the experiments. The code provided freely on Github[3] provided a default configuration of 4 fully connected layers with 768, 256, 64, and 17 nodes respectively. The final represented the output layer of 17 classes, which represents the high level categories detailed by the FasTag approach (Venkataraman et al., 2020). Each hidden layer was activated by a LeakyReLU, and featured a dropout rate of 0.5.

Our own implementation followed the general structure of the original Feed Forward model described above, but also included some changes. Instead of 4 layers, our classifier ended up using 10 fully connected layers (8 hidden layers) with the following node configuration: [768, 1024, 512, 256, 64, 256, 512, 1024, 64, 17]. Furthermore, our ablation concerning the downstream involved 2 blocks of the previously mentioned Temporal Convolution Network (TCN) that used filter sizes 128 and 64

[1] www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes

[2] https://github.com/rivas-lab/FasTag

[3] https://github.com/autonlab/KeyClass

respectively with kernel size of 3 and dropout value of 0.1. After a mix of max and average pooling layers, the result was passed through 2 fully connected layers with node sizes 128 and 64 respectively as well as a dropout layer of the same value.

Each model was trained using the nn.BCELoss function from Pytorch to support multilabel classification. However, though the classifiers were able to produce multilabel results reflected in probabilities for each class, only a single label was derived using an adaptive optimal threshold based on each class.

Here is our final code: Github

3.5 Computational requirements

For our experiments, we used a VM in the Compute Engine system of the Google Cloud Platform that uses an n1-highmem-8 CPU with 52 GB of memory and a NVIDIA T4 GPU with 16 GB of memory. This was configured on a machine with 150GB of Disk space. Our initial run through with the full MIMIC-III Notes dataset was completed in just undar an hour.

4 Results

4.1 Claim 1

We were not able to validate the original authors' claims that KeyClass performed as well as FasTag. In the original paper, Weakly Supervised FasTag with a BlueBERT embedding boasted a Recall over 70% and Precision near 45%. As Table 1 shows, this is far higher than what we were able to accomplish. In large part, we believe this is due to the lack of documentation of the downstream classifier architecture in the KeyClass paper.

Table 1: Model Performance

Score	KeyClass with Fully Connected	Original KeyClass	FasTag
Precision	0.346	0.507	0.436
Recall	0.244	0.896	0.734
F1	0.286	0.625	0.525

4.2 Claim 2

The original paper made a key assumption that MPNET would not perform as well as BlueBERT. The main rationale is that MPNET is trained on a large, but generic, corpus, whereas BlueBERT was trained on medical records specifically. However, what we observed is that MPNET with a Feed Forward Architecture outperformed BlueBERT. This suggests that the advantages of MPNET over BERT architecture could extend beyond the potential limitations of a more generalized training set. Results from this study are reported in Table 2.

Table 2: Encoder Performance

Score	BlueBERT	MPNET
Precision	0.346	0.453
Recall	0.244	0.471
F1	0.286	0.462

4.3 Claim 3

One of the more powerful claims of KeyClass is that the overall approach can be used with any downstream architecture. We tested this claim by conducting an ablation study where we replaced the Fully Connected architecture with a TCN architecture specified by Bai et al. (2018). As mentioned before, this architecture featured two TCN blocks, the merged results of two different global pooling layers, and a final fully connected layer with dropout prior to the output layer. With a more sophisticated architecture, we expected to see better results than the fully connected architecture, but did not observe this (as shown in Table 3). Likely this is due to overfitting.

Table 3: Architecture Performance

Score	BERT FC	MPNET FC	BERT TCN	MPNET TCN
Precision	0.346	0.453	0.115	0.074
Recall	0.244	0.471	0.024	0.034
F1	0.286	0.462	0.039	0.046

5 Discussion

There are several factors that have made the original paper difficult to reproduce. The first is that the configurations of the original model, including learning rates, class balance priors, neural network architecture, and class description definitions have all lacked sufficient descriptions for reproduction. While the configuration files provided in the author's Github detailed the hyperparameters used in those specific settings, they were for simpler classification problems with more readily available data. Additionally, none of the provided code for the models was able to support a multi-label set up, and instead required applications to be multi-class. As mentioned before, we were able to modify parts of the code to allow for multi-label training and prediction, but given the multiple moving parts, we could not guarantee that our approach to the multi-labeling approach was in line with what the original authors had done. Finally, the MIMIC data is fairly messy and can introduce a lot of noise in the embeddings. We followed the approach the FasTag authors laid out, since we assumed that this is what the KeyClass authors did too, but given our results did not line up with theirs in the end, it's natural to assume this could have used more clarity too.

Despite not being able to reproduce their results, we did come across some interesting findings. Most interesting was how MPNET embeddings outperformed BlueBERT embeddings. Our understanding from the original authors' work, and common tips for NLP applications, was that domain-relevant training can massively improve downstream model results when using embeddings. Seeing MPNET outperform BlueBERT suggests that its architecture is potentially robust to domain-specific performance, which makes it much more applicable.

5.1 What was easy

For a simple classification problem, the KeyClass Github provides a lot of the backbone required to replicate their original results on more readily available datasets. Fortunately, that means that for even more complicated problems, we do not have to recreate the wheel, and can rely on much of the provided architecture. This is especially useful for handling complex software line Snorkely.AI, which is used for the Labeling Models.

5.2 What was difficult

While KeyClass does provide a relatively straightforward set up process for simple classification problems, there was a lot of re-work required for more complex data like MIMIC-III. First and foremost was the multi-label issue discussed extensively above. However, without understanding the number of keywords originally used, extensive details of the pre-processing steps for class descriptions, or downstream model architecture and hyperparameters, it has been very difficult to recreate the multi-class version of the multi-label model that the authors originally had. We will continue our experiments moving forward, but this limitation may keep us from achieving the accuracy they alluded to in the KeyClass paper.

5.3 Recommendations for reproducibility

Providing more descriptions of the data processing, neural network architecture, and hyperparameters used in the original application would make this process a lot easier. Much of what we've had to go off of is educated guesses based on our understandings of neural networks and the configurations for the provided examples. We would recommend explicit code examples in Github, not just for generic datasets, but for ones specifically required to reproduce the paper's results. Typically, this is provided in academic papers, so it was surprising to not find this here.

6 Communication with original authors

We were lucky enough to be in contact with one of the original authors, Chufan Gao, who, when asked about MIMIC-III data preprocessing, suggested to follow the method used by FasTag (Venkataraman et al., 2020). He mentioned that this was essentially the steps they followed for their own experiments, which helped us, but did not give us the fullest picture for what they had done to get the results they espoused.

References

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake Vanderplas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. Api design for machine learning software: experiences from the scikit-learn project.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Chufan Gao, Mononito Goswami, Jieshi Chen, and Artur Dubrawski. 2022. Classifying unstructured clinical notes via automatic weak supervision. *Machine Learning for Healthcare Conference*.

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101(23).

Alistair Johnson, Tom Pollard, and Roger Mark. 2023. Mimic-iii clinical database.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly.

Guhan Ram Venkataraman, Arturo Lopez Pineda, Oliver J Bear Don't Walk IV, Ashley M Zehnder, Sandeep Ayyar, Rodney L Page, Carlos D Bustamante, and Manuel A Rivas. 2020. Fastag: Automatic text classification of unstructured medical narratives. *PLoS one, 15 (6):e0234647*.