Iowa Initiative for Artificial Intelligence
Final Report

Project title:	A Self-Supervised Machine Learning Framework for Automating Heart Failure Identification
Principal Investigator:	Samuel Zetumer, Kishlay Jha, and Hans Johnson
Prepared by (IIAI):	Avinash Mudireddy
Other investigators:	
Date:	11/1/2023

Were specific aims fulfilled:	No
Readiness for extramural proposal?	No
If yes … Planned submission date	
Funding agency	
Grant mechanism	
If no … Why not? What went wrong?	• IIAI team could not exactly match the original performance of MeshprobeNet competition during the given project duration. • Work on Caboodle could not begin due to unavailability of target labels. The idea is to train on MeshProbenet to predict Mesh terms for Caboodle data which can then be used to predict targets. Delay in step 1 delayed step2

Brief summary of accomplished results:

Research report:

Aims (provided by PI):

Heart failure, a condition in which the heart cannot maintain sufficient outflow of blood to the body's organs, is one of the most common cardiovascular diseases in the United States. The development of novel drugs and devices to manage heart failure is outpacing our ability to clinically evaluate their effectiveness during retrospective reviews. A critical bottleneck occurs because we can't study real-world populations affected by novel developments due to the excessive time needed for experts to review retrospective clinical information. Our proposal addresses this critical impediment by leveraging recent advances in machine learning as applied to medical free text to accelerate identifying appropriate case studies for analysis.

Our ultimate goal is to automate the collection of data from medical notes for heart failure research. For this pilot, we will demonstrate proof of concept implementations suitable for supporting a larger grant submission. The pilot project aims **to develop and train a machine learning algorithm that identifies hospitalized patients with heart failure exacerbation based on their medical notes.** This is a necessary task for conducting heart failure research.

Variables: This pilot machine learning algorithm will use the free text of patients' medical notes to make predictions (X). These data are found in the ClinicalNoteFact and ClinicalNoteTextFact tables of Caboodle. The outcome to be predicted is whether a patient was admitted to the hospital with heart failure exacerbation (Y), as determined by a laboriously written heuristic algorithm written by Dr. Zetumer followed by a manual review. The

heuristic algorithm uses medication orders, diagnostic codes, test results, vital signs, but not any information from medical notes. This hard-coded algorithm has been manually audited for accuracy and used in prior research.
Statement of AI/ML and Engineering Relevance: Electronic medical records present a rich source of data for clinical researchers but extracting actionable insights from them remains a challenge. When posed as a text classification problem, identifying these patients becomes amenable to machine learning approaches. However, the ML formulation of this task presents technical/engineering challenges such as named entity recognition, concept normalization, and relationship extraction. Solving these problems requires massive computational resources and the implementation of very recently developed architectures.
Strategies to advance AI/ML technology/theory: To address these challenges, we will apply a recently developed computational platform that exploits self-supervised machine learning to learn high-quality feature representations of clinical notes. Using these meaningful feature representations as input, the classifier will be trained to identify patients admitted to the hospital with heart failure exacerbation. We believe this will successfully complete the task because the innovative integration of self-supervision and attention with a bidirectional recurrent neural network has provided benchmark results in many non-medical domains. If it succeeds here as it has in other domains, it will outperform other architectures in the specific task of identifying patients with heart failure exacerbation.
Specifics on how improvements of the AI/ML technology/theory will advance the research: Most of the existing approaches of analyzing clinical text adopt a supervised machine learning strategy relying on the availability of human-annotated training data. This presents a serious obstacle for clinical domains in which annotated medical notes are costly to obtain. To address this issue, we propose a novel machine learning approach that relies on the input data (i.e., massive clinical notes) in addition to the output variable label to 'supervise' training the neural network.
The architecture of this model has three components. The first component is a bidirectional Recurrent Neural Network (RNN) on the text of the EMRs. The bidirectional RNN generates the feature representations for words, i.e., the hidden states, which capture the semantics of the clinical narrative. The second component is a set of attention heads, which are responsible for extracting useful information from the RNN hidden states and converting variable length EMR texts into fixed-dimension feature matrices. The attention heads are a set of vectors that assign weights to the RNN hidden states based on their relevance. These attention heads attenuate the unimportant hidden states and amplify the contribution of more salient states. The third component, self-supervision, increases the 'amount of training' the model can do. If the model only had a binary output to train with ('heart failure exacerbation, yes/no'), it would not correctly learn meaningful hidden states. However, by using the notes themselves as both input and outputs during training, we increase the richness of the learned features. This is done by masking parts of the note and asking the model to predict the masked component. In this manner, unlabeled data (the clinic note) is transformed into labeled data, with the masked part serving as the label. This additional resource has been essential to recent advances in natural language processing but has not yet been applied to medical notes.
Data Characteristics and Accessibility: We will be using data from Caboodle, a relational database containing all patient data recorded in UIHC's Epic Systems, for this pilot study. We will use tables containing data from blood tests, vitals, diagnoses, test results, and the free text of medical notes. The ClinicalNoteFact table contains 64 million medical notes. It is expected to be sufficient for training deep neural networks based on previous success with similarly structured data for other projects. **The investigators currently have access to Caboodle and analyze clinical notes regularly. Dr. Zetumer has an abstract accepted to a peer-reviewed conference based on a preliminary analysis of these clinical notes.** The abstract has been submitted.

AIM defined for IIAI team:

Prior to delving into the Caboodle data, the team aims to assess the performance of the proposed algorithm by conducting tests on publicly available PUBMED data. In a publication authored by Professor Kishlay Jha's team titled "**MeSHProbeNet: a self-attentive probe net for MeSH indexing"** they introduced an end-to-end framework named MeSHProbeNet. This framework leverages deep learning techniques and self-attentive MeSH probes to automate the indexing of MeSH terms for biomedical articles. MeSHProbeNet facilitates the extraction of precise biomedical knowledge from input articles, offering comprehensive information and interpretability down to the word level. The system recommends MeSH terms using a unified classifier, making it both time-efficient and space-efficient, thereby addressing the limitations of existing automatic MeSH indexing systems. MeSHProbeNet achieved first place in the most recent iteration of Task A in the 2018 BioASQ challenge, and the results on the final test set of the competition are presented in the same paper.

However, it's worth noting that the paper was published in 2019, and the previously available code from the team remains incomplete, while the data used in the competition is also unavailable. Consequently, the IIAI team is tasked with the construction of MeSHProbeNet and its training on the 2022 BIOASQ dataset.

Throughout the project, the team also explored the implementation of a BERT-based model and another dataset featured in the paper "https://arxiv.org/pdf/2102.07349.pdf."

Data for Aims defined for IIAI team:

The dataset used in this study consists of annotated articles from PubMed, where MeSH (Medical Subject Headings) terms have been assigned to the articles by human curators. The dataset is available in various versions, each corresponding to a specific year, with variations primarily in the MeSH terms used. In the context of the study conducted in 2022, we focus on the "Training v.2022" dataset, which provides the following details:

- Dataset version: Training v.2022 (available in both text and Lucene formats)
- Number of articles: 16,218,838
- Average MeSH terms per article: 12.68
- MeSH terms covered: 29,681 unique MeSH terms
- Size in zip/unzip (text format): 8.9 gigabytes when compressed, expanding to 28.9 gigabytes when uncompressed

The training data is provided in a JSON format, with each line representing a JSON object containing information about a single article. Each JSON object includes the following fields:

- "abstractText": The text of the article's abstract.
- "journal": The journal in which the article was published.
- "meshMajor": An array of MeSH terms assigned to the article.
- "pmid": The PubMed Identifier (PMID) of the article.
- "title": The title of the article.
- "year": The year in which the article was published (YYYY).

This extensive dataset serves as a valuable resource for the development and evaluation of systems designed to automatically assign MeSH terms to scientific articles. It is particularly relevant for tasks in natural language processing and information retrieval within the medical and scientific domains.

AI/ML Approach:

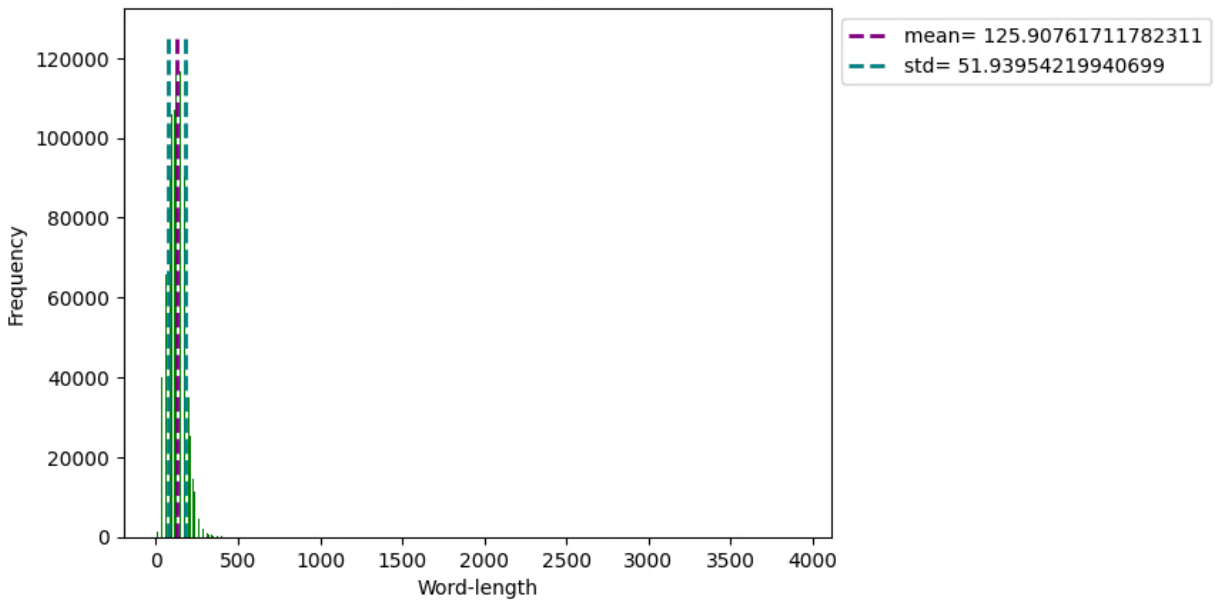
In this section, we outline the methodology employed for our analysis, which focuses on predicting MESH terms. We implemented the MeSHProbeNet in TensorFlow.

Data Pre-processing:

Here are some preprocessing steps that we adopted to significantly reduce the vocabulary size:

1. Covert to lower case (to avoid duplicates)
2. Remove HTML tags (Not relevant to prediction as they can be unique to document)
3. Contraction_mapping (Replace commonly used abbreviations like "ain't": "is not", "aren't": "are not", “can’t": "cannot", "cause": “because” etc.)
4. Removing Hyphens for meaningful words (ex: pregnancy-related -> pregnecy related. This helps reduce duplicity and avoid elimination of the words in the next steps as they might become not valuable otherwise.)
5. Making sure all the special characters and digits are removed (we focus on root word)
6. Remove those words which are only 1 character. (Is not useful)
7. Lemmatize the text using wordnet lemmatizer
8. Remove English stop words (Ex: of, the, are, etc.)

After performing the above tasks, the results the word frequency of the documents is



In the above chart we can observe that mean and std of the words per document are in the ranges we like them to be, but range of the word length is too large. It indicates there is still scope for removing words that may not contribute to the final prediction.
The goal here is to reduce the dimensionality of the inputs as it significantly affects the prediction. Dimensionality in our case is vocabulary space (Number of words)

Please note that we have 16.2 million documents and average cardinality of abstract text is 121. And 0.001% constitutes to 162 documents.

There are two experiments further conducted:

- Experiment 1: Removing less important words from the data to reduce vocabulary size.
- Experiment 2: Training a custom Tokenizer to reduce the vocabulary size.

Note that the result in both the cases is similar and did not change much. Hence, we are listing both the experiments here:

Experiment 1:
The standard processes available for us to reduce the dimensionality are:

1. Rare word removal: These are the words that occur very less times in the corpus that they don't have statistical significance towards prediction. After multiple experiments and manually looking at the quality words that are being removed, we adopted the following strategy and corresponding thresholds:
    a. Remove those words which are very less frequent in a particular document as well as very less frequent in overall corpus. Threshold chosen were 3 and 100 respectively after trial and error. That is less than or equal to 3 times in a single document and simultaneously occurring less than 50 times in whole corpus of all words. Hence compared to the 16 million documents these thresholds are very insignificant.
    b. After removing the above, we remove all those words which are very rare (not more than in 50 documents) and probably repeated in within same documents (50 times in whole corpus). This is slightly different from the above criteria as the above looks at 100 documents and strictly 3 times withing a document whereas this looks at probable repetitions in very limited documents)

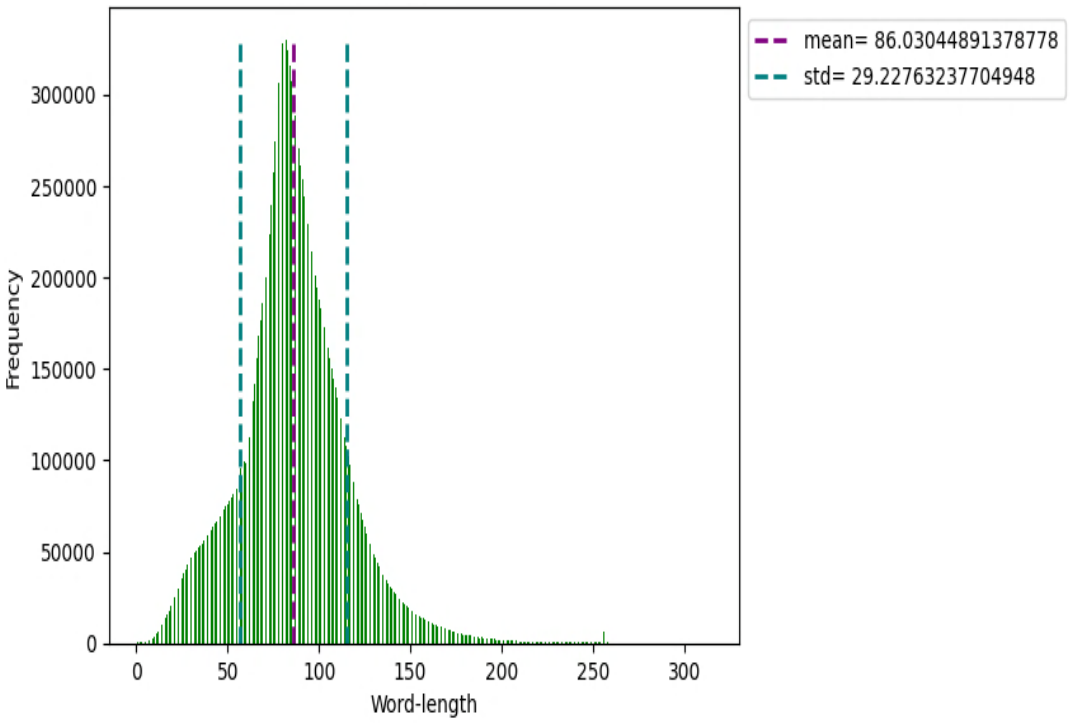
To perform this task term frequency matrix is used.

2. Most common words of the corpus removal:
    a. In this step We removed top 50 words which are highly repetitive in all the documents. Like rare words, common words don't contribute to predictions as well.
3. TFIDF scores-based word elimination:

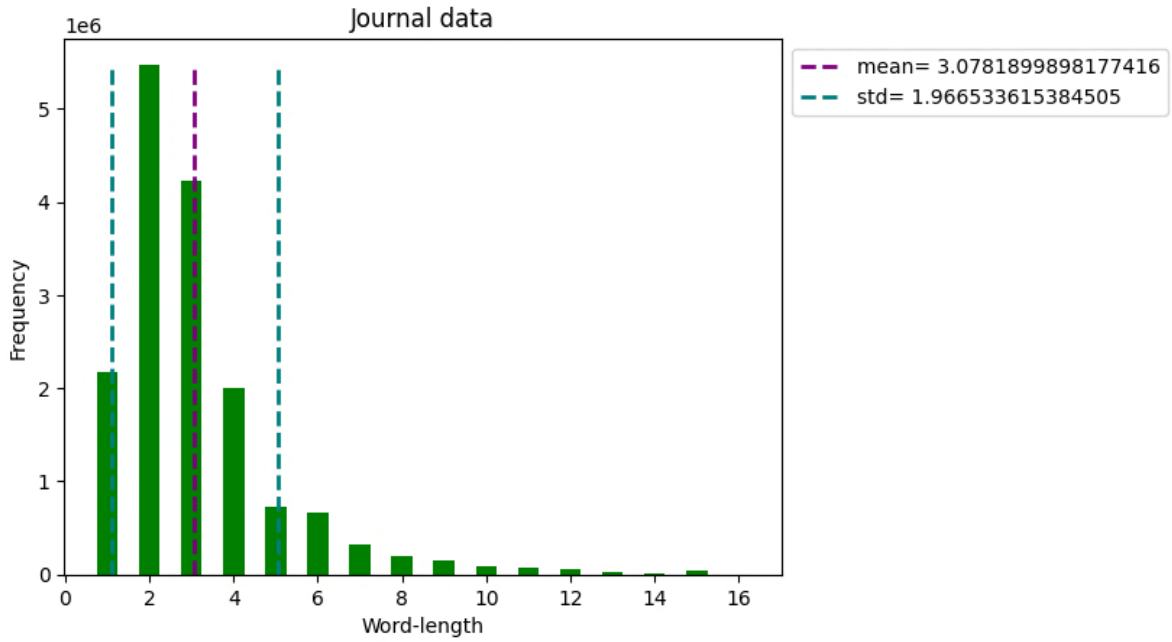
a. Term Frequency (TF): This component measures the frequency of a term within a document. It represents how often a term appears in a document relative to the total number of terms in that document.
b. Inverse Document Frequency (IDF): This component measures the significance of a term across a collection of documents. It assigns a weight to a term based on its rarity in the entire corpus. Terms that appear frequently across documents have lower IDF values, while terms that appear rarely have higher IDF values
c. The TF-IDF score for a term in a document is obtained by multiplying its TF value by its IDF value. The resulting score reflects the relative importance of a term within a specific document in the context of the entire corpus.
d. After construction TFIDF scores, we can remove the words in each document based on a global score threshold and local(within document) Threshold. Also it is important to perform this steps only after above 2 steps, as presence of rare and common words significantly affect the scores.
e. The present removal criteria is minimum (50th percentile of corpus wide thresholds, 50th percentile of within the document).
f. We are being aggressive by choosing 50th percentile but We want to first try this because, this can reduce a lot of noise. We plan to revisit this in future trials after looking at model performance)
g. Please note that, this exercise doesn't guarantee the removal of a particular word from the whole corpus. This only eliminates irrelevant words within a particular document.
h. Finally restrict only top 256 words, 15 words in document in abstractText and JournalText

After doing the above dimensionality reduction steps, the total vocabulary size is reduced to 217895. (This is a significant reduction from previously used 900000 vocab size. The previous only is attained by simply removing less than 10 infrequent words in the corpus. And none of the above steps were implemented to mimic the paper)

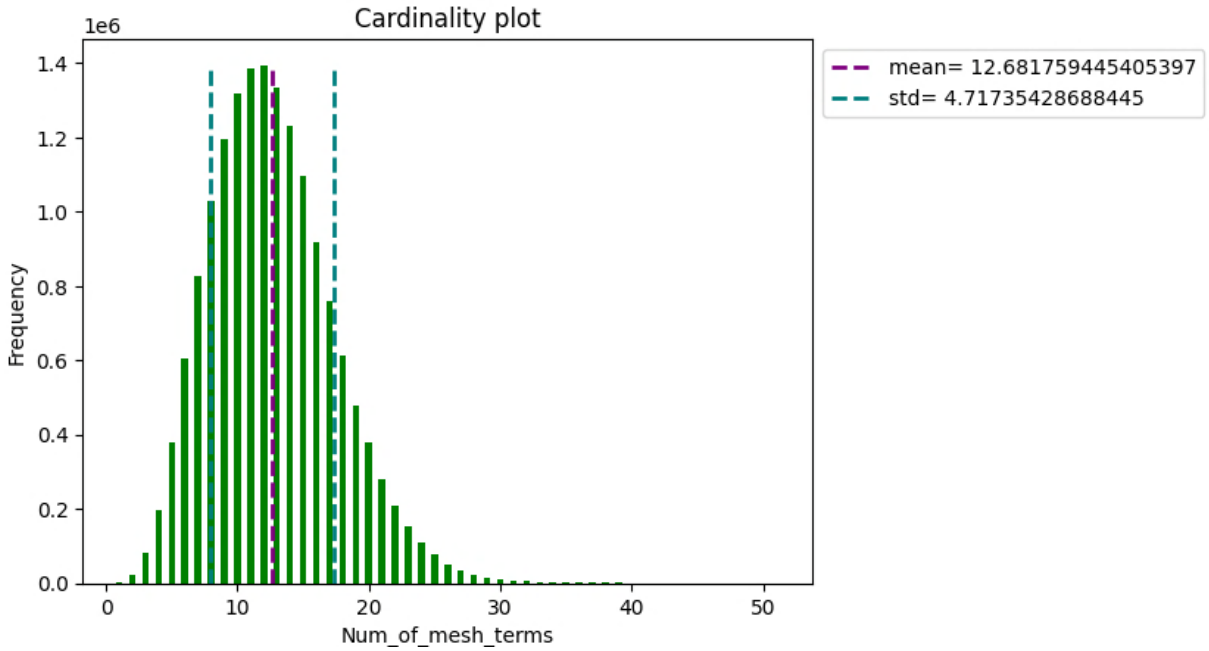
The graph for abstract text looks like this:



Graph for Journal text:



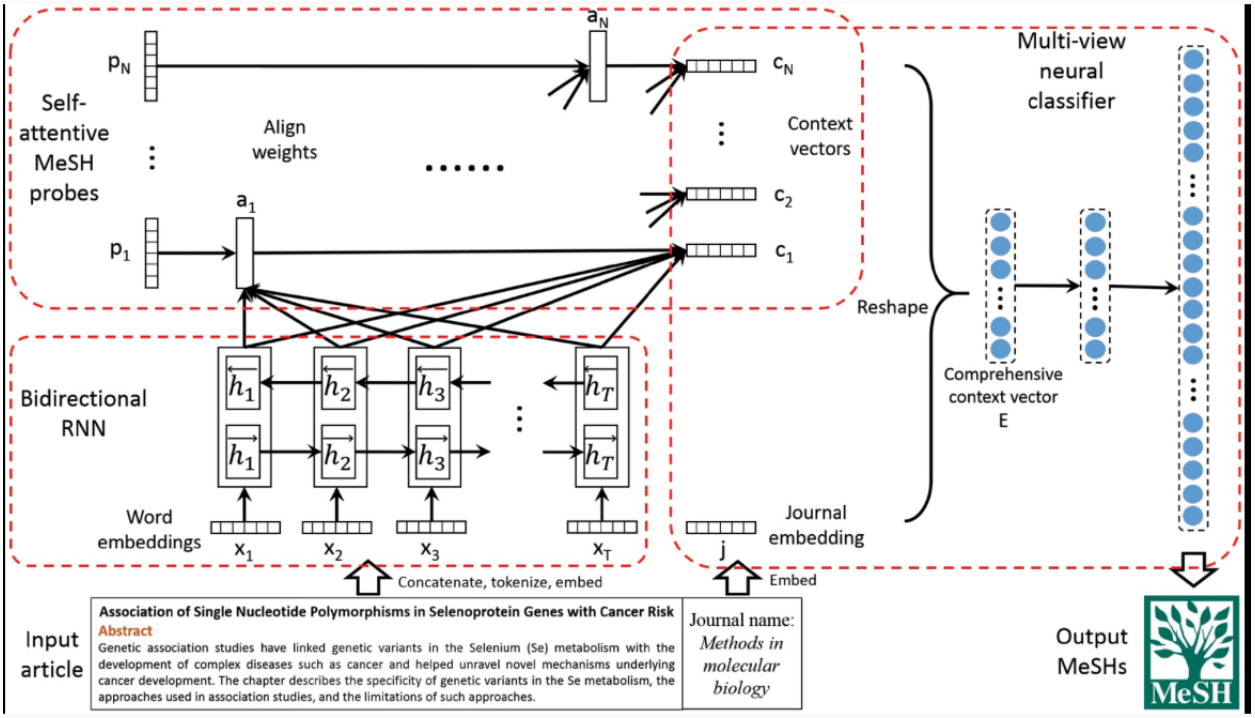
Additionally, We analyzed the cardinality of Mesh Terms per document in the new dataset:



Experiment 2- Data Tokenization:

In the context of our study, we employed the ScentencePiece Tokenizer to train a custom tokenizer specifically tailored to the 2022 BioASQ dataset. This custom tokenizer was configured with a vocabulary size of 35,000, ensuring its capacity to effectively capture the unique linguistic characteristics present in the dataset. Subsequently, this custom tokenizer played a crucial role in encoding each abstract, facilitating the transformation of textual content into numerical representations. This process is instrumental in enabling the effective analysis and manipulation of the dataset for our research purposes.

Model design:
IIAI team just re implemented the original MeshProbeNet algorithm. Here is the algo:



The neural network model designed in this study comprises several key components:

1. **MeSHProbes Layer:**
   - The `MeSHProbes` layer is a custom layer used for multi-head attention mechanisms. It is designed to operate on the input data and perform attention calculations.
   - The number of probes, specified as `n_probes`, is a configurable parameter for this layer.

2. **Model Architecture:**
   - The model architecture is defined using the Keras functional API.
   - The input layer consists of two parts: one for abstractText data (`inputs1`) and another Journal (`inputs2`).
   - An embedding layer is applied to the numerical input (`inputs1`) with an output dimension of 250. A dropout layer with a rate of 0.5 is added after the embedding layer.
   - Bidirectional GRU (Gated Recurrent Unit) layers with 200 units and return sequences are used for processing the embeddings.
   - The `MeSHProbes` layer is applied to the output of the Bidirectional GRU.
   - A dropout layer with a rate of 0.5 follows the `MeSHProbes` layer.
   - Inputs2 is also passed through the embeddings and the flattened embeddings is added to the `MeSHProbes` layer output.
   - The output layer is a dense layer with 29681 units, which corresponds to the number of classes or categories to predict.
   - Regularization techniques, such as L2 regularization, are applied in certain layers.

Training Process:

The training process involves several steps:

1. **Model Compilation:**

- The Adam optimizer is used with specific hyperparameters, including a learning rate, beta values, and weight decay.
- The model is compiled using a custom loss function (`multi_label_soft_margin_loss`) and performance metrics, including precision and recall.

2. **Training Loop:**
- The model is trained using the `model.fit()` method.
- The training data is provided by `train_gen`, and the validation data is provided by `val_gen`.
- Training occurs over multiple epochs (in this case, 5 epochs).
- Various callbacks are applied during training, including early stopping and learning rate reduction based on validation loss.

Notes:

- We also implemented a BERT based classification model. However, we did not continue with it because each epoch takes days to train. Hence, not describing the model here.
- We also used the same MeshProbenet model on a smaller dataset described in the beginning.

Metrics used for results:

The metrics used for evaluation are implemented similar to this paper
https://arxiv.org/pdf/2102.07349.pdf

Here is a screenshot of description:

Evaluation Metrics. In many multi-label classification datasets, even if the label space is large, each document only has very few relevant labels. For example, in Table 1, we show that both MAG-CS and PubMed have over 15K labels in total, but each document has 5.60 and 7.78 labels on average, respectively. Considering the sparsity of labels, a short-ranked list of potentially relevant labels for each testing document is commonly used to represent classification quality. Following previous studies on extreme multi-label text classification [27, 58, 63], we adopt two rank-based metrics: the precision at top k (P@k) and the normalized Discounted Cumulative Gain at top k (NDCG@k), where $k = 1, 3, 5$. For a document d, let $\boldsymbol{y}_d \in \{0, 1\}^{|\mathcal{L}|}$ be its ground truth label vector and $\text{rank}(i)$ be the index of the i-th highest predicted label according to the output probability $\boldsymbol{\pi}_d$. Then, P@k and NDCG@k are formally defined as

$$\text{P@}k = \frac{1}{k}\sum_{i=1}^{k} y_{d,\text{rank}(i)}.$$

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{y_{d,\text{rank}(i)}}{\log(i+1)}, \tag{21}$$

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\sum_{i=1}^{\min(k,||\boldsymbol{y}_d||_0)} \frac{1}{\log(i+1)}}.$$

It is easy to show that P@1 ≡ NDCG@1 if each document has at least one true label.

Experimental methods, validation approach:

The train: validation: test split is 80:10:10.

We used Adam with following configuration:
learning_rate=5e-4, decay= 1e-5
loss = multi_label_soft_margin_loss

All the models are trained for 5 to 30 epochs with 4096 as batch size.

Results:

The best results IIAI team could achieve are

- P@1: 0.900830090
- P@3: 0.728185177
- P@5: 0.620312512
- P@10: 0.438788086
- P@15: 0.339100927
- NDCG@3: 0.692891442
- NDCG@5: 0.568575497
- NDCG@10: 0.380031077
- NDCG@15: 0.313000906