Debiasing methods for Entity Detection in Bio-medical domain

An M. Tech. Phase II Report Submitted
in Fulfillment of the Requirements
for the Degree
of

Master of Technology

by

Kishore M (214101062)

Advisor

Dr. Amit Awekar

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work described in this thesis, titled "**Debiasing methods for Entity Detection in Bio-medical domain**" was done by **Kishore M (Roll No. 214101062)**, in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, under my advice, and has not been submitted for a degree elsewhere.*

Dr. Amit Awekar

Associate Professor

Computer Science and Engineering,

Indian Institute of Technology Guwahati,

Guwahati-781039, Assam, India.

Declaration

This is to certify that the thesis entitled "**Debiasing methods for Entity Detection in Bio-medical domain**", submitted by me to the *Indian Institute of Technology Guwahati*, for the award of the Master of Technology degree, is a genuine work that I completed under the advice of Dr. Amit Awekar. This thesis has not been submitted, in whole or in part, to any other university or institute for the award of any degree or diploma. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Kishore M

Computer Science and Engineering,

Indian Institute of Technology Guwahati,

Guwahati-781039, Assam, India.

Acknowledgements

First of all, I with utmost sincerity, proffer my deepest heartfelt gratitude to my thesis advisor **Dr. Amit Awekar**, of Computer Science and Engineering, Indian Institute of Technology, Guwahati for his invaluable, enthusiasm for my work, support, and encouragement. His constant effort and advice enabled me to gain essential and valuable lessons, which fueled my enthusiasm for this project and will also benefit me in my future career.

It brings me great pleasure to offer my sincere gratitude to my father **Mr. Muthumurugan** and mother **Mrs. Packialakshmi** for their unwavering support, encouragement, and belief in me.

Abstract

The overarching goal of our work is to remove the bias in entity detection task specific to biomedical domain. Entity detection is a task of identifying entities and labelling them. Majority of the existing models perform well on entities seen during training but are unable to generalize well on unseen novel entities. Our work focuses on removing such bias from models in biomedical domain. Currently there aren't many debiased methods specific to entity detection task. We utilized the debiasing methods which are used in different tasks and created our own baselines. We proposed a new method for debiasing and experimented them on 3 datasets. Our debiased models achieve significant gains over baseline models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Entity is a word or sequence of words which refers to the same thing. Entity detection task in biomedical domain involves identifying biomedical entities and labelling them. Currently the existing entity detection models performs well on entities seen during training but are not able to generalize well on unseen novel entities. We call this as bias and our task is to debias models that suffers from such bias.

Typical debiasing methods tries to debias the dataset by removing bias from the dataset and make the dataset unbiased. But this remains costly and requires detailed knowledge about different biases in different datasets.

The next good thing to do is debias models. Making the model not to capture bias features and learn from other useful features is model debiasing. This helps us use the existing datasets irrespective of it being biased or unbiased.

There are various debiasing methods to debias models but they are designed

for different tasks. The debiasing methods for entity detection task still remains unexplored or less explored.

Existing model debiasing methods comes under different paradigms. Some of them (but are not limited to) are listed below.

1. Two Models
2. Knowledge Distillation
3. Regularizer
4. Adding noise

Further details about these paradigms are discussed in chapter 3.

1.2 Problem Statement for entity detection

Given a sentence x_i with n tokens $x_i = (x_i^1, x_i^2, ...x_i^n)$ and labels $y_i = (y_i^1, y_i^2, ...y_i^n)$ where each $y_i^j \in (y_i^{j,1}, y_i^{j,2}, ...y_i^{j,C})$ is the predefined tag associated to each token in x_i and C is total number of possible tags. Our task is to predict the tags for each token in the given sentence.

The work in this thesis is split into following phases:

1. We first made a study of existing debiasing methods irrespective of the tasks for which they were designed.
2. Then we utilize those methods for entity detection task and consider the performance of these models as baselines.

3. We proposed little changes in existing debiasing methods.

4. The newly proposed debiasing method's performance are reported.

5. Error analysis on where we got improvement and where not are discussed.

6. Finally limitations of current work and future work about where this project can be further carried out are discussed.

Chapter 2

Dataset Analysis

2.1 Bio-medical Datasets

There are several Bio-medical datasets available for research purpose. Among them we have used 3 most popular and commonly used datasets. They are,

- BC5CDR
- NCBI-disease
- Medmentions

2.1.1 BC5CDR

BC5CDR is one of the commonly used dataset in entity detection and entity recognition task. The BC5CDR (BioCreative V Chemical Disease Relations) dataset is a widely used benchmark dataset for chemical and disease entity recognition, entity detetction and relation extraction. It was created as part of the BioCreative V

challenge, a community-wide effort to advance biomedical text mining and natural language processing techniques. The BC5CDR dataset consists of PubMed abstracts annotated with chemical and disease entities and their relations. BC5CDR corpus is created from 1500 PubMed articles which has 4409 annotated chemicals, 5818 diseases and 3116 chemical-disease interactions. This dataset is manually annotated. We won't be using the chemical disease interactions as they are not useful for our task.

2.1.2 NCBI-disease

As the name says this corpus consits only of disease entities being labelled as entities. The NCBI-disease dataset is a resource for disease name recognition and normalization, compiled by the National Center for Biotechnology Information (NCBI). It consists of abstracts from PubMed, annotated with disease mentions and their corresponding concepts from the Medical Subject Headings (MeSH) vocabulary. The annotations include the offsets of the disease mentions and their corresponding MeSH identifiers. NCBI-disease is a manually annotated dataset. It is created from 793 pubmed abstracts among which 593 abstracts are used in training and 100 each for testing and development set. It has 6892 disease mentions being labelled as entities.

2.1.3 Medmentions

1 MedMentions is a large corpus of biomedical papers annotated with mentions of UMLS entities. It was created by the Chan Zuckerberg Initiative Meta team and released in 2019. Medmentions is a manually annotated dataset. it differs from other

dataset in terms of size and its diverse range of covering entity mention of different classes. this corpus is made of 4392 abstracts and titles of pubmed articles. It consists of fine grained entity types of 128 entity classes. This is the biggest dataset available for bio-medical domain.

The formats of all three datasets discussed above are almost similar and a sample is shown in figure 2.1

```
8701013|t|Famotidine-associated delirium. A series of six cases.
8701013|a|Famotidine is a histamine H2-receptor antagonist used in inpatient settings for prevention of stress ulcers and is showing increasing popularity because of its low cost. Although
all of the currently available H2-receptor antagonists have shown the propensity to cause delirium, only two previously reported cases have been associated with famotidine. The authors
report on six cases of famotidine-associated delirium in hospitalized patients who cleared completely upon removal of famotidine. The pharmacokinetics of famotidine are reviewed, with no
change in its metabolism in the elderly population seen. The implications of using famotidine in elderly persons are discussed.
8701013     0     10    Famotidine  Chemical    D015738
8701013     22    30    delirium    Disease     D003693
8701013     55    65    Famotidine  Chemical    D015738
8701013     156   162   ulcers      Disease     D014456
8701013     324   332   delirium    Disease     D003693
8701013     395   405   famotidine  Chemical    D015738
8701013     442   452   famotidine  Chemical    D015738
8701013     464   472   delirium    Disease     D003693
8701013     537   547   famotidine  Chemical    D015738
8701013     573   583   famotidine  Chemical    D015738
8701013     689   699   famotidine  Chemical    D015738
8701013     CID   D015738     D003693
```

Fig. 2.1 Format of a sample in raw corpus

We preprocess this dataset to a different format to make it use in our entity detection task. The new format is shown in figure 2.2,

The raw corpus starts with pubmed id followed by |t| which means this is a title and has the sentence. We extract all the sentences in title and abstract (|a|). Further it has 0-based character index of starting and ending of an entity mentions in both title and abstract of corresponding pubmed id.After the position of entity mention it has the original entity mention which appeared in sentences and has its entity class type. The last alhpanumerical value indicates the UMLS concept id of the entity. The last line of the sample has the Chemical Induced Disease id's mentioning which chemical id induces which disease id. The medmentions corpus has slightly different format as such instead of entity type class, a semantic id of that particular class is

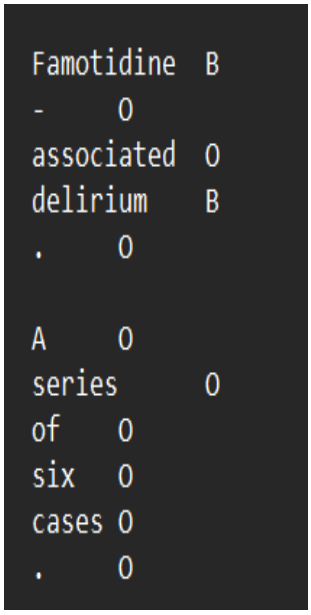

Fig. 2.2 Format of a sample in preprocessed corpus

present by which we can identify the class.

We convert the following raw corpus into a simpler format to use in our entity detection task, We take each sentence from the raw corpus and tokenize them to separate words. The tagging scheme used here is IOB format.

- B - Beginning of an entity
- I - Inside of an entity
- O - Outside of an entity

Dataset	Category	Sentences	B tags	I tags	O tags
BC5CDR	train	4841	8793	3502	101260
BC5CDR	devel	0	0	0	0
BC5CDR	test	5146	8550	3269	107352
NCBI-disease	train	5421	5111	6048	12732
NCBI-disease	devel	0	0	0	0
NCBI-disease	test	932	948	1052	22149
Medmentions	train	29232	196039	102501	434115
Medmentions	devel	0	0	0	0
Medmentions	test	9763	65549	33274	146095

Table 2.1 Properties of Datasets

Chapter 3

Literature Survey

This survey attempts to give readers knowledge about existing debiasing approaches that can be used in Entity Detection (ED) task to avoid bias in ED models. As mentioned in chapter 1 about different paradigms, each of them are unique in their own way. The details about each paradigm are discussed below.

3.1 Two Models

As the name says, debiasing methods which comes under this paradigm uses two models for debiasing. The idea of using 2 models is to make one model capture bias and using this bias model we make other model to be robust. The model which captures bias is referred to as bias model and the debiased model is referred to as main model. The main challenge here is to design a proper bias model in a way that it captures bias. This is the high level idea of using two models for debiasing. Questions like "How to make a model robust with the help of bias model

?" and "How to utilize the outputs of 2 models ?" will be answered below. Earlier approaches requires knowledge about dataset bias to build a bias model. The idea of those methods are to first identify the biased samples and then make the main model learn from unbiased samples. With the knowledge of dataset bias, a bias model was designed in a way that it identifies biased samples.

1. **Bias Product:** [1]

   A sample x can have both bias features and important features in it. It is important for a model to not capture bias features and learn more from important features. But usually what happens is bias features are easy to learn and the correlation between bias features and output class is very high. So by debiasing what we try to do is make the model learn more from important features. As mentioned the bias feature of sample is x^b and important features of the sample sample be x^{-b}. The assumption made by the authors here is that there exists a conditional independence between x^b and x^{-b}.

   The bias model used here is a pre-trained bias model which was designed with the knowledge about dataset bias. This bias model is made in a way that it captures bias features x^b.

$$\hat{p_i} = softmax(\log(p_{Mi}) + \log(p_{Bi})) \tag{3.1}$$

   where p_{Mi} is the probability distribution from main model and p_{Bi} is the probability distribution from bias model for a sample i.

   The intuition here is that when a sample is biased, the bias model predicts

ground truth with high confidence. Now adding the logarithmic value of bias model probability vector to the logarithmic values of main model probability vector will decrease the loss saying that it is a biased sample so don't learn much from those samples.

2. **Learned-Mixin:** [1] Sometimes when the main model predicts well and because of the poor performance of bias model the main model has to compensate. This makes the main model to change its behavior and doesn't perform well in out-of-domain setting. So by adding a weight of how much to believe from bias model helps the model generalize better.

$$\hat{p_i} = softmax(\log(p_{Mi}) + g(x_i)\log(p_{Bi})) \tag{3.2}$$

where $g(x) = softplus(w.h_i)$, w is a learned vector, h_i is the last hidden layer of model main model for a sample x_i, and $softplus(x) = \log(1 + exp(x))$.

Again another issue that happened for few samples is that the weight we add becomes 0 ($g(x_i) = 0$). Now what happens is the bias model predictions is not taken into account which means no debiasing done for those samples. So to handle this case a entropy penalty was added to the loss.

3. **Learned-Mixin +H:** [1]

An entropy penalty R is added to the loss:

$$R = wH(softmax(g(x_i)\log(b_i))) \tag{3.3}$$

where $H(z) = -\sum_j z_j \log(z_j)$.

4. **Residual Fitting:** [3]

   Neural models generally doesn't perform well when trained on insufficient features ($I(x)$). So if such neural network is predicting well for some samples, we call them as biased samples. The approach used here is to fit the residuals of the bias model. When the bias model is not able to perform well on some samples, our main model should learn more patterns than bias model as those samples are unbiased samples. This is done by adding the logits of the bias model to the logits of the main model before calculating loss. Let $f_M : X_i \to \mathbb{R}$ and $f_B : X_i \to \mathbb{R}$ be the biased and debiased classifiers. The pre-softmax logits of both classifiers are added together to minimise the loss when the sample is biased and maximize loss when sample is unbiased and our model makes wrong prediction. The overall objective of the minimizing emperical risk of the described method can be defined as,

$$\min_{\phi} E_P[L(f_B(I(x); \theta^*) + f_M(x; \phi), y)] \tag{3.4}$$

5. **Greedy gradient ensemble:** [4]

   Making the main model focus more on harder examples can make it generalize better.Here authors made an attempt to tell the main model about the importance of a each samples with the help of gradients. This can be done by using negative gradients of samples from bias model and use them as pseudo labels. As they found out that there is distribution bias, The bias model used here is a

class prior which calculates prior probability for each sample from the training dataset. The main model is learned from Loss of predictions made by it and the negative gradients of predictions made by bias model. Intuitively the main idea here is for a sample that is said to be fit to bias, the negative gradient of loss will be relatively low which tells the main model to learn from more complex examples. The main model is learned by,

$$Loss = CE(\sigma(\bar{(}y), -\nabla CE(B_d, y)) \tag{3.5}$$

where $\bar{(}y)$ is the output logits from main model and B_d is the class prior probability.

Approaches discussed till now required prior knowledge about bias to design a bias model. The upcoming debiasing methods in this paradigm doesn't require prior knowledge about bias. Some of them train the bias model simultaneously along with main model and others train a small neural network that captures easy features which includes bias.

6. **Annealing mechanism:** [5]

   The authors made an observation that the initial features captured by the model are easy features which includes bias features.The model captures biased features just after seeing a small amount of training data. So they decided to create a neural network with subset of training data such that it captures bias. To make this model overfit to bias, it is trained for multiple epochs. This bias model can be used with the existing debiasing approaches like re-weighting,

product of experts, confidence regularization (discussed in section 3.2) can be used. As this neural network is capable of capturing multiple bias, a lot of training samples might be down-weighted which results in reduce in size of training data. This results in performance drop of in-distribution data. So to avoid this they proposed an annealing mechanism, such that model learns from all samples including biased ones. Making the model learn from all samples is done by lowering the probability values of bias model at each training step.

$$p_B^{(i,j)} = \frac{p_B^{(i,j)^{\alpha_t}}}{\sum_{c=1}^{C} p_B^{(i,c)^{\alpha_t}}} \tag{3.6}$$

The value of α_t is decreased linearly throughout the training by,

$$\alpha_t = 1 - t(\frac{1-a}{T}) \tag{3.7}$$

where T is the total number of training steps. and a is set as an hyperparameter ($a \in [0, 1]$).

7. **Mixed Capacity Ensemble:** [2]

   Earlier discussed methods make bias model with the help of knowledge about bias. Now to make a bias model domain generic one, the authors tried to create a simpler neural network as bias model and make it learn simultaneously as main model. The assumption made here is the feature captured by bias model and main model is conditionally independent.

The ensembled predictions and predictions of both models are made by,

$$\hat{y}_i^E = softmax(\log(f_M(x_i)) + \log(f_B(x_i)) + \log(p_y))$$
$$\hat{y}_i^B = softmax(\log(f_B(x_i)) + \log(p_y))$$
$$\hat{y}_i^M = softmax(\log(f_M(x_i)) + \log(p_y))$$

where p_y is class prior probabilty calculated for each token training set. The ensembled model loss is calculated as,

$$Loss(L) = \sum_{i=1}^{n} CE(\hat{y}_i^E, y_i) + wCE(\hat{y}_i^M, y_i) \tag{3.8}$$

where w is hyperparameter. The loss of bias model is added to main model to make surer that both bias model and main model captures different features. In other words it can be said that bias model captures simpler features and main model captures complex predictive features. As the main model is trained to learn complex features, during inference time only main model is used.

8. **Generalized Cross Entropy Loss:** [6] Neural networks tends to rely on bias only when it is easy to learn than the target feature. With this observation they designed a model that is intentionally trained to fit to bias. This is done by up-weighting the gradients of samples which are simpler to predict. By doing this we make sure model focuses more on simpler features which includes bias features. Making the model do such thing can be done by Generalized Cross

Entropy loss (GCE)[7] .

$$GCE(p_B(x;\theta), y) = 1 - \frac{p_{By}(x;\theta)^q}{q} \tag{3.9}$$

where $p_B(x;\theta)$ is the softmax output of bias model and $p_{By}(x;\theta)$ is the softmax value assigned to target class y. When $\lim_{q\to 0}$ GCE becomes same as standard Cross entropy loss (CE). While bias model is intentionally trained to fit bias by GCE, Main model is trained simultaneously using weighted cross entropy loss to be debiased. The weight is calculated using cross entropy loss of both bias and main model.

$$W(x) = \frac{CE(f_B(x), y)}{CE(f_B(x), y) + CE(f_M(x), y)} \tag{3.10}$$

This weight indicates whether the sample is bias aligned or bias conflicting. When a sample is bias aligned, the loss from bias model will be low and small weight will be assigned to those samples and when a sample is bias conflicting , the loss from bias model will be high and large weight will be assigned to those samples.

9. **Debiased Focal Loss:** [8] Focal loss firstly proposed in [9] to down-weight samples that are classified with high confidence. Now using the same idea in [**?**],focal loss is used to down-weight samples which are predicted with high confidence by out bias model. This allows the model to focus on hard samples.

Debiased focal loss is defined as,

$$LC(\theta_M;\theta_B) = -\frac{1}{N}\sum_{i=1}^{N}(1-\sigma(f_B^{y_i}(x_i)))^{\gamma}\log(\sigma(f_M^{y_i}(x_i)) \tag{3.11}$$

where N is number of samples in dataset and $f_B^{y_i}(x_i)$ and $f_M^{y_i}(x_i)$ are the logits from corresponding classifiers.

10. **Debias encoded representation:** [10] With the knowledge of bias in the dataset, [10] proposed an adversarial debiasing technique which deals with removing bias from the representation vector and generate unbiased representation.This is done with the help of gradient reversal. The main idea here is if model is able to make right prediction with insufficient features, then that feature representation has bias, so we reverse the gradient to generate better representation. This makes the representation vector unbiased and conveys unbiased information. [8] used a discriminator which is trained with insufficient features and gradient reversal to generate better representation and a classifier which is debiased. This proposed method is for NLI task and they are able to get better performance only using hypothesis which is bias. So to generate better encoded representation for hypothesis, the gradient reversal is done.

$$\theta_{Eh}^{\text{new}} = \theta_{Eh}^{\text{old}} - \frac{1}{m}\sum_{i=1}^{m}\nabla_{\theta^{E_h}}[\log p_C(y_i|E_h(x_i^h), E_p(x_i^p))] + \gamma\frac{1}{m}\sum_{i=1}^{m}\nabla_{\theta^{E_h}}[\log p_D(y_i|E_h(x_i^h))] \tag{3.12}$$

where $+\gamma\frac{1}{m}\sum_{i=1}^{m}\nabla_{\theta^{E_h}}[\log p_D(y_i|E_h(x_i^h))]$ is the gradient reversal part and E_h

and E_p are hypothesis and premise encoder, $\log p_C(y_i|E_h(x_i^h), E_p(x_i^p))$ is the prediction of classifier with both hypothesis and premise representation on ground truth class and $\log p_D(y_i|E_h(x_i^h))$ is the prediction of discriminator with only hypothesis representation on ground truth class. If this discriminator is able to predict well only with insufficient features then gradients are reversed and better representation is produced.

11. **Self Debiasing framework:** [11]

    Lower layers of transformer based model captures lexical and semantic features. This observation made the authors to create a bias model that takes intermediate representations of main model as input and predict tags. The main model used here is transformer based model. An attention scalar value is created for each word by a feed forward neural network and normalized. High value of attention for a token indicates that it has bias features. Bias model makes predictions with the intermediate representation h_i^k of k^{th} layer along with attention. As high attention value for a token indicates it has bias features, the bias model is said to fit to bias. Now to de-emphasize these features from main model, weighted noise vector is added to the hidden representation. For tokens with high attention value large noise is added to make the main model not to capture those bias features. The intermediate representations of main model are updated by,

$$h_i^k \leftarrow a_i z_i + h_i^k \tag{3.13}$$

where a_i and z_i is the attention value and noise for token i.

If biased model is highly confident on ground truth (> 0.8) then those samples are re-weighted by $(1 - p_B^{yi})$. Along with this if main model predicted with less confidence for a sample ($<= 0.5$) then the predicted value s used as a weight to bias model for that sample.

12. **Feature Augmentation:** [12] Majority of the samples in dataset are biased. If we split dataset into 2 categories like bias aligned samples which are biased and bias conflicting samples which are unbiased, the bias aligned samples seems to be big in number. The authors created a model with 2 pairs of encoder, classifier. One pair of encoder-classifier (E_B, C_B) is trained to overfit to bias with the help of Generalized Cross Entropy Loss [7] and other pair of encoder-classifier (E_M, C_M) is trained to make useful learn from useful features. A relative difficulty score (W(x)) is found for each samples. z_M and z_B are embedding of sample x produced by encoders E_M and E_B respectively. These embedding vectors are concatenated to $z = [z_B, z_M]$ and passed to classifiers C_i and C_M.

$$W(z) = \frac{CE(C_B(z), y)}{CE(C_B(z), y) + CE(C_M(z), y)} \tag{3.14}$$

Bias conflicting samples will be having high importance and (E_M, C_M) are trained to learn from these samples. The overall objective becomes,

$$L_{dis} = W(z)CE(C_i(z), y) + \lambda_{dis} GCE(C_b(z), y) \tag{3.15}$$

With the above mentioned loss the main pair of encoder-classifier (E_M, C_M) learns only from bias conflicting samples and those samples are not scarce. This makes model learn from only few samples of training data. Now to utilize bias aligned samples and learn important features in them a new technique call feature swapping for augmentation is proposed. Inside a mini-batch, there will be feature vectors for samples generated by E_B and E_M. Earlier for a sample x_i the feature vectors generated by 2 encoders for that sample x_i were concatenated. Now the feature vectors are swapped. The feature vector generated by E_B for a samples i is now concatenated with feature vector generated by E_M for some other sample j($z = [\tilde{z}_B, z_M]$). The intuition here is that as the important features and bias features concatenated are from two different samples, there won't be much correlation between them. This makes the model to utilize all bias aligned samples for training. Along with the L_{dis} loss function another loss function is used to make the model learn from augmented features.

$$L_{\text{swap}} = W(z)\text{CE}(C_M(z_{\text{swap}}), y) + \lambda_{\text{swap}}\text{GCE}(C_B(z_{\text{swap}}), \tilde{y}) \tag{3.16}$$

where $\tilde{y}$ is the ground truth of swapped sample. The overall combined loss function is,

$$L_{total} = L_{dis} + \lambda_{swap} L_{swap} \tag{3.17}$$

where λ_{swap} controls the importance of feature augmentation.

The feature augmentation should be applied after a few number of warm up

training, so that both encoder-classifier pairs learn to capture corresponding feature attributes. The encoder classifier pairs are updated with 2 loss functions L_{dis} and L_{total}. L_{total} should be calculated after some number of warm-up training.

3.2 Knowledge Distillation

Knowledge distillation is a generally used to transfer knowledge from large complex neural network to a simpler neural network. The goal is to achieve similar performance on a task while reducing the computational resources required for inference. The process of knowledge distillation involves training a teacher model, on a particular task. The teacher model learns to make accurate predictions and captures rich information about the data. Once the teacher model is trained, a smaller student model is trained to mimic the behavior of the teacher model. The student model is usually shallower or has fewer parameters than the teacher model, making it computationally efficient. During training, the student model tries to reproduce the output probabilities and learn similar feature representations as the teacher model.

1. **Confidence Regularization:** [13]

   We call teacher model as (F_t) and main model as (F_m) and bias model as (F_b). The idea here is to regularize the confidence of teacher model predictions using bias model predictions before distilling it to main model. Here both teacher model and main model is parameterized identically.

   For a given sample x, the teacher model prediction be $F_t(x) = [p_1, p_2, ..., p_C]$, is

the probability distribution among all possible classes C. Now before distilling it to student model, we scale this probability vector with the help of bias model's predictions. Let β_i be the probability value assigned by bias model for ground truth class. Using this value the teacher model probability vector is scaled by,

$$S(\hat{p}_i, \beta_i)_j = \frac{\hat{p}_{i,j}^{(1-\beta_i)}}{\sum_{k=1}^{K} \hat{p}_{i,k}^{(1-\beta_i)}} \tag{3.18}$$

for $j = 1, ..., C$. The scaling function reduces the confidence of teacher model when an instance is biased (β_i is close to 1) and teacher model prediction remains unchanged when a instance is unbiased (β_i is close to 0). Finally the loss of main model is calculated between its prediction and scaled output of teacher model.

$$L(x_i, S(\hat{p}_i, \beta_i)) = -S(\hat{p}_i, \beta_i). \log F_m(x_i) \tag{3.19}$$

2. **Learning with Biased Committee:** [14]

   Instead of using single neural network as bias model, the authors decided to use committee of classifiers as bias models. The idea is to make each classifier in committee to capture different bias. Let the committee has k classifiers $f_1, f_2, ..., f_k$ and the main classifier m. First there is a common feature extractor which generates representations for each sample in the training data.

   Random subsets are taken from training to pass them to different classifiers of

committee. Those subsets are taken with replacement and they are denoted by $S_1, S_2, ..., S_m$. Each classifier f_l in committee is trained with random subset S_l of training data. Initially a warm-up training is done with those subsets of data with each classifier to make the classifiers learn to assign weights for each samples. Once warm-up training is done for few epochs to the committee of classifiers, they will be able to identify whether a sample is biased or not. By using this a weight is calculated for each sample and the main model is trained using weighted cross entropy loss.

The weight is calculated with the help of outputs from the committee.

$$w(x) = \frac{1}{\sum_{l=1}^{k} 1(f_l(x) = y)/k + \alpha} \tag{3.20}$$

where k is size of committee. This weight indicates that if more classifiers of committee are able to predict well for a sample then, it means it is a biased sample and weight assigned will be less. And in other way when most of the classifiers of the committee aren't able to predict a sample, then it is unbiased and more weight will be assigned to that sample. The main model is trained with weighted cross entropy loss.

$$\mathcal{L} = w(x_i)CE(m(x_i), y_i) \tag{3.21}$$

While debiasing the main classifier, the authors also debias the committee. This is done to make the committee learn to predict easy bias conflicting samples too. Already the bias committee trained with random subset of data was

able to predict bias aligned samples. So now theya are making it to predict easy bias conflicting samples. This makes the main model to learn from hard bias conflicting samples and so it generalizes better. Debiasing committee to some extent is done by letting the committee know the quality of main model. This is done by distilling the information about main model to committee by minimizing KL divergence loss.

$$\mathcal{L}_{KD} = \sum_{l=1}^{m} \sum_{(x,y)\in B\setminus S_l} KL\left(softmax\left(\frac{g(x)}{\tau}\right), softmax\left(\frac{f_l(x)}{\tau}\right)\right) \tag{3.22}$$

where τ is temperature parameter. This $\mathcal{L}_{KD}$ loss is applied for each classifier in committee in a way that the sample does not belong to subset S_l on which that classifier is trained. This makes all the classifiers in the committee to be different. The final onjective of the committee is to minimize the combination of both CE loss and KD loss

$$\mathcal{L}_{committee} = (1-\lambda)\mathcal{L}_{CE} + \lambda\mathcal{L}_{KD} \tag{3.23}$$

where λ is balancing hyperparameter.

3. **Introspective distillation:** [15]

   The term introspective distillation means that before distilling knowledge to student model , an introspection is to be done. Here two causal teacher models are used. Causal models are used to capture the causal relationships between input sample and output class. The word "causal" means "relating to cause

and effect". It can also mean "having the power to cause something".

As mentioned earlier 2 causal models are used to capture the causal relationships. One causal teacher model is trained to capture the relation in In-domain data and another causal teacher model is trained to capture the relation of OOD data. But how to get OOD data as we don't have them during training?. To handle this problem a counterfactual based method is used [19]. the ID-teacher model learned from the factual training data and OOD-teacher model learns from counterfact of the training data and makes predictions P^{ID} and P^{OOD} respectively. Once the prediction is made, introspection is done. For a sample, if ID teacher model makes better prediction, it means this factual sample is biased to in-domain and we tell the student model to learn more from OOD-teacher. And the other way if OOD teacher model makes better prediction then we tell student model to learn more from ID teacher as this counterfactual sample is biased towards OOD world.

The score (s^{ID}, s^{OOD}) used to tell which model made better prediction is the reciprocal of standard Cross entropy loss of both models.

$$s^{ID} = \frac{1}{CE(P^{ID} + P^{GT})}, \qquad s^{OOD} = \frac{1}{CE(P^{OOD} + P^{GT})} \tag{3.24}$$

where P^{GT} is the ground truth.

Now to get the relative importance of both teacher model, a weighting scheme

is used. Weight is calculataed by the scores of ID teacher and OOD teacher.

$$w^{ID} = \frac{s^{OOD}}{s^{ID} + s^{OOD}}, \qquad w^{OOD} = \frac{s^{ID}}{s^{ID} + s^{OOD}} \tag{3.25}$$

If $w_{ID} > w_{OOD}$, then the student model learns maximum from ID teacher model and vice versa. The final probability vector which is distilled to student model is,

$$P^T = w^{ID}.P^{ID} + w^{OOD}.P^{OOD} \tag{3.26}$$

The student model is trained to minimize the KL divergence loss of predictions of student mode P^S and distilled teacher probability vector P^T .

$$\mathcal{L} = KL(P^T, P^S) = \sum_{x \in X} P^T(x) \log \frac{P^T(x)}{P^S(x)} \tag{3.27}$$

3.3 Regularization

1. **Entangling and Disentangling:** [16] When the bias features is in form of attributes, we can decorrelate samples having same bias class and correlate samples having same target class. If we can correlate samples having same target class and make the model learn from them and de-correlate samples having same bias class debias their representation then model's generalization capability can be increased. This can be done with the help of a regularizer. Here as discussed [**?**] tried to debias the representation of the feature vectors with the help of regularizer. the feature vector is taken from intermediate representation

of k^{th} layer. The overall objective of our model becomes minimizing the loss (L) and the regularizer term (R). Regularizer R is comprised of 2 terms, disentangling $R \perp$ and entangling $R_{||}$ terms. A similarity matrix (G) is built with each pairs of samples and each entry $g_{i,j}$ tells the correlation between sample i and sample j and their values lines in [-1,1]. Gramian similarity matrix can be built by,

$$G = (\tilde{y})^{'} \cdot y, \tag{3.28}$$

where y is output from k^th layer for a batch and $(.)^{'}$ is the transposed matrix. Disentangling term: To disentangle patterns which has same bias features, corresponding gramian matrix is built from samples having same bias class. Then there are enforced to de-correlate with the help of the following regularizer term,

$$R \perp = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{(M^{-,b})^2} \sum_{i,j} |g_{i,j}^{-,b}| \tag{3.29}$$

where $(M^{-,b})$ is the cardinality of samples having same bias class irrespective of target class. This regularizer term tries to de-correlate samples having same bias class and is done for all types of bias features. Basically it minimizes the off-diagonal elements of the gramian matrix.

Entangling Term: Now to make strong correlation between samples belonging to same target class, an entangling term is required. So, a gramian matrix is built from samples having same target class. Using similar approach as like dis entangling term wont work here. The reason is as we try to entangle all

samples having same target class irrespective of bias features, it might happen that samples having same bias features gets correlated. So to avoid this they maximize the correlations of samples having same target class and different bias class. This is done by ,

$$R_{||} = 1 - \frac{1}{M}\sum_{i=1}^{M}\frac{1}{\sum_{b\neq B(y_i)} M^{T(y_i),b}}\sum_{j}\overline{\delta}_{B(y_i),B(y_j)} \cdot g_{i,j}^{T(y_i),-} \tag{3.30}$$

where M is the batch size, $B(y_i)$ gives the bias class of sample i and $T(y_i)$ gives the target class of sample i, $M^{T(y_i),b}$ is the cardinality of samples having same target class as y_i and different bias b.

$$\overline{\delta}_{a,b} = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases} \tag{3.31}$$

2. **Diversity Regularizer:** [17]

   The aim of this method is to overcome simplicity bias [20]. Generally models trained with normal stochastic gradient descent tends to capture simpler features than complex useful features. These simple features highly correlate with output classes and perform better for in-domain data. But same model fails to generalize better in real world data. Such bias is called simplicity bias. Models which learns from these features are generally said to be biased as they haven't learned about the actual task instead learned shortcuts. So a new regularizer based method is proposed to make model not to capture simplicity bias and

generalize better.

The idea here is to use multiple set of classifiers $g_k()$ to be trained parallelly. But they wanted all classifiers to learn different features. To make this possible, initializing models with different parameters, different hyper-parameters was not useful as doing these doesn't avoid all models to converge at same point. So to force each model to differently capture features, a regularizer is used. This applies penalty with respect to the alignment of input gradients.

A feature extractor $f(x)$ is used to generate features for sample x. This feature extractor is pre-trained and the output of this feature extractor is sent as input to all classifiers. Diversity induces complexity as model that learned default will capture the simplest features. With the regularizer other model learns complex features and has complex decision boundaries.

Noe to quantify diversity among classifiers, the authors tried to not converge any 2 models at one point. They are doing his by finding similarity of gradients of each pair of model at each hidden layer. the similarity of output gradients with respect to their input gradients for 2 models is found by.

$$\delta_{g_i,g_j}(h) = \nabla_h g_i^*(h).\nabla_h g_j^*(h) \tag{3.32}$$

where h is the input representation and $\nabla_h g_i^*$ is the gradients of classifier i. The overall objective of the classifiers is minimizing Cross entropy loss and the

diversity regularizer term.

$$\min_{\theta} \sum_{i}^{n} CELoss + \lambda \sum_{i \neq j} \sum_{k}^{K} \delta_{g_i, g_j}(h^k) \tag{3.33}$$

where n is number of classifiers, K is the number of samples, λ controls the strength of diversity regularizer.

3.4 Adding Noise

1. **Add learnable noise:** [18] Generally noise is added to entity mentions to make the model focus more on context and make predictions. This makes sure that model doesn't rely only on entity mentions for prediction. Here the noise are added to entity mention which satisfies the following constraints. Entity mentions should be preceded or followed by 3 context words. We have a sentence with n tokens ($x = x_1, x_2, ...x_n$) and labels ($y = y_1, y_2, ...y_n$) associated for each token in x. Now a noisy label $y^{'}$ is created by randomly replacing mention type from set of pre-defined class (Dis -> Chem or Chem -> Drug etc). This replacement is done in uniform distribution and it will not be replaced by it's ground truth type. A noisy embedding matrix is created $E^{'} \in \mathbb{R}^{m \times d}$ where $m = |C| \times |C - 1|$ meaning number of switching possibilities over C classes and d is the dimension of input representation. For tokens whichever has noisy labels, the corresponding noise is added from the noisy matrix and for tokens without noisy labels, a zero vector of dimension d is added. The model is extended by generating additional logits with respect to noisy tags

using a learnable projection layer. So the model makes two predictions, one with respect to original label and another with respect to noisy label. The noisy embedding matrix is learned with the loss of noisy tags and softmax outputs from projection layer. Final objective of this model is to minimize two losses. The noisy embedding matrix and noisy label projection layer are trained independently from original model parameters. As we are adding learnable noise to the entity mentions, the model tries to collect evidence from context and make predictions. This makes model generalize better to unseen novel entities.

Chapter 4

Experiments

This chapter discusses about all the experiments and results made by us for debiasing in entity detection. We chosen 3 methods from two models paradigm and used it as our baseline. The methods we chosen are,

1. Bias Product
2. Reweighting
3. Learned MixinH

These methods are designed for different NLU tasks such as VQA, NLI etc. We utilized their methods in Entity detection task. To do so, we require one main model and one bias model.

1. **Main Model:** We fixed our main model as Bio BERT + CRF model. the architecture of main model is shown in 4.1. Initially the sentences are tokenized and padded to equal length. Then all the sentences are fed to BioBert model

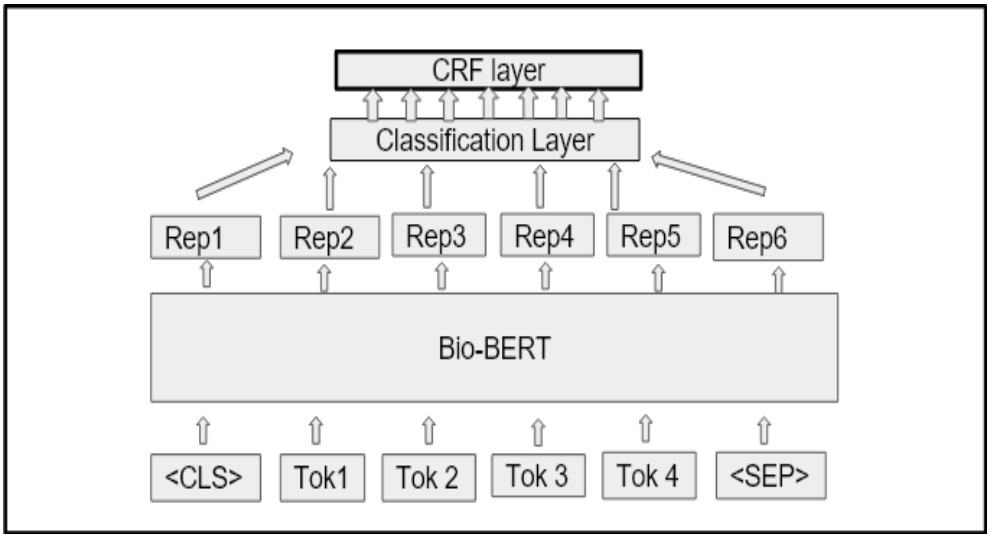

Fig. 4.1 Main model

which generates embedding for each token. These embeddings are then sent to a classification layer and generates probability distribution among all possible classes. These values are further used by CRF layer to decode labels. CRF layer predicts the label of current token by its probability value and also by considering the label of previous token into account. Then loss is calculated and backpropogated through all the layers till Bio-BERT.

2. **Bias model 1:** We also need a bias model that captures bias. We used a Class prior model as bias model. The class prior probability for each token is calculated with the help of training data. The prior probability vector for a token is calculated by,

$$k_{(n,i)} = \frac{\sum_{m=1}^{N} \sum_{j=1}^{L_m} 1_{\left|x(m,j)=x(n,i) \wedge y^{k}_{(m,j)}=1\right|}}{\sum_{m=1}^{N} \sum_{j=1}^{L_m} 1_{|x(m,j)=x(n,i)|}} \tag{4.1}$$

where N is the number of sentences and L_m is the length of m'th sentence. Basically this formula calculates, for a token with class k, number of times that token appeared with same class (k) dividing by the total number of times that particular token appeared in training set.

3. **Bias model 2 :** We also used another simple bias model to capture bias. This bias model is a simple bilstm model that detects entities. This model has an embedding layer, 2 bilstm layers and a classification layer to generate probability values.

The reported results are Overall precision, recall and F1 scores. Along with that 3 more recall scores of different categories are reported. These 3 categories are made based on presence or absence of entity in training set.

1. **Memorization (Mem)** : Both entity and its Concept Unique Id (CUI) are present in training data.

2. **Synonym Generalization (Syn)** : Only CUI is seen during training and entity of different surface form is unseen.

3. **Concept Generalization (Con)** : Both entity and its CUI are unseen. These entities are called novel entities.

4.1 Baseline methods

We firstly trained a vanilla model without any debiasing and reported its results. Further we reported the results of other baselines with existing debiasing methods.

All the baselines are trained and tested on 3 different datasets.

Model	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
Class Prior (B)	26.9	69.7	38.8	75.6	58.6	62.2
LSTM Model (B)	61.1	48.8	54.2	70.5	11.1	11.3
Vanilla (M)	83.5	89.3	86.3	95.1	78.9	81.1

Table 4.1 Performance of Bias models (B) and Main model(M) on BC5CDR

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	81.5	86.1	83.7	89.8	78.8	82.4
Reweight	76.1	89.3	82.1	91.8	84.3	87.1
Learned MixinH	75.3	84.1	79.4	85.8	80.4	83.0

Table 4.2 Performance of Existing debiasing methods on BC5CDR with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	70.0	86.3	76.9	89.2	80.6	83.4
Reweight	72.2	90.4	80.3	94.6	83.1	84.0
Learned MixinH	40.7	85.1	54.2	87.2	81.0	83.0

Table 4.3 Performance of Existing debiasing methods on BC5CDR with bilstm as bias model

Model	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
Class Prior (B)	12.5	57.4	27.5	68.5	37.2	40.7
LSTM Model (B)	74.1	54.5	62.8	72.5	28.3	19.1
Bio-BERT+CRF (M)	86.3	90.8	88.5	96.3	78.6	83.5

Table 4.4 Performance of Bias models (B) and Main model(M) on NCBI-disease

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	81.5	88.3	84.8	91.3	81.7	85.2
Reweight	77.0	88.4	82.3	91.8	80.6	85.2
Learned MixinH	77.1	85.8	81.2	87.4	78.0	88.9

Table 4.5 Performance of Existing debiasing methods on NCBI-disease with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	73.4	89.9	80.8	94.3	81.2	84.0
Reweight	76.3	89.1	82.2	93.0	82.7	82.7
Learned MixinH	31.8	70.5	43.8	73.7	67.6	62.3

Table 4.6 Performance of Existing debiasing methods on NCBI-disease with bilstm as bias model

Model	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
Class Prior (B)	38.9	50.0	43.8	56.1	35.3	35.7
LSTM Model (B)	55.5	62.8	58.9	67.3	51.8	50.6
Bio-BERT+CRF (M)	64.1	70.6	67.2	72.7	65.7	65.3

Table 4.7 Performance of Bias model (B) and Main model (M) on MedMentions

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	63.0	63.7	63.4	64.4	62.1	61.8
Reweight	62.3	68.2	65.1	70.3	63.1	63.5
Learned MixinH	40.7	47.7	43.9	46.4	50.7	50.4

Table 4.8 Performance of Existing debiaising methods on MedMentions with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	59.4	65.0	62.1	66.7	61.2	60.6
Reweight	61.1	64.2	62.6	65.8	60.4	60.0
Learned MixinH	26.4	24.9	25.6	23.3	28.9	28.5

Table 4.9 Performance of Existing debiaising methods on MedMentions with bilstm as bias model

4.2 Error Analysis

1. **Drop in memory :**

   From the experiments and results of baseline methods, we can see that the model is able to generalize better as the performance of model on synonym and concept generalization increased. Also we see that the performance of memorized entities got reduced. While doing error analysis of what memorized entities we missed after debaising, we found a pattern among those missed entities. Most of the missed entities are biased towards single in training distribution. Some of the examples are listed below in 4.10.

2. **Drop in precision and F1 score:**

   Another good thing to notice in results of existing debiasing methods is the decrease of precision and overall f1 score. Increase of false positives decreases precision and f1 score. While analysing false positives, we found that most

Token	B-tag	I-tag	O-tag
Alzhiemer	6	0	0
thrombotic	19	0	0
glumerenolephritis	10	0	0
pilomatricomas	3	0	0
DCTN4	8	0	0

Table 4.10 Frequency of tags among missed memorized entities

of them are adjectives. Model predicted adjectives because of noise in the dataset. Dataset themselves have adjectives labelled as a part of entities in few instances while not marked as part of entities in other instances. Because of this model gets confused where to predict adjectives as part of entity and where not to. Some of the examples from datasets are given below in table 4.11. Also the evaluation metric used are of exact match where the predicted entity boundary should exactly match with ground truth entity boundary. With this exact match evaluation, even though our model predicted a part of entity still it is considered as missed one. Entities that are actually predicted as subset or superset, but still considered as missed are shown in 4.11

Ground truth entity	Predicted entity
Drug - induced liver injury	liver injury
bleeding	intestinal bleeding
dysmenorrhea	primary dysmenorrhea
severe malaria	malaria
isolated foveal hypoplasia	foveal hypoplasia
unilateral retinoblastoma	isolated unilateral retinoblastoma
patient prognosis	poor patient prognosis

Table 4.11 Ground truth entities and Predicted entities

4.3 Proposed methods

1. **Maintaining recall score of memory :** After applying debiasing to the model we can see that the recall score of memory category drops significantly. So we propose a solution to make the model generalize better by maintaining the recall score of memory category. From the error analysis we made, we found that debiasing is not required of all samples. In simple words, debiasing all the samples makes the model not to learn anything from highly biased samples. This means, even though a particular entity is present in training set, just because our bias model says it is highly biased we tell the main model to not learn anything from it. This makes the model not predict well on those highly biased samples. So to avoid this we don't debias highly biased instance.

Even though a sample is highly biased, let the model learn from it. This brings the balance between model's generalization capability and memorized entities. We define a sample being highly biased by checking if it's confidence on ground truth tag is above 90% . The results below shows that model is able to generalize better by maintaining its performance on memory category.

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE (90%)	82.6	89.4	85.9	93.9	81.1	84.2
Reweight (90%)	83.1	89.9	86.4	94.4	81.6	84.8
Learned Mixin (90%)	82.0	90.2	85.9	94.4	82.1	86.0

Table 4.12 Performance of Proposed debiasing methods on BC5CDR with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE (90%)	82.6	89.4	85.9	93.9	81.1	84.2
Reweight (90%)	83.1	89.9	86.4	94.4	81.6	84.8
Learned Mixin (90%)	82.0	90.2	85.9	94.4	82.1	86.0

Table 4.13 Performance of Proposed debiasing methods on NCBI-disease with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	62.3	70.1	66.0	72.9	63.5	63.4
Reweight(90%)	62.1	70.6	66.1	73.4	63.8	62.8
Learned MixinH(90%)	58.8	67.7	62.9	69.1	64.2	63.6

Table 4.14 Performance of Proposed debiasing methods on MedMentions with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	84.8	89.9	87.3	95.5	80.0	82.6
Reweight(90%)	85.6	88.9	87.2	94.9	78.0	81.4
Learned Mixin(90%)	85.1	89.3	87.1	95.6	77.8	81.1

Table 4.15 Performance of Proposed debiasing methods on BC5CDR with bilstm as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	85.4	89.0	87.2	94.5	79.6	81.2
Reweight(90%)	86.2	92.3	89.2	96.1	84.8	87.0
Learned Mixin(90%)	86.1	90.1	88.1	95.5	81.7	80.2

Table 4.16 Performance of Proposed debiasing methods on NCBI-disease with bilstm as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	62.8	68.4	65.5	70.8	62.7	61.7
Reweight(90%)	63.9	69.1	66.4	71.5	63.4	63.2
Learned MixinH(90%)	62.9	68.1	65.4	70.6	62.0	61.7

Table 4.17 Performance of Proposed debiasing methods on MedMentions with bilstm as bias model

When we used bilstm as a bias model, we found that the performance boils down to the performance of vannila model in most of the cases. The reason is that this bias model has learnt more than what it is expected to learn. This bilstm bias model predicts tag for each token in training set with high confidence ($>= 0.9$). So as in our proposed method we don't debias samples that is highly biased, it becomes we are not applying debiasing. So the performance of model is similar to vannila model.

2. **Relaxed matching metric :**

   As discussed in subsection (4.2), while analysing the missed out entities on all categories, we found that those entities are not actually missing. They are detected with wromg boundaries. So we used relaxed match score to evaluate these models. Along with that as most of the false positives are adjectives, we ignore these false positives during evaluation. This evaluation metric clearly picturises the performance of existing and proposed methods. The relaxed matching scores of all methods experimented are given below.

Model	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
LSTM Model (B)	81.3	67.3	73.6	88.6	44.3	50.0
Class Prior (B)	43.2	84.0	57.0	90.8	75.5	76.4
Bio-BERT+CRF (M)	88.2	92.5	90.3	98.4	84.8	87.6

Table 4.18 Relaxed match Performance of Bias models (B) and Main model(M) on BC5CDR

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	84.9	92.1	88.4	96.0	86.7	89.9
Reweight	80.8	95.1	87.4	98.4	90.7	92.5
Learned MixinH	78.7	93.5	85.5	95.7	90.3	93.0

Table 4.19 Relaxed match Performance of Existing debiasing methods on BC5CDR with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	85.8	94.2	89.8	98.6	88.2	91.2
Reweight (90%)	87.1	93.6	90.2	98.4	87.1	90.2
Learned MixinH(90%)	85.2	94.3	89.5	98.5	88.4	92.0

Table 4.20 Relaxed match Performance of Proposed debiasing methods on BC5CDR with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	72.4	96.0	82.6	99.1	91.7	93.9
Reweight	76.9	95.5	85.2	99.1	90.5	93.0
Learned MixinH	36.8	98.2	53.5	99.6	96.3	97.2

Table 4.21 Relaxed match Performance of Existing debiasing methods on BC5CDR with bilstm as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	88.0	93.0	90.5	98.6	85.8	88.0
Reweight(90%)	89.0	92.4	90.7	98.8	84.2	86.5
Learned MixinH(90%)	89.5	92.2	90.8	98.7	83.3	87.4

Table 4.22 Relaxed match Performance of Proposed debiasing methods on BC5CDR with bilstm as bias model

Model	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
LSTM Model (B)	91.7	75.5	82.8	92.6	67.3	38.4
Class Prior (B)	35.4	82.2	49.4	85.4	73.0	75.1
Bio-BERT+CRF (M)	94.2	95.3	94.8	98.5	91.8	91.7

Table 4.23 Relaxed match Performance of Bias models (B) and Main model(M) on NCBI-disease

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	91.5	95.5	93.5	97.2	92.8	94.7
Reweight	88.4	97.1	92.6	97.0	96.6	93.0
Learned MixinH	88.3	95.9	91.9	96.3	95.1	96.1

Table 4.24 Relaxed match Performance of Existing debiasing methods on NCBI-disease with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	94.0	94.9	94.4	97.2	93.1	91.0
Reweight (90%)	94.1	95.1	94.6	98.7	91.4	90.8
Learned MixinH(90%)	93.9	95.5	94.9	98.4	94.1	91.5

Table 4.25 Relaxed match Performance of Proposed debiasing methods on NCBI-disease with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	84.5	97.2	90.4	98.5	95.8	95.9
Reweight	89.1	96.3	92.6	97.1	95.9	94.9
Learned MixinH	46.2	94.7	62.1	93.2	96.2	96.4

Table 4.26 Relaxed match Performance of Existing debiasing methods on NCBI-disease with bilstm as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	93.7	95.1	94.4	98.9	90.1	91.2
Reweight(90%)	94.4	96.8	95.6	99.4	94.0	93.7
Learned MixinH(90%)	93.9	95.9	94.9	99.1	93.2	91.0

Table 4.27 Relaxed match Performance of Proposed debiasing methods on NCBI-disease with bilstm as bias model

Model	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
LSTM Model (B)	82.9	90.0	86.3	90.3	89.8	89.3
Class Prior (B)	74.1	79.9	76.9	80.3	79.4	79.3
Bio-BERT+CRF (M)	84.8	92.2	88.3	92.0	92.4	92.6

Table 4.28 Relaxed match Performance of Bias models (B) and Main model(M) on MedMentions

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	84.8	87.2	86.0	85.3	89.5	90.1
Reweight	83.9	90.4	87.0	89.3	91.8	92.5
Learned MixinH	67.2	83.2	74.4	80.3	86.9	88.0

Table 4.29 Relaxed match Performance of Existing debiasing methods on MedMentions with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	84.0	91.0	87.4	90.3	91.9	92.4
Reweight(90%)	83.9	91.1	87.3	90.6	91.7	92.0
Learned MixinH(90%)	81.4	90.0	85.5	88.8	91.7	91.7

Table 4.30 Relaxed match Performance of Proposed debiasing methods on MedMentions with class prior as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE	88.5	82.7	85.5	87.2	90.6	90.6
Reweight	84.0	87.1	85.5	85.1	89.5	90.1
Learned MixinH	53.0	64.3	58.1	60.2	69.8	69.8

Table 4.31 Relaxed match Performance of Existing debiasing methods on MedMentions with bilstm as bias model

Method	Overall			Mem	Syn	Con
	Precision	Recall	F1-score	Recall	Recall	Recall
POE(90%)	84.9	88.7	86.7	87.9	89.7	89.7
Reweight(90%)	85.5	89.3	87.3	88.2	90.7	90.9
Learned MixinH(90%)	85.1	89.0	87.0	87.9	90.4	90.3

Table 4.32 Relaxed match Performance of Proposed debiasing methods on MedMentions with bilstm as bias model

Chapter 5

Limitations of current work and Future work

To design a bias model, we need to have knowledge about bias in the dataset. Even-though existing debiasing and proposed debiasing methods shows improvement, still there's a huge gap betweeen performance of memory category and other 2 generalization categories. Currently these models are experimented with highly biased and noisy dataset. Cleaning these noisy datasets and adding more debiased samples can improve the generalization capability of model to a greater extent.

Chapter 6

Conclusion

The main contribution from our thesis is, there doesn't exist much debasing methods available for entity detection task. So we designed model and utilized existing debiasing methods designed for some other tasks in entity detection task. After applying debiasing we found that we are losing out on some memorized entities. So we proposed few changes to existing debiasing methods, so that model generalizes better by maintaining performance on memorized entities. Along with that, we also proposed a relaxed match evaluation metric, that picturises the performance of all model and methods in a better manner as the exact match evaluation metric was too harsh on entity boundaries.

References

[1] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4069–4082, Hong Kong, China. Association for Computational Linguistics.

[2] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2020. Learning to Model and Ignore Dataset Bias with Mixed Capacity Ensembles. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3031–3045, Online. Association for Computational Linguistics.

[3] He He, Sheng Zha, and Haohan Wang. 2019. Unlearn Dataset Bias in Natural Language Inference by Fitting the Residual. In Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019), pages 132–142, Hong Kong, China. Association for Computational Linguistics.

[4] X. Han, S. Wang, C. Su, Q. Huang and Q. Tian, "Greedy Gradient Ensemble for

Robust Visual Question Answering," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 1564-1573, doi: 10.1109/ICCV48922.2021.00161.

[5] Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Towards Debiasing NLU Models from Unknown Biases. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7597–7610, Online. Association for Computational Linguistics.

[6] Nam, Jun Hyun, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee and Jinwoo Shin. "Learning from Failure: De-biasing Classifier from Biased Classifier." Neural Information Processing Systems (2020).

[7] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In Advances in Neural Information Processing Systems, 2018.

[8] Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-End Bias Mitigation by Modelling Biases in Corpora. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8706–8716, Online. Association for Computational Linguistics.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision

[10] Tianyu Liu, Zheng Xin, Baobao Chang, and Zhifang Sui. 2020. HypoNLI: Exploring the Artificial Patterns of Hypothesis-only Bias in Natural Language Inference. In Proceedings of the Twelfth Language Resources and Evaluation Conference, pages 6852–6860, Marseille, France. European Language Resources Association.

[11] Abbas Ghaddar, Phillippe Langlais, Mehdi Rezagholizadeh, and Ahmad Rashid. 2021. End-to-End Self-Debiasing Framework for Robust NLU Training. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1923–1929, Online. Association for Computational Linguistics.

[12] Kim, Eungyeup, Jungsoo Lee, Juyoung Lee, Jihyeon Lee and Jaegul Choo. "Learning Debiased Representation via Disentangled Feature Augmentation." Neural Information Processing Systems (2021).

[13] Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Mind the Trade-off: Debiasing NLU Models without Degrading the In-distribution Performance. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8717–8729, Online. Association for Computational Linguistics.

[14] Kim, Nayeong Hwang, Sehyun Ahn, Sungsoo Park, Jaesik Kwak, Suha. (2022). Learning Debiased Classifier with Biased Committee.

[15] Niu, Yulei Zhang, Hanwang. (2021). Introspective Distillation for Robust Question Answering.

[16] E. Tartaglione, C. A. Barbano and M. Grangetto, "EnD: Entangling and Disentangling deep representations for bias correction," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 13503-13512, doi: 10.1109/CVPR46437.2021.01330.

[17] D. Teney, E. Abbasnejad, S. Lucey and A. Van den Hengel, "Evading the Simplicity Bias: Training a Diverse Set of Models Discovers Solutions with Superior OOD Generalization," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 16740-16751, doi: 10.1109/CVPR52688.2022.01626.

[18] Abbas Ghaddar, Philippe Langlais, Ahmad Rashid, and Mehdi Rezagholizadeh. 2021. Context-aware Adversarial Training for Name Regularity Bias in Named Entity Recognition. Transactions of the Association for Computational Linguistics, 9:586–604.

[19] Niu, Yulei, et al. "Counterfactual vqa: A cause-effect look at language bias." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[20] Shah, Harshay, et al. "The pitfalls of simplicity bias in neural networks." Advances in Neural Information Processing Systems 33 (2020): 9573-9585.