ENS DE LYON

LABORATOIRE HUBERT CURIEN
UMR • CNRS • 5516 • SAINT-ETIENNE

Bias of Few-shots NLP

Master 2 internship report

Author:
Dina El Zein
École Normale Supérieure de Lyon

Supervisor:
Christophe Gravier
Hubert Curien Laboratory,
Université Jean Monnet

Abstract

In few-shot classification for Natural Language Processing (henceforth NLP) tasks, we are interested in learning a classifier to recognize classes which were not seen at training time, provided only a handful of labeled examples in the inference regime. To do so, significant progress in few-shot classification has featured meta-learning, a parameterized model for learning algorithm, that is trained on episodes, each corresponding to a different classification mini-task – each episodes comes with a small labeled support set and query set. Although these NLP models have shown success in classification tasks, they are based on a handful of labeled data so that they are prone to propagate all kind of bias found in language models. In this internship we focus on gender bias found in text corpora. Few-shot NLP models use a pre-trained Transformer neural network to get an embedding representation of text samples that, unless addressed, are prone to learn intrinsic gender-bias in the dataset. We present different approaches that have shown success in quantifying and mitigating gender-bias in the transformer models. From these approaches, we propose different methods to quantify and mitigate bias in meta-learning for NLP tasks.

Contents

1 Introduction

This introduction first aims at giving a brief overview of few-shot learning, especially focusing on gender bias in meta-learning frameworks which is at the core of the internship. We then present a roadmap of our results.

1.1 Motivations

Deep learning models have achieved great success in many fields such as visual recognition tasks [KSH12], [SZ14], [He+16], speech recognition, and natural language processing [Kua+18]. However, these models need a large amount of labeled data which limits their scalability to new classes due to annotation cost. While these approaches can tackle a certain problem with lots of labeled data, these models struggle in tackling problems where labeled data is scarce. Conversely, humans are very good in rapidly learning new classes and recognizing objects with few examples. For example, children who have seen cats and dogs only a few times can quickly tell apart. This gap between human and machine learning casts the question on how to develop deep learning models than can learn from a very limited set of examples, just like human do. There is therefore a growing interest for few-shot learning in deep learning [SS20; Pen+20]. Few-shot learning (henceforth FSL) is dedicated to recognize novel classes from very few labeled examples. The availability of one or few labeled examples conflict with the classical fine-tuning method in deep learning [And+16] since the model must integrate its prior experience with a small amount of information while avoiding over-fitting. Researchers have explored meta-learning to decompose the training into several yet very small similar tasks where knowledge can be learned in a transferable way just as human [FAL17]. In meta-learning, the training procedure is divided into multiple meta tasks where each represents a FSL setting, following the testing procedure matches the training one[1] – that is, the model is evaluated on the entire tested set made of unseen sample and classes, without further updates. The fact that meta learning training is made of very small tasks makes it hard to come up with a model that actually learns yet it is designed to prevent over-fitting. On the contrary, this offers a way for the model to transfer knowledge by switching from one meta task to another one and so few shot models can classify classes with a small support set. Such models would be able to learn from a limited set of training and testing data – we would be able to create models that are *learning to learn*, hence the name *meta-learning*.

While we provide a more formal description of the meta-learning paradigm along with the main existing meta-learning approaches in Section 1.3, we are focusing in this internship on bias in NLP tasks under the meta-learning training strategy. We will first discuss the notion of bias and its ramification with FSL. Bias has been found as an innate human strategy for decision-making. There are numerous types of bias: social, behavioral, ethical, etc. and we will focus on gender bias in this report, or more precisely what is called gender stereotyping [Hoo+20]. Stereotypes are kind of biases that are held among group of people. Gender stereotypes can obstruct gender neutrality in different areas like education, work etc. For example, common gender biases link female terms with liberal arts and family and male terms with science and careers [NBG02]. Bias can be seen in word morphology, i.e. words such as actor are, by default, associated with the dominant class, and female versions of these words i.e. actress, are marked. A bias can be acquired either by an induction process (a limited view over all possible samples or situations) or learned from others (educational or observed). There is a significant bias in social and cultural norms despite work toward gender neutrality.

[1] The training and the testing set will actually be referred as support and query sets respectively in meta-learning frameworks, but it is the same paradigm than the traditional supervised learning setup.

This kind of bias is presented in data we use and thus machine learning models are likely to reproduce and propagate existing gender bias.

These issues can leak down to language models [2] and classification tasks in NLP and produce biased predictions. For instance, remind that all language models are trained using a token masking strategy: a word vector representation depends on its neighbors (the context) in which it occurs [Dev+18; Liu+19]. A model may not associate *female* gender with *doctor* positions because this context does not often occur in the training corpus, then the system is likely to give a higher rank for *male* candidates for *doctor* positions than equally qualified female candidates [BNG20]. NLP applications can reveal gender bias in daily life, and though the study of bias in artificial intelligence is not new, to the best of our knowledge there are few methods to analyze gender bias in few-shot NLP. Thus, gender bias in NLP models have become an area of research [Sun+19]. In our work we are treating gender as binary, though we are aware that this does not include people who identify as non-binary, which can create representational harm [Blo+20].

Figure 1 illustrates the pipeline of few-shot classification task. As shown in the figure, the structure of our framework consists of three main parts: episode sampler, embedding module, and few-shot model classifier. Episode sampler samples episodes randomly in $C-$way $K-$shot task composed of $C \times K$ support instances and several query instances extracted from the training set. For the embedding model [3], we use a pre-trained text encoder, i.e., Bert, and fine-tune it on the dataset we are using for the classification task. The instances of the episode are first fed into the embedding model, which generates representation vectors for each instance. Then the encoded support and query vectors are forwarded to the few-shot model. Few-shot models aim to classify the query instances according to the representations of support instances. The loss is calculated across the query points and then back propagation process is scheduled. Using this pipeline we obtain a classifier which can quickly adapt to new few-shot classification tasks. During testing, we use both the embedding model and the few-shot model to make predictions on the test episodes sampled from a disjoint test set.

The studies on gender bias in NLP are mainly divided into two categories: bias in the embedding spaces, and bias in downstream tasks [Blo+20], which is meta-learning text classification in our case. Given the pipeline of our few-shot classification framework in Figure 1, we can therefore encounter bias in two main parts: embedding model and few-shot model. We present different techniques that we use to quantify and mitigate bias at the level of the text encoder −embedding model (see Section 3), and at the level of meta-learning itself (see Section 4). The current work contributes to the advancements of fairness in NLP by investigating methods for quantifying and mitigating gender bias in few-shot meta-learning networks.

Research Questions (RQ). Framing our research in questions, we would like to answer the following: how can we quantify gender bias in few-shots meta-learning NLP models (RQ1), how do different meta-learning NLP models compare in terms of gender bias (RQ2), and how can we mitigate gender bias in few-shots meta-learning NLP models (RQ3).

[2] The language model's task is to model the distribution of sentences in a given corpus [Lu+18]. Given a sentence prefix, the model computes the likelihood for every word indicating how (un)likely it is to follow the prefix in its text distribution.

[3] Embedding model is a language model tool that transforms syntactic elements (words) into real vectors capturing syntactic and semantic relationships among words.

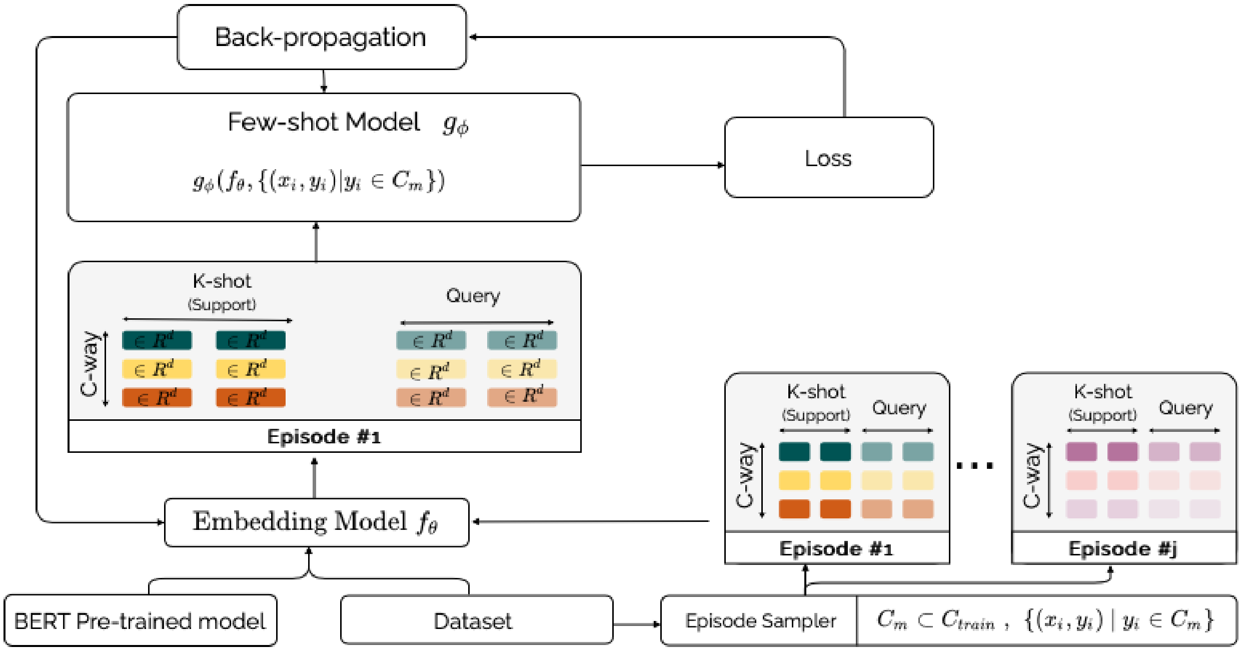

Figure 1 – Few-shot classification pipeline illustrated on a 3-way 2-shots text classification meta-training task $(C = 3, K = 2)$.

1.2 Few-shot Learning

Few-shot learning is a type of machine learning where the model aims to classify between classes from only a handful of sample representatives. FSL is different from the classical machine learning models where the model recognize the data in the training set and then generalize on the testing set given it is fed as much data as it needs to give better performance. FSL is devoted to solve the data deficiency problem by recognizing novel classes from very few labeled examples. Few-shot classification is a task in which a classifier must be adapted to accommodate new classes unseen in training, given only a few examples of each of these classes.

Consider the problem of few-shot classification task τ that contains a supporting set S and a query set Q. The supporting set, $S = \{(x_1, y_1), ..., (x_s, y_s)\}$, consists of labeled samples and the query set, $Q = \{(x_1, x_2, ...x_t)\}$ contains unlabeled samples different than the ones in S. If the support set, S, contains K labeled examples for each of the C unique classes, the target few-shot problem is called a C-way K-shot problem. The supporting and querying sets have the same label space. The goal is to use the support set to correctly classify the querying set containing a different set of samples sampled the same C classes. One-shot learning is an extreme case of FSL, which utilizes only a single support sample from each class $(k = 1)$.

With the support set only, typically one can train a classifier to assign a class label to each sample in the query set using the classical learning strategy [4]. However, due to the lack of labelled samples in the support set, the performance of such a classifier is usually not satisfactory. Classifier models cannot learn the exact distribution of the data due to the variance of the supporting set, and they tend to overfit to the small supporting data set because of the high bias. To solve the limitation of data availability, one can perform meta-

[4] The classical learning strategy refers to the process of building algorithms that can learn from existing observations, and leverage that learning to predict new observations, or determine the output of new input.

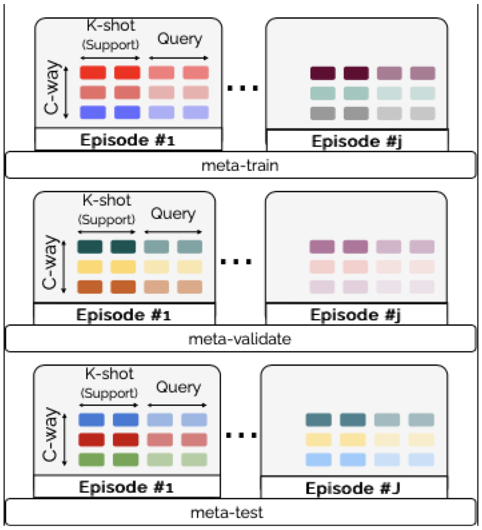

Figure 2 – Meta-learning framework

learning on the training set in order to extract some transferrable knowledge that allow to perform better on the support set and thus classify the query set more successfully without suffering from the overfitting that might be expected when applying deep models to sparse data problems.

1.3 Meta-Learning

Meta-learning is a popular method for FSL that does not learn a single task – such as learning a particular concept – but it learns many tasks that have similar concept. In meta-learning, the model is trained on a variety of different tasks but nevertheless having many similarities to achieve the ability of quick adaptation to new tasks without using many data. The learner aims to leverage common points across these tasks not only to become better at solving a task but to solve future tasks rapidly and efficiently. Meta-learning's main challenge is to learn from prior experience in a systematic way, first, by collecting data that describes the learning task, second, by learning from this meta-data to extract and transfer knowledge that guides the search for optimal models for new tasks. In different words, meta-learning covers learning based on prior experience with other tasks and so when a new task represents a new phenomena, experience will not be effective.

1.3.1 Meta-Learning Methods

All meta learning frameworks are three-stage process as follows: meta-training, meta-validation and meta-testing (each associated with a corresponding dataset: D_{train}, $D_{validate}$, and D_{test} respectively). Each of this three phase contains a certain number of few-shot learning tasks $\{\tau\}, \forall\tau \in \{\tau\}, \tau \ \sim P(\tau)$ as described above. Unlike the standard supervised learning setting, the classes for each phase are strictly non-overlapping. The D_{train} trains the meta-learner to generate good task metrics for FSL tasks and evaluate its performance on the D_{test}. $D_{validate}$ is used to select good hyper-parameters. The meta-learning framework for FSL classification is illustrated in figure 2.

In order to best describe the different meta-learning methods, first consider the few-shot learning task $\tau = \{S, Q, \mathcal{L}\}$ where S and Q are the support and query sets respectively containing pairs (x, y) and $\mathcal{L}$ is a loss function. The meta-model q_ϕ, a parameterized function q with parameter ϕ, is trained on a distribution over $P(\tau)$.

Meta-learning algorithms core are designed to be two-fold:

(i) Learner step: a classifier $q_\theta(\cdot)$ trained for operating a given task on the support set S

(ii) Optimization step: choose how to optimize the meta-parameter θ with respect to max-likelihood objective using query points: update θ using $\nabla_\theta \mathcal{L}(q_\theta(\cdot), Q)$

These steps can be adapted according to the different approaches of meta-learning which can be divided in three categories: model-based, metric-based, and optimization-based.

Model-Based Methods, are called Black-Box methods, depends on architecture design to utilize neural network to improve model generalization capability (adaptation) [Pen20].

Metric-Based Methods, [MBL20] aims to learn a metric or distance function κ over objects. It consists of designing an embedding function that can capture dis/similarity between data points. To construct such measure, most metric learning algorithms optimize a loss function that aims at bringing similar samples of the same label together while pushing apart ones of different labels like in nearest neighbors algorithms. Learning a good embedding is crucial for the success of a metric-based meta-learning model. Following a metric based approach, the two steps(i) and (ii) become:

(i) compute $\hat{y} = \sum_{(x,y)\in S} q_\theta(\kappa(\hat{x}, x)y)$ where $\hat{x} \in Q$

(ii) update θ by $\nabla_\theta \mathcal{L}(\hat{y}, Q)$

Optimization-Based Methods, learning model is optimized using gradient descent algorithm which takes many iterative steps to perform well because it starts from a random initialization of its parameters which results in late convergence. Learning common initialization parameters that can be used across multiple tasks saves the many iterative steps. The main idea behind the optimization gradient-based algorithms is to discover a model whose parmeters can be adapted with just a few steps of gradient descent. The representative algorithm of this family: Model Agnostic Meta Learning algorithm (MAML) [FAL17] does so. MAML initializes the model's parameters by training on different data-set to achieve optimal fast learning with less number of gradient steps. By doing so, the model can use already initialized parameters by fine tuning the architecture through one or more gradient steps. Following an optimization approach, the two steps become:

(i) $\theta^* = \theta - \alpha \nabla_\theta \mathcal{L}(q_\theta, S)$ where θ is a prior meta model parameter initialization

(ii) $\theta = \theta - \alpha \nabla_\theta \mathcal{L}(q_{\theta^*}, Q)$

Optimization based methods can be learned with a limited number of both labeled examples and gradient update steps [FAL17]. These methods are capable of achieving rapid adaptation with a limited number of supporting examples for different classes but they have difficulty in handling domain shifts between classes [FAL17]. Metric based methods are based on a similarity measure between samples. They address the meta-learning problem by *learning to compare* intuition [KZS+15]. Specifically, if a model can determine the similarity between two samples, it can classify the the unlabeled samples with labeled instances easily.

[Che+19] and [DGL21] have shown, using their results, that model-based classifiers achieve competitive performance in comparison to optimization based ones and thus they are the best when equipped with the latest encoder techniques. Therefore, in our work, we focus on diverse metric-based FSL networks like Matching, Prototypical, and Relation networks introduced in Section 2 with one optimization-based network called Induction networks.

1.3.2 Training for Meta-Learning

Training procedures should be carefully selected to match test time inferences. An effective way to use the training set is to decompose the training process into auxiliary meta-learning stages, and imitate the few-shot learning setting via episode training as proposed by [Vin+16]. For this, each training iteration is split in very small batches called episodes in order to compute gradients and update the model. Any episode is formed randomly by selecting C classes from the training set with K labeled samples from each of the classes to act as a support set $S = \{(x_i, y_i)\}_{i=1}^{|S|}$ where $|S| = C \times K$. As the episode classes C are sampled randomly and the training set is built, a subset of the remaining samples from the episode classes C is chosen to serve as the query set Q s.t. $Q = \{(x_j, y_j)\}_{j=1}^{n}$. The meta-training process learns to learn from S to minimise the loss over Q. This strategy is called episode-based meta-training and the details are shown in Algorithm 1. During meta-testing, we check if the model can be adapted to never-seen-before classes given only a few examples of each of these new classes. In a nutshell, the support/query set split during training is designed to simulate the support/query set that will be encountered at test time – during meta-testing. The model that is trained from support/query set at train time can be further fine-tuned using the support set that will be encountered at test time.

Algorithm 1 Episode-Based Meta Training

```
for each episode iteration do
    (i) Randomly select C classes from the class space of the training set;
    (ii) Randomly select $K$ labeled samples from each of the $C$ classes as support set $S = \{(x_i, y_i)\}_{i=1}^{K \times C}$,
and select a fraction of the reminder of those $C$ classes' samples as query set $Q = \{(x_i, y_i)\}_{i=1}^{n}$;
    (iii) Feed the support set $S$ to the model and update the parameters by minimizing the loss in the query
set $Q$;
end for
```

2 Preliminaries

In this section, we present few-shot classification methods evaluated in this work: Matching Network [Vin+16], Prototypical Network [SSZ17] and its variants, Relation Network [Sun+18], and Induction Network [Gen+19]. We begin by formally introducing the task of supervised FSL classification.

2.1 Notations

We remind the following notations for a *C-way-K-shot* classification task τ that contains a supporting set S and a query set Q. S consists of K labeled samples for each of the C classes (i.e. $|S| = K \times C$) and Q contains different samples from the same C classes. A model is given $S = (x_1, y_1), ..., (x_s, y_s)$ with a goal to classify the query set $Q = (x_1, y_1), ..., (x_q, y_q)$. Formulating this classification task in the meta-learning framework, we have different meta-sets: meta-train, meta-valid, and meta-test (D_{train}, D_{valid}, and D_{test} respectively) sampled from the task distribution $p(\tau)$ such that $D_{train} \cap D_{valid} \cap D_{test} = \phi$. Meta-learning algorithms are trained and tested by constructing consecutive episodes from D_{train} and D_{test} respectively. Let C_{train} be the set of all classes available for training. We note C_{valid} and C_{test}, the set of classes used for validating and testing respectively, with $C_{train} \cap C_{valid} \cap C_{test} = \phi$. Each episode consists of *C-way-K-shot* task. The query set in each episode serves as episode-scale optimization where the model parameters are updated based on the prediction loss on Q, given S as an input.

2.2 Few-Shot Classification Methods

2.2.1 Matching Network

Matching Networks (henceforth MN) introduced by [Vin+16] consists of a neural network embedding function and an augmented memory. The embedding function, f_θ, with learnable parameter θ maps an input x to a d-length vector, i.e., $f_\theta : x \rightarrow \mathbb{R}^d$. The augmented memory stores the support set $S = \{(x_i, y_i)\}_{i=1}^{|S|}$ where x_i is a sample and y_i is the corresponding label. MN defines a classifier conditioned on S that relies on a similarity measure between query point and the support set. For any new data $\hat{x}$, the MN predicts its label $\hat{y}$ using a similarity function $\alpha(\cdot)$ between $\hat{x}$ and S.

$$\hat{y} = P(.|\hat{x}, S; \theta) = \sum_{i=1}^{|S|} \alpha(\hat{x}, x_i; \theta) y_i \tag{1}$$

Note that Equation 1 describes the output class for the point $\hat{x}$ as a linear combination of the labels in the support set, $\{y_i\}_{i=1}^{|S|}$. Thus, the predicted output class of the point $\hat{x}$ becomes $argmax_y P(.|\hat{x}, S, \theta)$, the one with the highest similarity.

Specifically, $\alpha(,)$, the similarity function is defined to be a softmax distribution given some kind of distance between the test instance $\hat{x}$ and the supporting instance x_i

$$\alpha(\hat{x}, x_i; \theta) = \frac{exp[d(f_\theta(\hat{x}), f_\theta(x_i))]}{\sum_{j=1}^{|S|} exp[d(f_\theta(\hat{x}), f_\theta(x_j))]} \tag{2}$$

where θ is the parameter of the embedding function f_θ and $d : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, +\infty)$ is a distance function. The embedding function f_θ is parameterized as deep convolutional neural networks for the tasks and the distance function is defined as the cosine distance (as in the seminal paper) or the euclidean distance.

To train Matching Networks, the training procedure has to be chosen carefully as the model has to perform well with support and query sets that contain classes never seen during training (i.e., meta-test). For the training procedure (i.e., meta-train), FSL tasks are sampled from D_{train} with support set S and query set Q that are used to compute gradients and update the model. The objective function to optimize the embedding function parameter, θ, is to minimize the prediction error of the query samples given the support set as follows:

$$\mathop{\mathbb{E}}_{\mathcal{T} \sim p(\mathcal{T})} \left[\mathop{\mathbb{E}}_{S,Q \sim \mathcal{T}} \left[\sum_{(x,y) \in Q} log(P(y|x, S; \theta)) \right] \right] \tag{3}$$

The parameter of the embedding function, θ, is optimized via stochastic gradient descent methods.

Training with Equation 3 yields a model that works well when sampling a task from different distribution of labels, i.e., tasks $\tau \in D_{test}$, without fine tuning on the classes that have never seen due to its non-parametric nature. However, if we test on a task τ' that diverges very far from the tasks which were sampled to learn θ (i.e. not sampled from $p(\tau)$), the model will not work due to domain drift.

2.2.2 Prototypical Network

Prototypical Networks (henceforth PN) were introduced by [SSZ17] as an extension of Matching Networks. They solve the problem of FSL by addressing the key issue of overfitting. Because data is severely limited in few-shot classification tasks, they work under the assumption that a classifier should have a very simple inductive bias. The main idea behind their approach is that there exists an embedding in which points cluster around a single prototype representation for each class. For this, they use a neural network embedding function, f_θ, parametrized by a trainable parameter θ to learn a non-linear mapping of the input point to the embedding space. More specifically, support and query samples are mapped into a feature space through the non-linear mapping function $f_\theta : \mathbb{R} \to \mathbb{R}^d$. PN compute a class representation $c_i \in \mathbb{R}^d$, or prototype, of each class $c_i \in C_{train}$through f_θ. Each prototype is the mean of the support samples belonging to its class in the embedding space:

$$c_i = \frac{1}{|S_i|} \sum_{(x_j, y_j \in S_i)} f_\theta(x_j) \tag{4}$$

where S_i denotes the set of examples labeled with class i in the given supporting set and $\sum f_\theta(x_j)$ is a component-wise sum for all $(x_j, y_j) \in S_i$.

The classification task for test points is performed by finding the nearest class prototype. Specifically, given a query point $\hat{x} \in Q$ and a distance function, $d : \mathbb{R}^d \times \mathbb{R}^d \to [0, +\infty)$ the model computes a similarity, using a similarity function $\alpha(,)$, between the embedding of the query point and each of the prototypes. A softmax over the distances produces a probability distribution P over the classes seen in the support set:

$$\hat{y} = P(.|\hat{x}, S; \theta) = \sum_{i=1}^{C} \alpha(\hat{x}, c_i; \theta) y_i \tag{5}$$

$\hat{y}$ in Equation 5 is a distribution over the different classes thus the predicted class of the point $\hat{x}$ is $argmax_y P(.|\hat{x}, S, \theta)$, the one for which the distance is smallest.

$\alpha(,)$, the similarity function, is defined as follows:

$$\alpha(\hat{x}, c_i; \theta) = \frac{exp[-d(f_\theta(\hat{x}), c_i))]}{\sum_{i'=1}^{C} exp[-d(f_\theta(\hat{x}), c_{i'})]} \tag{6}$$

d, the distance function, is the Euclidean distance as in the PN's paper [SSZ17]. As any distance metric is permissible, we add the cosine similarity-based distance in our experiments in order to measure the impact of selecting another distance metric.

The model is meta-trained by minimizing the negative log probability $-log\, p(y = i|\hat{x}; \theta)$ of the correct class assignment i for all query points via stochastic gradient descent.

At test time, episodes are created using classes from C_{test}, and accuracy is measured as the query points assignments, given prototypes derived from the support points.

[Ren+18] proposed an extension to PN where they use unlabeled data points, U, with support and query points. They augment the original PN [SSZ17] by proposing extensions to produce refined prototypes using unlabeled data starting from the basic definition of prototypes. After computing each class's prototype, a soft k-means technique is applied to further refine those prototypes using unlabeled data points. Each unlabeled sample, $x_u \in U$, is given a partial ('soft') assignment $\alpha(x_u, c_i; \theta)$ to each cluster (of each class c_i) based on their distance to the centroid locations. The refined prototypes, $\tilde{c_i}$, are as follows:

$$\tilde{c_i} = \frac{\sum_{(x_j, y_j \in S_i)} f_\theta(x_j) + \sum_{(x_u \in U)} f_\theta(x_u)\alpha(x_u, c_i; \theta)}{|S_i| + \sum_{(x_u \in U)} \alpha(x_u, c_i; \theta)} \tag{7}$$

and $\alpha(x_u, c_i; \theta)$ is calculated using Equation 6.

This additional step in the refined process aims at correcting the support points selection bias and making the method more robust. After obtaining the refined prototypes, they train the model the same way as ordinary PN.

2.2.3 Relation Network

Relation Networks (henceforth RN) introduced by [Sun+18] challenges the idea of using a pre-defined metric. They augment the original PN [SSZ17] and replace the distance measure, d, with a relation module. Thus, RN consist of two modules: the embedding module, f_θ, and the relation module, g_ϕ, both parametrized by trainable parameters θ and ϕ respectively. $f_\theta : \mathbb{R} \rightarrow \mathbb{R}^d$ produces a feature map of each of the support data $x_i \in S$ and the query point $\hat{x} \in Q$. The feature maps $f_\theta(x_i)$ of the support points are used to calculate prototypes c_i for each class the same way in PN. Then relation module models the relationship between the embedding of the query point $f_\theta(\hat{x})$ and each class prototype c_i yielding a similarity score $\alpha(\hat{x}, c_i, \theta, \phi)$ in range 0 to 1, which is called relation score. Thus, for the C classes setting, it generates C relation scores for the relation between one query input $\hat{x}$ and prototypes. Instead of using a pre-defined distance metric like the euclidean or the cosine one, this approach allows such networks to learn this metric by themselves.

Specifically, given a query point $\hat{x} \in Q$, the classification task proceeds as follows:

$$\hat{y} = P(.|\hat{x}, S; \theta, \phi) = \sum_{i=1}^{C} \alpha(\hat{x}, c_i; \theta, \phi) y_i \tag{8}$$

where c_i, the prototype for class i, is calculated using Equation 4 .

Similar to other networks, $\hat{y}$ in Equation 8 is a a linear combination over the different classes. Thus, the predicted output class of the point $\hat{x}$ becomes $argmax_y P(.|\hat{x}, S; \theta, \phi)$, the one with the highest relation score.

To calculate the relation score, $\alpha(\hat{x}, c_i; \theta; \phi)$, the feature map $f_\theta(\hat{x})$ and the prototypes are fed into the relation module to produce relation scores between a class's prototype c_i and the target query point $\hat{x}$.

$$\alpha(\hat{x}, c_i; \theta, \phi) = g_\phi(f_\theta(\hat{x}), c_i) \tag{9}$$

To train the model, mean square error (MSE) loss is used regressing the relation score $\alpha(\hat{x}, c_i; \theta; \phi)$ to the ground truth: matched class where the relation is closet has similarity 1 and 0 otherwise. Thus, learning proceeds by minimizing the loss function:

$$\theta, \phi \leftarrow \underset{\theta, \phi}{\operatorname{argmin}} \sum_{i=1}^{C} \sum_{(x,y) \in T} (\alpha(x, c_i; \theta, \phi) - \mathbb{1}(y == c_i))^2 \tag{10}$$

There are two different relation module g_ϕ architectures:

base. The base relation module combines the feature map $f_\theta(\hat{x})$ and the prototypes with operator $C(f_\theta(\hat{x}), c_i)$, a concatenation function that concatenates the query point with each prototype. Then it applies a small feed-forward neural network on the concatenation of both $f_\theta(\hat{x})$ and c_i composed of two linear layers, with a ReLU activation function, $F(.)$, in between. Following, Equation 9 becomes:

$$\alpha(\hat{x}, c_i; \theta, \phi) = \langle w, M_2(F(M_1(C(f_\theta(\hat{x}), c_i))))\rangle \tag{11}$$

where w, $M1$, $M2$ are learnable parameters.

NTL. The Neural Tensor Layer relation module introduced by introduced by [Soc+13] uses intermediate learnable matrices $M_t \in \mathbb{R}^{d\times d}$ to model the relation between support points and prototypes. Thus Equation 9 becomes:

$$\alpha(\hat{x}, c_i; \theta, \phi) = \langle w, F(f_\theta(\hat{x})^T M_t c_i)\rangle \quad , \quad w \in \mathbb{R}^h \quad , \quad t \in [1:h] \tag{12}$$

where w is a learnable parameter and $t \in [1 : h]$ is one slice of the tensor parameters and F is a non-linear activation function called ReLU [GBB11].

2.2.4 Induction Network

Induction Networks (henceforth IN) introduced by [Gen+19] aims at finding a non-linear mapping between the support points and the class representatives (prototypes). The main motivation for this network is that calculating the class prototypes by averaging the vectors of its data points – what is done in PN and RN – is too restrictive. IN consist of three modules: embedding module, induction module, and relation module. Similar to other networks, the embedding module, $f_\theta : \mathbb{R} \rightarrow \mathbb{R}^d$ with learnable parameter θ, maps an input point to a feature vector in the embedding space. Induction module extracts the representation of each class with a goal of having a non-linear mapping from the embedding of the sample point to the class prototype c_i:

$$\{f_\theta(x) \in \mathbb{R}^d\}_{(x,y)\in S} \rightarrow \{c_i \in \mathbb{R}^d\}_{i=1}^C \tag{13}$$

For this, they apply a dynamic routing algorithm [SFH17] on the class embedded support points. To handle the support set at any scale – in case of $K > 1$ –, a weight-sharable transformation $W_s \in \mathbb{R}^{d\times d}$ and a bias $b_s \in \mathbb{R}^d$ are employed across all sample vectors in the embedded support set. Each sample embedded vector $f_\theta(x)$ becomes a sample prediction vector $\widehat{f_\theta(x)}$:

$$\widehat{f_\theta(x)} = squash(W_s f_\theta(x) + b_s) \tag{14}$$

where $squash$ is a non-linear squashing function that keeps the direction of the vector unchanged but decreases its magnitude. More formally, given the input vector x, $squash$ is defined as:

$$squash(x) = \frac{\|x\|^2}{1+\|x\|^2}\frac{x}{\|x\|} \tag{15}$$

Each class prototype is the weighted sum of all the sample prediction vectors belong to the class i:

$$\widehat{c_i} = \sum_{(x,y)\in S_i} \widehat{f_\theta(x)} \tag{16}$$

where S_i is the set of support samples for class i.

Then a non-linear 'squashing' function is applied to $\widehat{c_i}$ to ensure that the length of the prototype vector doesn't exceed 1:

$$c_i = squash(\widehat{c_i}) \tag{17}$$

Now, the class prototypes are calculated, the relation module takes as an input a query point $\hat{x} \in Q$ and the prototypes of the classes to perform the classification task which is similar to equation 8 in RN. The output class label for the query point $\hat{x}$ is the one with highest correlation.

The relation module, g_ϕ, parametrized by a learnable parameter ϕ, measures the correlation between a query point $\hat{x}$ and each of the class prototypes to give a relation score which is a scalar between 0 and 1 similar to RN. Specifically, they use the Neural Tensor Layer (NTL) relation module: this is the same as the one introduced earlier in Section 2.2.3, as it has shown great results in modeling the relationship between two vectors [HKW11; Gen+17].

Similar to RN, IN uses the mean square error (MSE) loss to train the model with the same loss function as in Equation 10

In our work, we are applying these few-shot classification networks to text samples, when doing so, two main system components are considered: the text feature extractor and the downstream part of the neural network. The text feature extractor – embedding functions – are mapping raw-data into low dimensional embeddings. Thus text feature extractors have become one of the fundamental tools for extensive tasks in natural language processing [She+19; Che+21]. The downstream part of the neural networks – learning part of the networks – provides a learning strategy over few shots. In order to compare these systems against gender bias – second question in the research questions in Section 1.1 –, they should be plugged with the same feature extractor because changing the feature extractor can lead to significant performance changes and thus it is hard to conclude if these changes are due to the improvements of the few-shot learning techniques or to significant advances made by text feature extractors [DGL21]. These large-scale neural text extractors are pre-trained on massive corpora and they are prone to bias. Thus they play a large role in gender bias in all aforementioned few-shot networks. Therefore, We will explore the relevant sota for gender bias in text encoders.

3 Text Encoders and Bias

Our few-shot classification problem motivated the use of embedding functions – or what is called text encoders. Since text encoders are optimized to capture the statistical properties of training data, they tend to pick up on and amplify stereotypes and bias present in the data. In this section, we study the methods used in quantifying and mitigating bias in encoders.

3.1 Text Encoders

Text Encoders are defined by finding a model that maps a sequence of text data into fixed-length vector representation. This vector encodes the meaning of the sentence and thus can be used in downstream tasks in natural language processing. The encoders are low-dimensional; they learn continuous vector representations of discrete variables. With the development of deep learning, large scale neural sentence encoders such as BERT ([Dev+18]), Bi-LSTM ([HXY15]), ELMo ([Pet+18]), and GPT ([Rad+18]) have shown desirable performance on many NLP tasks. Since BERT is the state-of-the-art in many applications and has been used in recent few-shot learning models, we employ it for sentence encoding.

3.2 BERT

BERT, Bidirectional Encoder Representations from Transformers, is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [Dev+18]. There are two main steps in BERT framework: pre-training and fine-tuning. For the pre-training phase, the model is trained on unlabeled data, i.e., *BooksCorpus* (800 M words) [Zhu+15] and *English Wikipedia* (2,500 M words), over different tasks using masked language model (henceforth MLM). MLM is to mask some percentage of the input tokens at random, and then predict those masked tokens. For the other phase, fine-tuning, BERT is initialized with the pre-trained parameters, and then all the parameters are fine-tuned using labeled data from the downstream task (with access only to text samples, without the labels/classes). The pre-trained BERT model can be fine-tuned with just one additional output layer on several tasks.

Although BERT have been studied comprehensively, the fairness and bias do exist in the pre-trained encoder. The fairness problem is recognized as social bias denoting the behavior of the model for socially sensitive topics, such as gender, race, and religion [Lia+20]. Resources supporting the pre-training of BERT have severe gender imbalance. Though this bias is mainly caused by the unbalanced data of the corpora used in pre-training phase [Bol+16], retrain the large scale deep model sentence encoder, BERT, requires a massive amount of computational resources [Dev+18].

3.3 Gender Bias and stereotype in English

Gender bias is an unequal systematic treatment on the basis of gender [Sun+19]. In the context with our work, gender bias occurs when a certain gender is more closely with a profession than another resulting in biased language models. A biased representation in language is related either to the employment situation in real world or stereotypes in the society. The latter case leads representational harm because actual participation in the work world is invisible [Blo+20]. Adding to this, if word representations are used in downstream systems for hiring decisions, gender bias, irrespective of whether it is representative of real-world data may lead to allocational harm because female and male are not equally associated with a profession [Blo+20].

3.4 BERT Bias Quantification

Focusing on BERT, we measure gender bias by studying associations between gender denoting target words and names of professions [5]. Following [Kur+19] and [BNG20], we measure the influence of the attribute (A), which can be *occupation*, on the likelihood of the target (T), which denotes a *male* or *female* person: $P(T|A)$. In BERT language model, the likelihood of a token is influenced by all other tokens in the sentence. In BERT it is assumed that the likelihood of the target is different depending if the attribute is present or not: $P(T) \neq P(T|A)$. Also, the likelihood of male and female denoting the target are influenced differently by the attribute: $P(T_{female}|A) \neq P(T_{male}|A)$. Thus we need to measure the association between the target word and the attribute. Since BERT is trained using masked language modeling - to predict masked tokens, denoted [MASK], in a sentence given the entire context - we use the prediction over the [MASK] word to measure bias. For this, we need to get the likelihood of the masked target from the BERT language model in

[5] This paragraph is very similar to Section 2 [Kur+19] and Section 4.4 in [BNG20] where this idea is taken from

two different settings: with the attribute masked, call it prior probability (p_{prior}), and not masked, call it target probability (p_T). For example, to measure the association between the target female gender and the attribute *professor*, we feed in the masked sentence '[MASK] *is a professor*' to BERT, and compute the probability for the [MASK] word to be *she*: p_T. This measurement is based on how much BERT prefers the *female* gender with the attribute *professor* compared to the *male* gender. The prior probability of the model towards predicting the *female* gender is measured by: masking the attribute word *professor* and then feeding BERT with the sentence '[MASK] *is a* [MASK]'. This prior probability indicates the likelihood of the word *she* in BERT, given only the structure of the sentence with no other evidence about the attribute. Finally, the difference between the normalized predictions for the words *he* and *she* can be used to measure the gender bias in BERT for the *professor* attribute. In general, to compute the association between a target and an attribute, the procedure is as follows:

(i) Prepare a template sentence with a target and attribute words i.e., '[TARGET] is a [ATTRIBUTE]'

(ii) Mask the target word, i.e., '[MASK] is a [ATTRIBUTE]'

(iii) Compute the probability of the target word in the sentence $p_T = P(\text{[MASK]} = \text{[TARGET]}|\text{sentence})$

(iv) Mask both [TARGET] and [ATTRIBUTE], i.e., '[MASK] is a [MASK]'

(v) Compute the prior probability $p_{prior} = P(\text{[MASK]} = \text{[TARGET]}/\text{masked sentence})$

(vi) Compute the association as by dividing the target probability by the prior and take the natural logarithm $log\frac{p_T}{p_{prior}}$

If we get a negative values for the association between a target and an attribute, it means that the probability of the target is lower than the prior probability, i.e. the probability of the target decreased through the combination with the attribute. A positive association value means that the probability of the target increased through the combination with the attribute, with respect to the prior probability.

3.5 BERT Bias Mitigation

There are different approaches to mitigate gender bias in BERT, either through changing the resulting vector representation (working on the embedding module) or modifying the training data. [GG19] has shown that one of the most effective strategies for removing bias in traditional word embeddings involves modifying the training data rather than changing the resulting vector representation. One such strategy is **Counterfactual Data Substitution** [Mau+19] the derivative of **Counterfactual Data Augmentation** [Lu+18] in which the gendered words in the training corpus are swapped in place in order to counterbalance bias.

3.5.1 Counterfactual Data Augmentation

Following [Lu+18], Counterfactual Data Augmentation (henceforth CDA) is defined as follows:

Definition 3.1
(CDA). Given an intervention c, the dataset D of input instances (X, Y) can be c-augmented, D/c, to produce the dataset $D \cup \{(c(x), y)\}_{(x,y)\in D}$.

Where interventions are transformations of instances to their matches under a particular concept such as gender (in our case). In CDA, text transformation designed to invert bias is performed on a text corpus, the result of which is then appended to the original, to form a new bias-mitigated corpus used for training embeddings. For example, *'the man cleaned the kitchen'* becomes *'the woman cleaned the kitchen'* as *man-woman* is on list (gender swapped intervention list). Both versions would then together be used in embedding training, in effect neutralising the *man–woman* bias. The augmented sentences generated need to remain semantically and grammatically sound. Thus grammar intervention uses coreference information to swap gender words when they co-refer to a proper noun. Such rules avoid the case: *'Elizabeth ... she ... queen'* being changed to, for instance, *'Elizabeth ... he ... king'*. As a result, CDA does not flip gendered words if they are in a cluster with a proper noun. It also uses part-of-speech (POS) tag [6] information to avoid ambiguity between personal pronoun and possessive determiner. For example, both *his* and *him* would be flipped to *her*, while *her* should be flipped to *him* if it is an objective pronoun and to *his* if it is a possessive pronoun. In this context, *'her teacher was proud of her'*, becomes *'his teacher was proud of him'*.

In this context, interventions do not change the ground truth of the data, on the contrary, it highlights the core feature of the method. Thus an unbiased model should not distinguish between matched pair.

3.5.2 Counterfactual Data Substitution

CDA results in a duplication of the text but for both genders. It produces debiased corpora with peculiar statistical properties unlike those of naturally occurring text – notably all observed word frequencies will be even. As such, [Mau+19] proposed CDS an extention to CDA, mainly differs in applying 'substitutions probabilistically' (with a a certain probability), which results in a non-duplicated counterfactual training corpus. These substitutions are performed on the original corpus to maintain grammatically and discourse coherence. This simple change should have advantages in terms of naturalness of text and processing efficiency, as well as theoretical foundation. For name intervention, CDA approach does not swap proper names to the opposite gender and so does not swap the gender of words which co-refer with proper nouns. Consider the sentence *'Elizabeth ... she ... smart'*, since *she* and *Elizabeth* co-refer, the counterfactual corpus copy will not replace *she* with *he* in this instance, and as the method involves a duplication of text, this would result in a stronger, not weaker, association between *she* and gender-stereotypic concepts [Mau+19]. Thus, CDA reinforce certain biases. [Mau+19], in their extensive approach, proposed an explicit treatment of first names only. This rule adds the case: *'Elizabeth ... she ... smart'* being changed to, for instance, *James ... he ... smart.* They fixed associate pairs of names for swapping, as such, they expanded [Lu+18]'s list of gender pairs vastly.

To mitigate gender bias in BERT, we use CDS to swap in place the gender words in the training corpus in order to counterbalance bias.

4 Bias in Meta-Learning

The core of our work is to check upon gender bias in meta-learning – mainly answering questions 1 and 3 in research questions in Section 1.1. We highlight two sources of gender bias that can cause few-shot models to fail fair results: text corpus and episode sampler. Following, we propose some strategies to quantify and mitigate the bias.

[6] POS tag is a special label assigned to each token (word) in a text corpus to indicate its part of speech

4.1 Few-Shot NLP Bias Detection

To detect gender bias in few-shot classification networks, we present three approaches: Odds Ratio, Stereotypical Gender Association Test and Stereotypical Downstream Performance Discrepancies. Odds Ratio is used to check upon bias in the text corpus while the other two are used to check upon bias in the few-shot model.

4.1.1 Odds Ratio

As presented in [SP21], we use the Odds Ratio (OR) to measure gender bias in the training corpora. From the copora, we get two dictionaries C_f and C_m that have the class labels presented in our data as keys and their corresponding occurring frequencies $|c|$ as values: $C_f = \{c_1 : |c_1^f|, c_2 : |c_2^f|, ...c_n : |c_n^f|\}$ and $C_m = \{c_1 : |c_1^m|, c_2 : |c_2^m|, ...c_n : |c_n^m|\}$ mapping class labels to their frequency for female and male respectively. We use OR [Szu10] to find the occupations with larger frequency differences for males and females which indicates that they might potentially manifest gender biases in the corpora. For a class label c_j, the odds ratio is calculated as the odds of having it in the C_m divided by the odds of having it in the C_f:

$$\frac{C^m(c_j^m)}{\sum_{i,(c_i \neq c_j^m)}^n C^m(c_i^m)} \Big/ \frac{C^f(c_j^f)}{\sum_{i,(c_i^f \neq c_j^f)}^n C^f(c_i^f)} \tag{18}$$

The larger the OR is, the more likely the class label, occupation in our study, will occur in male than female sections. After obtaining a list of occupations and their corresponding OR, we sort them in descending order. The top k occupations are more likely to appear for males and the last k ones for females.

This method measures the percentage of bias presented in the data set rather than behavior of the model toward a biased data set.

4.1.2 Stereotypical Gender Association Test

Inspired by the idea of measuring bias in BERT presented in Section 3.4 by [Kur+19]. Stereotypical gender association test is defined by the ability of the model to predict a certain occupation for a sample text independent of the gender used in the text. This measure revels the association between genders, *male* and *female*, and the class label we are predicting i.e., *occupation*, call it target class. Consider a classification setting with features $X \in \mathbb{R}^n$, target class T, and sensitive attributes $A = \{female, male\}$. Our goal is to have a model that outputs predictions fair with respect to the sensitive attributes. This statistical parity can be formalized as :

$$P(T|X\ and\ A_{female}) \quad vs\ P(T|X\ and\ A_{male}) \tag{19}$$

Note that the sensitive attribute is not used directly by the classification model, we are just using it to compare results. Generally, the procedure is as follows:

(i) Pick a text sample – call it sentence – from the corpora

(ii) Check the probability of the class label for the sentence $p_{ori} = P(T/sentence)$

(iii) Augment the sentence by applying gender-swapping technique

(iv) Compute the probability $p_{gender_swapped} = P(T/\text{gender-swapped sentence})$

Original Sample	→ His main research interests are in the areas of ...
Pro-Stereotyoical	→ His main research interests are in the areas of ...
Anti-Stereotyoical	→ Her main research interests are in the areas of ...

Table 1 – Text sample from the *CommonCrawl* data set under label job Professor which is a male dominant job

(v) Compute the association by taking the natural logarithm function of the probabilities divided $log\frac{p_{ori}}{p_{gender-swapped}}$

For interpretation, we have three possible directions: negative, positive, or around zero associations. A negative association means that there is a weak association of *female (male)* sample text with statistically *male (female)* professions, which should be strengthened through bias mitigation techniques. On the other hand, a positive association value means that there is a strong relationship between *female (male)* person with statistically *female (male)* profession which should be reduced through mitigation bias techniques. In these two cases, if the association is significantly greater than or lower than zero, it means that the professions are not gender-balanced – the direction of bias or the inequity of genders changed with respect to the tested original sentence at hand. Lastly, located around zero association means that both association scores of *female* and *male* have approximately the same value, thus there is no difference between the associations of *female* and *male* person with statistically gender-balanced professions; the association score should not deviate much from its original value with mitigation bias techniques.

We use this approach while testing a few-shot classification network to check if the model used is biased for a specific gender direction.

4.1.3 Stereotypical Downstream Performance Discrepancies.

This approach is taken from the WinoBias data set presented in [Zha+18a]. To better identify gender bias, we build two data sets from our original corpora: ‘pro-stereotype’ and ‘anti-stereotype’ data sets. Pro-stereotype data set represents the over represented stereotypes in the original corpora, samples for occupations stereotypically associated with the gender of the pronoun, and the anti-stereotype data set represents under represented stereotypes, ones that require linking to anti-stereotypical occupations. For example, when woman dominates a profession, we call linking the noun phrase referring to the job with female and male pronoun as ‘pro-stereotypical’, and ‘anti-stereotypical’, respectively. Similarly, if the occupation is male dominated, linking the noun phrase with the male and female pronoun is called, ‘pro-stereotypical’ and ‘anti-stereotypical’, respectively (see prime example in table 1). To evaluate the classification model and check if it is prone to bias, we measure the accuracy on both data sets made. We say that a model is ϵ-robust to bias if it passes both pro-stereotyped and anti-stereotyped data sets with their respective accuracy difference at most ϵ. That margin, ϵ, is to be as small as possible.

Choosing suitable metrics to analyse gender bias is a challenge [Blo+21]. While a positive result in relation to a suitable metric indicates the presence of bias, a negative result does not imply that a model is completely bias-free. To measure gender bias, we use the two different metrics presented in [Man+21]: stereotype metric and skew metric. Stereotype metric is used to measure a model’s tendency to assign a gender to professions by calculating: $F1^{g}_{pro} - F1^{g}_{anti}$, the difference in $F1$ scores with respect to gender g, with positive (resp. negative) values indicating a pro - (resp. anti-) stereotypical assignment. Skew metric uses the difference in $F1$ scores with respect to a dataset D, across genders $F1^{\text{♂}}_{D} - F1^{\text{♀}}_{D}$ with positive (resp. negative) values capturing the tendency of a model to generally assign a male (resp. female) gender to any given profession.

Both these forms of gender unfairness are considered in the subsequent analysis using the mean skew and stereotype, taken across datasets and genders respectively as shown below:

$$\mu_{skew} = \frac{1}{2}(\mid F1_{pro}^{♂} - F1_{pro}^{♀} \mid + \mid F1_{anti}^{♂} - F1_{anti}^{♀} \mid) \tag{20}$$

$$\mu_{stero} = \frac{1}{2}(\mid F1_{pro}^{♂} - F1_{anti}^{♂} \mid + \mid F1_{pro}^{♀} - F1_{anti}^{♀} \mid) \tag{21}$$

where $F1_{pro}^{♂}$ denotes the $F1$ score on the pro-stereotypical dataset with male samples only.

To do so, we take the 'pro-stereotype' and 'anti-stereotype' data sets we constructed and divide each a part between male and female samples.

4.2 Few-Shot NLP Bias Mitigation

There are several approaches for debiasing gender bias in few-shot NLP by working on two tangents: text corpora and prediction algorithms. The first approach discuses **data augmentation** techniques to debias text corpora while the other one debias gender bias in NLP by adjusting predictions in NLP systems through taking the optimization function of an existing network and constrains that function to ensure its predictions fit defined conditions [Sun+19]. In text encoders, it has been shown that removing bias in traditional word embeddings involving modifying the training data is more effective than trying to change the resulting vector representation [BNG20]. To this end, we discuss the text copora tangent and highlight some challenges that may still faced by this approach in a few-shot setting. To adapt these challenges, we present two approaches: **neutralization** and **calibration**. The neutralization approach is a hybrid one that solves the challenge at the text copora level by taking benefits of both gender swapping and annoymization techniques. The calibration approach resolves the issue through a new tangent: episode sampling.

4.2.1 Data Augmentation

To remove gender bias in few-shot NLP using data augmentation, we construct an additional training corpus where all male entities are swapped for female entities and vice-versa. Methods can then be trained on both original and swapped corpora. For this, we follow the data augmentation with sensitive attributes: we first describe the sentence data augmentation process. Our gender bias topic consists of two potential bias directions: *female* and *male*. For a sensitive attribute word w, assume that we can always find another sensitive attribute word u such that w and u have equivalent semantic meaning but different bias direction. Then we can use u a replaceable word of w in the opposite direction. In our gender bias topic, the word *boy* is in the bias direction $male$, a replaceable word for *boy* in $female$ direction is $girl$. With this method, we can generate augmented sentence x' for each sentence x we consider.

Specifically, for a sentence x, we first need to find the sensitive words in the bias direction then find replacable words for them in the other direction. With this approach, we are making an assumption that the classification model's bias of a certain bias direction is only made by the sensitive words. Table 2 provides an example for sentence augmentation.

For detecting and swapping sensitive words in a sentence, we define a gender intervention bidirectional dictionary of 252 gendered pairs that include pronouns, nouns, adjectives, etc. such as *he: she*, *her: him/his*, *madam: mister*, *actor: actress*, *queen: king* following the CDA approach presented in Section 3.5. The complete list of the gendered words is a combination of previous gender lists [Zha+18a; Mau+19; Zha+18b] defined

	Bias direction	Sensitive Attribute words	Text Context
Original	female	she, her	{She} received {her} BSc degree
Augmentation	male	he, his	{He} received {his} BSc degree

Table 2 – Examples of generating an augmentation sentence under the sensitive topic 'gender'.

by adding to it some gender words that were used in our corpus (i.e., *Mr: Miss/Mrs*), to cover most of the gendered words presented in the data. We replace every occurrence of a gendered word in the original data with its dual (if it is in the defined gender intervention scope).

For swapping gender proper names, we define a dictionary name intervention consisting of $8,065$ *female* and $5,413$ *male* proper names by getting names for both genders from [SP21; Mau+19]. Opposite to CDA and similar to CDS approaches in Section 3.5, we are swapping gender names presented in corpora but at random. With CDS, gender swapping of proper names is only done on the first name using swapped fixed pairs. In our work, we are swapping gender names at random, for instance, *'James'* can be inverted to *'Elizabeth'* or any other female name in the list. We do it randomly in order to prevent the model to mostly learn predictions from a limited set of statistical clues – such as names in this case – which a tendancy for the text encoder [NK19]. We use the *Spacy* library to check if a word refers to a proper name included in our name intervention, and if it does, we swap it with a random proper name with the opposite gender.

After getting the augmentation of the sentences, we encode both sentences, the original one and the augmented one with the pre-trained text encoder, BERT in our case, to get the embedded sentences. The embeddings will have opposite bias directions since the sentences have the same meaning but different potential bias directions. This approach will get both the text encoder to have data for the different bias directions and thus it would be neutral. The embeddings are both used to train the few-shot model we are using to solve a classification task. Thus, the model will have access to opposite bias directions for the same label classes. This approach will result in completely balanced dataset – having the same number of samples per female and male for each class label – and thus the few-shot model we are using for classification will have access to the different bias directions and thus it would be neutral.

4.2.2 Neutralization

In the few-shot classification models, we train our models by episode sampling where only few samples for each class are sampled at random. Thus, even if the copora is gender balanced, it may happen that we sample data from only one gender per class. We present a method to get a gender free corpora. We adopt a simple rule to the gender swapping approach (described in Section 4.2.1). Instead of flipping gendered words to the opposite gender, we flip them to a neutral word. Common rules include *'she/he → they'*, *'his/her → their'*, *'Mr./Mrs. → Mx.'*. For the corpora to be gender neutral, proper names should be either flipped to a gender free proper name or anonymized words. To anonymize named entities, we use an automatic named entity finder [Lam+16]. Named entities are replaced consistently within document (i.e., *'Barak Obama ... Obama was re-elected.'* would be annoymized to *'E1 E2 ... E2 was re-elected.'*). Then we build a dictionary of gendered terms with their neutral equivalence. We have not find any neutral lists for gendered word online, therefore we start building one. Rules added to this list are validated by online dictionaries, i.e., Oxford one. After getting a neutral corpora, we fine-tune BERT on it and then use the neutral embeddings to train few shot classification models, this reveals gender bias that may occur during episode sampling. We have an assumption that this approach can have some drawbacks. BERT, the encoder, is pre-trained on large corpora that may have bias in. Even if we fine-tune it on the gender free corpora, the bias presented in the pre-trained BERT can leak into the embeddings of the neutral samples and thus affect the classification models.

4.2.3 Calibration

The difference of gender frequencies in the samples used for training the model in the episode might come from the meta-learning sample bias. To offset the potential confounding that could be brought by the episode sampling strategy and present a fair gender sampling approach per a class label, we propose the following calibration strategy: Instead of sampling random support points for a specific class, we take an equal number of male and female supports per class. In this fashion, the classification model will have access to a fair gendered support set regardless of the gender bias present in the data – unequal number of support samples for both genders. Thus the model is expected to behave neutrally.

5 Experiments

In this section, we describe the datasets used in our evaluation framework then we report our results.

5.1 Datasets

We address and evaluate the performance of our approaches on detecting and mitigating bias for few-shot NLP models in the meta-learning framework on text job classification datasets in English.

CommonCrawl[7]. The CommonCrawl dataset is composed of job descriptions as well as genders from 28 job categories. Each text sample belongs to one of the 28 occupations. This data has been famously used to train OpenAI's GPT-3 model. The data is therefore representative of what can be found on the English speaking part of the Internet, and thus contains a certain amount of bias. The number of text samples in this dataset is 216943 after dropping the duplicates (they were 217197). It is highly imbalanced: the most common class *(professor)* holds 69, 944 samples with 31399 people reported as female and 38545 as male while the least common one *(rapprt)* has 783 samples with 719 male and 64 female. Table 3 provides occupations statistic reported as female.

WikipediaGenderEvents[8]. This dataset consists of career and personal life descriptions for celebrities from 8 job categories taken from Wikipedia events. The number of text samples in this dataset is 555. Table 4 provides occupations statistic reported as female.

5.2 Evaluation Setup

To test the presented methodologies for detecting and mitigating bias in few-shot meta-learning, we used the few-shot classification framework implemented by [DGL21]. This framework implements the different end-to-end neural architectures for few-shot text classification presented in Section 2 using the same transformer-based feature extractor BERT in Section 3.1.

[7] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.646.4837&rep=rep1&type=pdf

[8] https://github.com/PlusLabNLP/ee-wiki-bias/blob/master/data/final_manual.csv

Occupation	%	Occupation	%
rapper	8.2	physician	39.5
surgeon	13.5	professor	44.9
dj	15	personal trainer	45
software engineer	15.1	painter	46.1
composer	16.3	journalist	49.9
comedian	21	poet	50.2
architect	22.5	teacher	59.3
pastor	24.1	psychologist	61.8
chiropractor	27.9	interior designer	80.9
filmmaker	33.8	model	82.6
dentist	34.8	paralegal	84.2
photographer	35	yoga teacher	85
accountant	36.2	nurse	91.1
attorney	37.7	dietitian	92.6

Table 3 – Occupations statistics used in *CommonCrawl* dataset, organized by the percent of people in the occupation who are reported as female.

To get the embedding functions, we use the pre-trained transformer BERT and fine-tune it on the datasets with block size 256 and number of training epoch 30. We average the results over the 30 epochs for evaluation. The test set is fixed for all experiments.

To run the few-shot classification networks, we set the few-shot evaluation setup presented in Section 2 as follows: we will denote the new classes unseen during training, C_{test}. During training, C_{train}, is the set of classes for training the model, the approaches assume that during training, a task-significant set of classes noted C_{train} is available, along with an accordingly task-significant number of labeled data for each class. C_{valid} is the set of classes used for validation. Each set of classes C_{train}, C_{valid}, and C_{test} contain labels from both female and male biased jobs. We create training episodes by first sampling C classes uniformly at random from C_{train}, we then sample K support examples and T test examples from each of these classes. The model is then iteratively trained using both query and support points. At validating and testing time, the same sampling strategy is made with classes among C_{valid} and C_{test} respectively, with $C_{test} \cup C_{valid} \cup C_{train} = \phi$. The model is then evaluated on its ability to predict labels for the Q query samples, using the K support samples. We use the the same values for C, Q, and K at testing, validating, and training time. Since we are hecking for few shots, we use $K = Q = 2, 6, 10, 16$ with C ranging from 2 to 5 in all our experiments. We are only considering even values for K in order to apply the Calibration approach we suggested in Section 4.2.3, where we have to choose equal number of samples for each gender per class. The number of episodes we used for training, validating, and testing is: 10000, 600, and 600 respectively. The results are averaged among different episodes and the test set is fixed among different experiments. .

Occupation	%	Occupation	%
writers	50.0	artists	52.1
models	53.9	musicians	43.7
podcasters	44.4	dancers	47.1
chefs	47.8	comedians	48.5
composer	16.3	comedians	49.9

Table 4 – Occupations statistics used in *WikipediaGenderEvents* dataset, organized by the percent of people in the occupation who are reported as female.

5.3 Results and Discussions

Unfortunately, the clearance process to access the lab cluster to run our implemented experiments is tedious for students outside Université Jean Monnet – we received access in the beginning of July. The code needs significant computational resources, especially GPUs, and can therefore hardly be ran on a personal computer –Neural networks especially transformers are usually fine-tuned on GPUs because they highly benefit from parallel computation. Until we received access to the lab GPU cluster, we tried to run the experiments using the $GRID5000$ cluster[9]. We did not succeed in running them because the code takes few days to fine-tune yet we were not allowed to run experiments for more than 160 core-hour per day on a machine. When we got access on the lab's clusters, we started running experiments directly and we were hopeful that the end of July and the month of August would allow us to run all our experiments. Unfortunately, the lab cluster went down in the beginning of August. The issue is still not fixed at the time of writing, as the engineer responsible for the cluster was not available during this holiday season and no recovery plan in such case is prepared. Consequently, we tried in the meantime to get access to the *IDRIS-Jean Zay* cluster[10] from the CNRS. Three weeks after submitting our application, we did not receive any answer yet. It is therefore understandable that we may not present in this report all the experiments we plan to run, but we will nonetheless give an insight on what has be run and collected before the lab cluster unexpected unavailability.

Figure 3 shows the loss for fine-tunning the language model $BERT$ on the original version of the $CommonCrowl$ dataset. We can see that the loss is decreasing for both the training and evaluation part –no overfitting, so we are still learning further. For other versions of the dataset described in Section 4.1.3 and Section 4.2 ('pro-stereotype', 'anti-stereotype', 'neutral', and 'gender-balanced'), similarly the loss is still decreasing.

6 Conclusion

Bias in NLP systems has the potential to not only mimic but also amplify stereotypes in society. Quantifying gender bias in few-shots NLP is challenging since few-shots models are closely linked to text encoders. Embedding models inherit implicit gender bias from the corpus trained on when there is unequally entities for male and female. This learned bias transfers to downstream tasks, such as few-shot models. We discuss state-of-the-art approaches for detecting and mitigating gender bias in text encoders, mainly BERT. From these approaches, we inspire different ways for detecting gender bias in few-shot NLP, either using the text

[9] `https://www.grid5000.fr/w/Grid5000:Home`

[10] `http://www.idris.fr/jean-zay/`

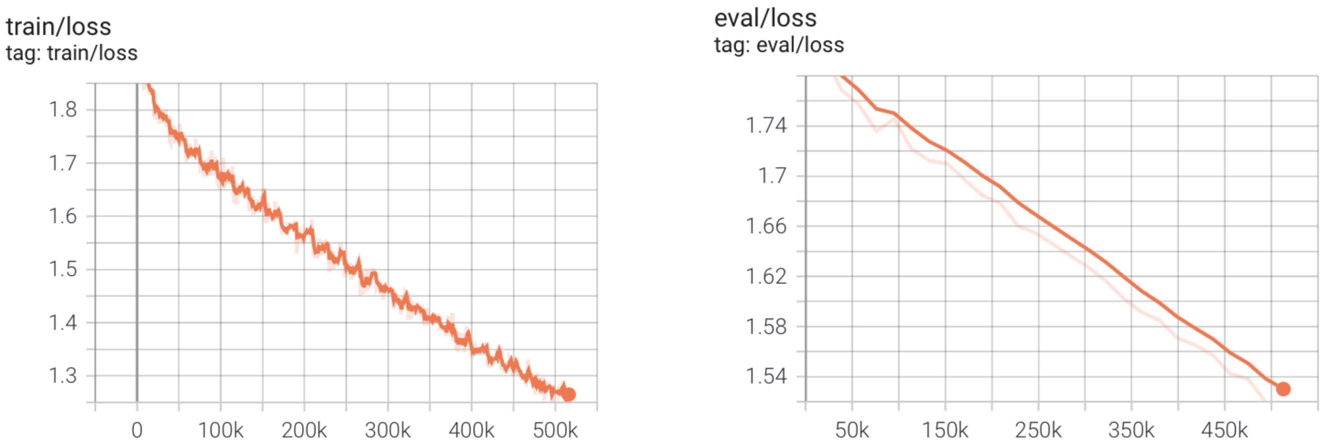

Figure 3 – Loss on the original CommonCrawl dataset for fine-tuning the BERT encoder with 30 training epochs. We can see that the loss (y-axis) is decreasing for both the training part (right figure) and the evaluation part (left figure), the x-axis shows the number of steps it went to so far –each epoch is m steps of batch size 8, $m =$ number of samples$/8$)

.

corpora or the predictive algorithm. We also explore bias mitigation techniques that makes few-shot NLP models more robust to gender bias cues. To completely mitigate bias effectively, it is important to understand how machine learning methods encode biases and how humans perceive biases.

Naturally, there are a number of limitations of this work. We are mainly working on detecting and debiasing gender bias in NLP assuming that the protected attribute being discriminated against is binary. Non-binary genders [Ric+16] are largely ignored in NLP and should be considered in future work.

7 Acknowledgement

I would like to thank my advisor and Thomas Dopierre for valuable discussions.

References

[And+16] Ark Anderson et al. “Beyond fine tuning: A modular approach to learning on small data”. In: *arXiv preprint arXiv:1611.01714* (2016).

[BNG20] Marion Bartl, Malvina Nissim, and Albert Gatt. “Unmasking Contextual Stereotypes: Measuring and Mitigating BERT’s Gender Bias”. In: *arXiv preprint arXiv:2010.14534* (2020).

[Blo+20] Su Lin Blodgett et al. “Language (technology) is power: A critical survey of" bias" in nlp”. In: *arXiv preprint arXiv:2005.14050* (2020).

[Blo+21] Su Lin Blodgett et al. “Stereotyping Norwegian salmon: an inventory of pitfalls in fairness benchmark datasets”. In: *Proc. 59th Annual Meeting of the Association for Computational Linguistics.* 2021.

[Bol+16] Tolga Bolukbasi et al. “Man is to computer programmer as woman is to homemaker? debiasing word embeddings”. In: *Advances in neural information processing systems* 29 (2016), pp. 4349–4357.

[Che+19] Wei-Yu Chen et al. “A closer look at few-shot classification”. In: *arXiv preprint arXiv:1904.04232* (2019).

[Che+21] Pengyu Cheng et al. “FairFil: Contrastive Neural Debiasing Method for Pretrained Text Encoders”. In: *arXiv preprint arXiv:2103.06413* (2021).

[Dev+18] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[DGL21] Thomas Dopierre, Christophe Gravier, and Wilfried Logerais. “A neural few-shot text classification reality check”. In: *arXiv preprint arXiv:2101.12073* (2021).

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[Gen+17] Ruiying Geng et al. “Implicit discourse relation identification based on tree structure neural network”. In: *2017 International Conference on Asian Language Processing (IALP).* IEEE. 2017, pp. 334–337.

[Gen+19] Ruiying Geng et al. “Induction networks for few-shot text classification”. In: *arXiv preprint arXiv:1902.1048* (2019).

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[GG19] Hila Gonen and Yoav Goldberg. “Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them”. In: *arXiv preprint arXiv:1903.03862* (2019).

[He+16] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[HKW11] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. “Transforming auto-encoders”. In: *International conference on artificial neural networks.* Springer. 2011, pp. 44–51.

[Hoo+20] Sara Hooker et al. “Characterising bias in compressed models”. In: *arXiv preprint arXiv:2010.03058* (2020).

[HXY15] Zhiheng Huang, Wei Xu, and Kai Yu. “Bidirectional LSTM-CRF models for sequence tagging”. In: *arXiv preprint arXiv:1508.01991* (2015).

[KZS+15] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. “Siamese neural networks for one-shot image recognition”. In: *ICML deep learning workshop*. Vol. 2. Lille. 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[Kua+18] Shaohui Kuang et al. “Attention Focusing for Neural Machine Translation by Bridging Source and Target Embeddings”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. July 2018.

[Kur+19] Keita Kurita et al. “Measuring bias in contextualized word representations”. In: *arXiv preprint arXiv:1906.07337* (2019).

[Lam+16] Guillaume Lample et al. “Neural architectures for named entity recognition”. In: *arXiv preprint arXiv:1603.01360* (2016).

[Lia+20] Paul Pu Liang et al. “Towards debiasing sentence representations”. In: *arXiv preprint arXiv:2007.08100* (2020).

[Liu+19] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[Lu+18] Kaiji Lu et al. “Gender Bias in Neural Natural Language Processing”. In: *arXiv e-prints* (2018), arXiv–1807.

[Man+21] Daniel de Vassimon Manela et al. “Stereotype and Skew: Quantifying Gender Bias in Pre-trained and Fine-tuned Language Models”. In: *arXiv preprint arXiv:2101.09688* (2021).

[Mau+19] Rowan Hall Maudslay et al. “It’s All in the Name: Mitigating Gender Bias with Name-Based Counterfactual Data Substitution”. In: *arXiv preprint arXiv:1909.00871* (2019).

[MBL20] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. “A metric learning reality check”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 681–699.

[NK19] Timothy Niven and Hung-Yu Kao. “Probing neural network comprehension of natural language arguments”. In: *arXiv preprint arXiv:1907.07355* (2019).

[NBG02] Brian A Nosek, Mahzarin R Banaji, and Anthony G Greenwald. “Harvesting implicit group attitudes and beliefs from a demonstration web site.” In: *Group Dynamics: Theory, Research, and Practice* 6.1 (2002), p. 101.

[Pen+20] Baolin Peng et al. “Few-shot natural language generation for task-oriented dialog”. In: *arXiv preprint arXiv:2002.12328* (2020).

[Pen20] Huimin Peng. “A comprehensive overview and survey of recent advances in meta-learning”. In: *arXiv preprint arXiv:2004.11149* (2020).

[Pet+18] Matthew E Peters et al. “Deep contextualized word representations”. In: *arXiv preprint arXiv:1802.05365* (2018).

[Rad+18] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[Ren+18] Mengye Ren et al. "Meta-learning for semi-supervised few-shot classification". In: *arXiv preprint arXiv:1803.00676* (2018).

[Ric+16] Christina Richards et al. "Non-binary or genderqueer genders". In: *International Review of Psychiatry* 28.1 (2016), pp. 95–102.

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules". In: *arXiv preprint arXiv:1710.09829* (2017).

[SS20] Timo Schick and Hinrich Schütze. "Exploiting cloze questions for few shot text classification and natural language inference". In: *arXiv preprint arXiv:2001.07676* (2020).

[She+19] Dinghan Shen et al. "Learning compressed sentence representations for on-device text processing". In: *arXiv preprint arXiv:1906.08340* (2019).

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[SSZ17] Jake Snell, Kevin Swersky, and Richard S Zemel. "Prototypical networks for few-shot learning". In: *arXiv preprint arXiv:1703.05175* (2017).

[Soc+13] Richard Socher et al. "Reasoning with neural tensor networks for knowledge base completion". In: *Advances in neural information processing systems*. 2013, pp. 926–934.

[SP21] Jiao Sun and Nanyun Peng. "Men Are Elected, Women Are Married: Events Gender Bias on Wikipedia". In: *arXiv preprint arXiv:2106.01601* (2021).

[Sun+19] Tony Sun et al. "Mitigating gender bias in natural language processing: Literature review". In: *arXiv preprint arXiv:1906.08976* (2019).

[Sun+18] Flood Sung et al. "Learning to compare: Relation network for few-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1199–1208.

[Szu10] Magdalena Szumilas. "Explaining odds ratios". In: *Journal of the Canadian academy of child and adolescent psychiatry* 19.3 (2010), p. 227.

[Vin+16] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Advances in neural information processing systems* 29 (2016), pp. 3630–3638.

[Zha+18a] Jieyu Zhao et al. "Gender bias in coreference resolution: Evaluation and debiasing methods". In: *arXiv preprint arXiv:1804.06876* (2018).

[Zha+18b] Jieyu Zhao et al. "Learning gender-neutral word embeddings". In: *arXiv preprint arXiv:1809.01496* (2018).

[Zhu+15] Yukun Zhu et al. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.