Fairness based Job Classification

Multi label Text Classification

MLDM-2 Project Report

Sri Kalidindi
University Jean Monnet
shanmugha.sri.siva.kalidindi@univ-st-etienne.fr

Guillaume Sacchetti
University Jean Monnet
guillaume.sacchetti@etu.univ-st-etienne.fr

Shamprikta Mehreen
University Jean Monnet
shamprikta.mehreen@etu.univ-st-etienne.fr

Oluchi Ibeneme
University Jean Monnet
oluchi.ibeneme@etu.univ-st-etienne.fr

Abstract

*The problem of text classification is one of the widely studied and used techniques. It is used in machine learning, data mining and information retrieval in the text domain. Text Classification is being used in various domains like news, medical, document organization and Job markets. We will be focusing on one of the domain which is **Job Classification** using the description of a person. In addition to classical text classification, this problem also tackles **Fairness**. Fairness here implies, gender neutral based job classification. This problem is posted as one of the collaborative student competition in kaggle called DEFI-IA-2021.*

1. Introduction

DEFI-IA-2021 is a NLP competition, where the ML algorithm has to assign one of the 28 job categories to a given job description. The goals of this task is two fold as follows, the machine learning algorithm has to be as accurate in classifying the job, in addition to this it also needs to be fair in terms of gender. An ideal gender based fairness algorithm should classify with out considering any gender based features.

1.1. Problem Description - Fairness

Text Classification is well defined and well known problem in various domains, It simply involves assigning a label to a text depending on the features used. We will focus more on defining fairness in this problem context. Modern machine learning problems suffer from fairness in various forms, some of the examples where machine learning imposes bias in various fields are, bank loan algorithms issuing loans based on color or gender of the applicant, hiring algorithm preferring people based on ethnicity or marital status, Insurance algorithms preferring a certain group of people on sensitive attributes.
More formally an algorithm is considered unfair if the outcomes or predictions Y are based on some sensitive attributes S, which ideally should not be considered in the decision process. Some of the characteristics of these attributes S are.

- S divides the data or observations into sensitive subgroups(e.g: black/white, male/female)
- The machine learning algorithm should not show different behavior on these divided subsets.

A machine learning algorithm is considered as unfair if one of the previous characteristics is met. In our problem, we need the classification algorithm to be fair in terms of gender. Wherein, the predictions should in no form be dependent on gender based attributes.

1.2. Data Description

DEFI-IA-2021 contains 217,197 train job description samples and 54,300 of test job descriptions. This data has been extracted using Common Crawl[1], thus this data set represents English part of the data that is found in the internet, and also thus contains significant amount of bias. Table: 1 shows the 28 labels in the data set. As observed, this

Job Category	Number of Samples	Class Number	Job Category	Number of Samples	Class Number
pastor	1497	0	nurse	12622	14
model	4115	1	poet	4292	15
yoga_teacher	944	2	dentist	5450	16
teacher	9145	3	chiropractor	1406	17
personal_trainer	807	4	filmmaker	4124	18
painter	4621	5	professor	70016	19
journalist	12295	6	photographer	14646	20
interior_designer	858	7	rapper	783	21
surgeon	6616	8	psychologist	10391	22
accountant	3121	9	paralegal	967	23
dj	831	10	architect	5841	24
physician	11607	11	composer	3395	25
comedian	1639	12	attorney	18820	26
software_engineer	4060	13	dietitian	2288	27

Table 1. Labels

data set is highly imbalanced. Seven categories(**journalist, physician, nurse, professor, photographer, psychologist and attorney**) hold 75 percent(150,397) of the training data with just **Professor** holding 30 Percent of the data(70,016). In addition to the Job Labels, Gender Labels are provided for each of the each job description.

2. Existing Techniques and State-of-the-art

TF-IDF: Term Frequency - Inverse Dense Frequency is a technique which involves building vector for every document using Term Frequency and Inverse Dense Frequency values by using the formulas in eq:1. This vector is then used with other machine learning algorithms.

$$
\begin{aligned}
TF &= \frac{\text{Number of repetitions of word in a document}}{\text{Number of words in a document}} \\
IDF &= log(\frac{\text{Number of documents}}{\text{Number of documents containing the word}})
\end{aligned}
\tag{1}
$$

Other than TF-IDF, one of the common and proven vectorization technique in NLP is word embedding.

Word Embeddings is a vector representation which allows the words with similar meaning to have a similar representation. Using word embeddings has been considered as a breakthrough in deep learning in the field of natural language processing problems. Most of the neural networks do not perform good in a high-dimensional and sparse vector space, The major advantage of the dense representation from word embedding is generalization power[2](page 92). Word embedding are a class of techniques where each word is assigned with a vector in a predefined space, these vectors are usually learned in the similar way that resembles Neural Network. Some of the famous word embedding in NLP are Word2Vec[8] and GloVe[9].

The vectors obtained by either TF-IDF or Word Embedding can be used to train both simple and powerful machine learning algorithms. Some of the classical techniques include using Naive Bayes, SVM, linear regression and logistic regression. Some powerful Deep learning methods include LSTM(Long Short Term Memory), RNN(Recurrent Neural Network).

Benchmarks like AGNews, DBpedia, 20News, Amazon and Yelp are used to compare performance of different techniques under text classification. In a wide variety of tasks some versions of BERT performed better than the other techniques. This version of BERT[4] is considered as the current State-of-the-art[5] for text classification. BERT has given state-of-the-art results not only in text classification but on variety of NLP tasks like question answering, NER etc., BERT(Bidirectional Encoder Representations for Transformers) has been trained on wikipedia and BooksCorpus. BERT is trained for masked language modeling and next sentence prediction.

3. Pre-Processing and Data Augmentation

Pre-Processing: We performed the standard Pre-Processing best practices like Noise removal(removing special characters), lower-casing(converting text to lowercase), stop word removal(removing stop words) and lemmatization(converting word to root word).

Data Augmentation: Since we are dealing with highly imbalanced scenario, one of the solution available is using data augmentation. We used three techniques namely Back Translation, NLP Albumentation and Easy Augmentation.

In Back Translation, we use language translation to augment the data. Samples from the minority class are randomly translate it to a language (E $\rightarrow$ L) and translated back from the language to english (L $\rightarrow$ E) and this new sample is added to the training set. In NLP Albumentation, we randomly replace a word with it's synonym and this sample is considered as a new sentence for that minority class. Easy Augmentation involves random deletion, random swap and random replacement of words on the original sample and this new sample is again added to the training data.

4. Metrics

F1-macro Macro-averaged F1 score is widely used to evaluate the quality of a machine learning model in an highly imbalanced multi class setting. F1-Macro is calculated independently for each class and then weighted by the support (number of samples of that class) as shown in the eq: 2.

$$
\begin{aligned}
&\text{Precision}P_x = \frac{TP}{TP+FP} \\
&\text{Recall}R_x = \frac{TP}{TP+FN} \\
&F_1 - macro = \frac{1}{n}\sum_x f1_x = \frac{1}{n}\sum_x \frac{1P_xR_x}{P_x+R_x}
\end{aligned}
\tag{2}
$$

Fairness Metric In addition to the f1-macro metric to evaluate classification performance. Fairness Metric is calculated using demographic parity across all classes as shown in eq:3. This ideal score should be equal to 1, meaning that both genders are equally considered by the machine learning algorithm.

$$\frac{1}{N}\sum_{i,g=m,f}\frac{\max(\text{job}_i^{g(f)},\text{job}_i^{g(m)})}{\min(\text{job}_i^{g(f)},\text{job}_i^{g(m)})} \tag{3}$$

5. Experimental Settings

We will be presenting the experimental setting chosen

Loss Function Another tool to address unbalanced multi label setting is to adapt the loss function with class weights, This seems to be the promising solution than data augmentation for the imbalance setting because Contextual word embeddings do not gain from data augmentation for the models like BERT [7]. This is also proven in the experiments as BERT without data augmentation outperforms BERT with data augmentation. Class weights are calculated using the eq:4. From eq: 5 θ represents the classical loss function. This classical loss function is adapted by dividing it with class weights calculated before. Using this loss function makes neural network model like BERT give more importance to classifying a sample belonging to a minority class (class with less number of samples) than a sample belonging to a majority class (class with lot number of samples).

$$w_j = \frac{\text{Total Number of Samples}}{\text{Number of classes} \times \text{Number of Samples of Class}_j} \tag{4}$$

$$\begin{aligned}
\theta &= -log(\frac{exp(x[class])}{\sum_j exp(x[j])})\\
\theta &= -x[class] + log(\sum_j exp(x[j]))\\
loss(x, class) &= weights[class] \times \theta
\end{aligned} \tag{5}$$

5.1. Confidence Pipeline

In the data provided by DEFI-IA-2021, we have some job descriptions with randomly shuffled labels, and some job description which are very vague and even hard for humans to classify. Getting inspired from the work of Jiang, Heinrich, et al[6]. Instead of relying on the $argmax$ of the softmax layer, we take the whole probability distribution from the softmax layer and build a confidence score using Entropy($\sigma(\rightarrow z)_i = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}}$). As shown in Jiang, Heinrich, et al[6] this score helps to separate the good predictions from the bad predictions using the last softmax layer. Using this score we split the samples with good confidence score and bad confidence score, we split it using a confidence value of 0.63, which we empirically find this by finding max f1-macro score of good predictions and min of f1-macro score of bad predictions. As shown in the fig: 1, from good predictions we find the outliers by matching the original labels to the predicted labels. One of this outlier example is presented in table:2. From the bad examples, we could seperate the vague samples one of the example is presented in table:2 with the respective confidence. For the bad samples, we used 4 different BERT models and perform voting to take the best prediction.

Outlier Example	Ambiguous Example
Chris has contributed his vast knowledge of HP Software and his reasonable, rational manner to strategic board decisions.A frequent Conference Track Chair and Customer of Excellence Award Judge, Chris is a seasoned software veteran, who will continue to add expertise and depth to the community.	He also is available for telephone consultations for those needing advice from a distance. He uses nutrition and Neurologic Relief Centers Technique to help stubborn health conditions including fibromyalgia, neuralgias, chronic pain, headaches, fatigue, weight gain and hormonal imbalances. He can be reached at 361-485-0449. His website is http://www.drholcombsnaturalhealth.c
Confidence Score: 87	**Confidence Score:** 52
Train Label: Architect	**Train Label:** Physician

Table 2. Confidence Example

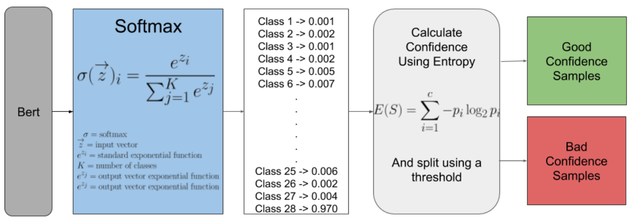

Bert
Softmax
$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
σ = softmax
$\vec{z}$ = input vector
e^{z_i} = standard exponential function
K = number of classes
e^{z_j} = output vector exponential function
e^{z_j} = output vector exponential function
Class 1 -> 0.001
Class 2 -> 0.002
Class 3 -> 0.001
Class 4 -> 0.002
Class 5 -> 0.005
Class 6 -> 0.007
Class 25 -> 0.006
Class 26 -> 0.002
Class 27 -> 0.004
Class 28 -> 0.970
Calculate Confidence Using Entropy
$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$
And split using a threshold
Good Confidence Samples
Bad Confidence Samples

Figure 1. Confidence Pipeline

5.2. Other Experimental Settings

Bert Based Models: We used various base versions of BERT based models like BERT, Roberta, DistillBERT, XLNET. Including the base versions, we also used Large versions of BERT and Roberta. The difference between Large and base versions is the large versions has more layers and around 3 times the parameters to the base model.

Token Size: BERT has a token size limit of 512. In BERT, words are converted into tokens and then fed into the network. Since most of the sentences lie below the length of 256, we chose the token size of 256 for all the experiments.

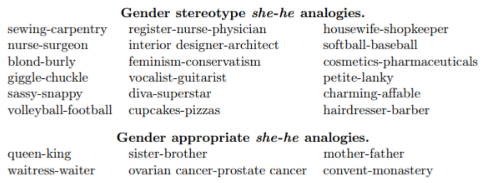
Batch Size: Due to limited computational power available, we use the batch size of 80 if we use Titan GPU's or 50 if we use 1080Ti GPU's.

Learning Rate: We first started with the recommended learning rate **2e-5** presented in Sun, Chi, et al[10]. But due to the limited computation power available for us and decreased batch size to the recommended setting, we got better results by increasing the learning rate to 3e-5. In contrast to what is recommended in Sun, Chi, et al[10], we found that the increased learning rate 3e-5 did not result in catastrophic forgetting.

Warmup Steps: Instead of using 10 percent of the total steps as warmup steps, we used a fixed rate of 700 steps as warmup steps. We used linear warmup scheduler, where the learning rate is increased linearly to 3e-5 within 700 steps.

Momentum: Momentum controls the aggressive gradient behavior, we used a recommended setting of 0.01

Learning Rate Decay: Learning decay helps decreasing the learning rate after each step in the batch gradient descent, we used a value of 0.001 which helps in fine tuning



Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Figure 2. Gender Bias in Word Embedding [3]

at the last steps.

6. Fairness

De-biasing Word Embeddings It is well accepted and defined that word embedding contain a lot of bias in terms of gender, as they are not designed with the fairness in mind, the only focus was on grouping the similar words in the vector space. This might make some gender sensitive words group together like(man ⇔ doctor) and (woman ⇔ nurse). This bias comes from the vector space as the word **doctor** may be closer to a man than **nurse**, this is proven by olukbasi, Tolga, et al[3]. De-biasing this vector is proposed in Bolukbasi, Tolga, et al[3]. In this technique, the goal is to debias the vector space for some gender sensitive professional related words. One important aspect to consider is not to debias all gender specific words in the vector space, some of the words needs to preserve this gender bias like King ⇔ Queen as these words are called Gender appropriate analogies, if these words are debiased then they loose their meaning. These words are shown in the fig: 2 the first part of the words represent Stereotype words which needs to be debiased and second part of the words represent gender appropriate words which should not be debiased. We tried this soft-debiased word embedding in conjunction with simple machine learning algorithms to address the Fairness Problem.

De-biasing Pre-Trained Models Like the word embeddings, pre-trained models like BERT carry a lot of gender bias as they are trained on gender biased data. So the weight in this pre-trained network inherently induce this bias in our text classification task. So it is important to Debias the pretrained network in terms of gender. One of the experiment we tried is to decompose the problem into **min-max** problem. Fig 3 shows the dual problem proposed for de-biasing pre-trained models. This has two training problems one is addressed for the job classification and other is addressed for gender classification. We use the same pre-trained model for the two problems. Orange side of the architecture represents the gender classification and blue side of the architecture represents job classification task. Job classification module takes job description and Job label, the blue classification head tries to classify the job label. Gender classification module in orange takes job description and gender as the label, the orange classification head tries to classify the gender label(Male of Female). Intuitively, this would exonerate the weight sensitive attributes, which is why instead of giving the original gender label we draw this randomly from (male and female). In this way about 50 percent of the time the algorithm makes wrong predictions. Ideally, this would make the gender sensitive attributes get debiased with the help of backpropogation.

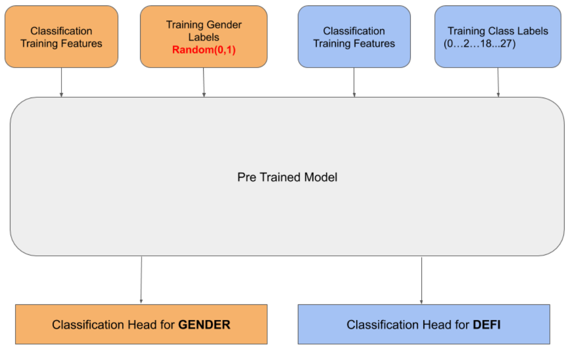

Classification Training Features
Training Gender Labels Random(0,1)
Classification Training Features
Training Class Labels (0…2…18...27)
Pre Trained Model
Classification Head for GENDER
Classification Head for DEFI

Figure 3. Debiasing Pre-Trained Models

7. Results

Technique	F1-Macro Score
TF-IDF with Logistic Regression	0.73244
Word Embedding with Random Forest	0.72456
BERT base	0.79567
RoBERTa	0.80453
DistillBERT	0.79241
XLNET	0.79365
BERT with Augmentation	0.78413
RoBERTa with Confidence Pipeline	0.79637
Voting BERT Based Classifiers	0.80928

Table 3. Classification Results

Fairness Technique	Fairness	F1 Score
TF-IDF Logistic Regression	5.1123	0.73244
Debiased Word Embedding with Random Forest	3.6783	0.71358
Word Embedding with Random Forest	4.5634	0.72456
Debiased BERT with Dual Task	2.9000	0.78769
BERT	3.1000	0.79758

Table 4. Debiased Technique and Results

8. Conclusion and Discussions

With this challenge, we focused on maximising accuracy while minimizing gender disparity index for job classification using text. We observed that gender disparity index increases with increase in the f1-score. Debiased BERT with Dual Task presented in this report tried to reduce the fairness metric but at the cost of decrease in f1-score. We believe an adversarial based approach might result in better performance.

References

[1] CommonCrawl commoncrawl. https://www.wikiwand.com/en/Common_Crawl. Accessed: 2010-09-30. 1

[2] Neural Network Methods mlp. https://www.amazon.com/Language-Processing-Synthesis-Lectures-Technologies/dp/1627052984/ref=as_li_ss_tl?ie=UTF8&qid=1502062931&sr=8-1&keywords=Neural+Network+Methods+in+Natural+Language+Processing&linkCode=sl1&tag=inspiredalgor-20&linkId=d63df073fea3ebe2d405820570b3ff03. Accessed: 2010-09-30. 2

[3] T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings, 2016. 4

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 2

[5] C. Du, Z. Chen, F. Feng, L. Zhu, T. Gan, and L. Nie. Explicit interaction model towards text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6359–6366, 2019. 2

[6] H. Jiang, B. Kim, M. Guan, and M. Gupta. To trust or not to trust a classifier. *Advances in neural information processing systems*, 31:5541–5552, 2018. 3

[7] H. T. Madabushi, E. Kochkina, and M. Castelle. Cost-sensitive bert for generalisable sentence classification with imbalanced data. *arXiv preprint arXiv:2003.11563*, 2020. 3

[8] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013. 2

[9] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 2

[10] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019. 3