PGM PROJECT - DELIVERABLE 2

Mohammad Nur AMIN **Jatin BABBAR** **Shirin BASHIRIAMID** **Madhubhani RANCHA GODAGE**

Orhan SOLAK

January 28, 2021

1 Implementation

In this paper, we discuss about the implementation details of all the models we implemented to address the problem of job classification.

The dataset for this task is a set of job descriptions and the gender of candidates. In the first step we make the data ready to be fed our models, the text data is cleaned and made easier to process by removing some unnecessary information like the punctuation, abbreviations, special characters, URLs, emails, non-digit data and stop-words.

After preprocessing, the most frequent and important words [Vu,] in dataset are extracted such as in figure 1.

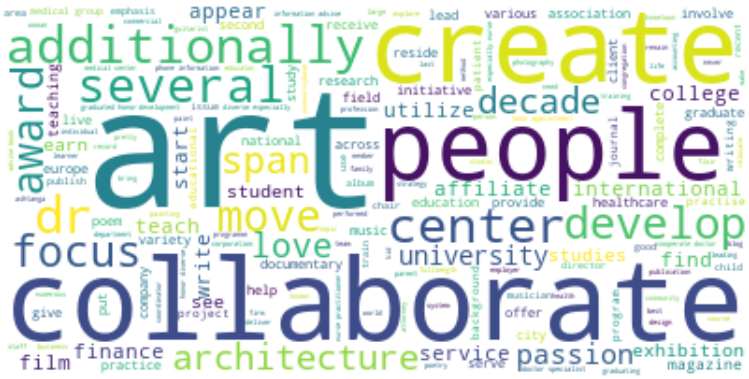

Figure 1: Cleaned data

1.1 Multinomial Naive Bayes

In this study we consider text objects as the input (X) and would like to allocate it to one category that is our output (Y). Our model learn a classifier function $f : X \rightarrow Y$ s.t $f(x) = y$ and $y \in Y$ gives the correct category for $x \in X$.

Due to the imbalance in the data, we select equal number of observations (750) randomly from each class. We trained the 70 percent of data and test on the remain part.

We used *MultinomialNB* function in *sklearn* library with **CountVectorizer** and **TfidfTransformer** over a pipeline. Basic explanations of these functions:

1. **CountVectorizer**: Convert a collection of text documents to a matrix of token counts
2. **TfidfTransformer**: Transform a count matrix to a normalized tf or tf-idf representation

1.2 Latent Dirichlet Allocation

We use LdaMulticore model in Gensim library. Before starting the modelling, we convert the text into bag of words [Kapadia,]. Dictionary is used to map key, value pairs where key is the word and value is the number of occurances of that word. Then dictionary is created for each document.

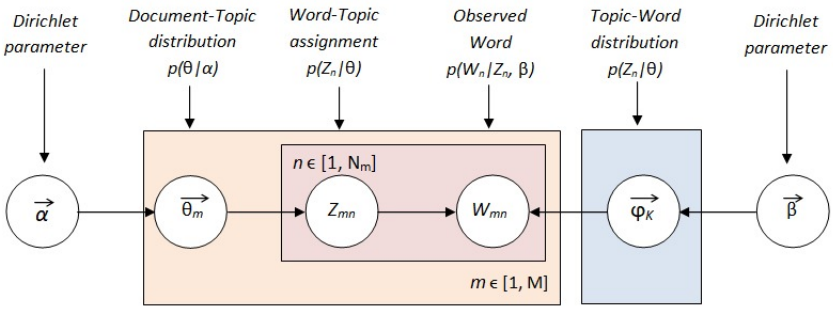

Figure 2: Graphical model of LDA [Lee et al., 2018]

Then we start training the model with default parameters. Number of topics is set to 28 because our scenario has 28 number of job categories. Bag of words and dictionary are passed to create the model. Using the created model, keywords for each topic can be seen (Figure 3).

```
[(15,
  '0.011*"college" + 0.009*"art" + 0.008*"collaborate" + 0.007*"receive" + '
  '0.007*"university" + 0.004*"ashtanga" + 0.004*"arts" + 0.004*"architecture" '
  '+ 0.004*"healthcare" + 0.004*"teach"'),
 (16,
  '0.013*"college" + 0.006*"architecture" + 0.006*"ashtanga" + 0.006*"decade" '
  '+ 0.005*"earn" + 0.005*"university" + 0.005*"art" + 0.004*"practise" + '
  '0.004*"several" + 0.004*"presently"'),
 (2,
  '0.007*"college" + 0.006*"blue" + 0.006*"dr" + 0.006*"university" + '
  '0.006*"health" + 0.006*"art" + 0.006*"additionally" + 0.005*"book" + '
  '0.004*"develop" + 0.004*"medical"'),
 (9,
  '0.006*"architecture" + 0.006*"collaborate" + 0.006*"several" + '
  '0.005*"decade" + 0.005*"additionally" + 0.005*"finance" + 0.005*"create" + '
  '0.005*"university" + 0.004*"book" + 0.004*"client"'),
 (17,
  '0.010*"art" + 0.010*"college" + 0.006*"decade" + 0.006*"university" + '
  '0.006*"medical" + 0.006*"finance" + 0.005*"love" + 0.005*"additionally" + '
  '0.004*"center" + 0.004*"people"'),
 (25,
  '0.012*"collaborate" + 0.010*"art" + 0.007*"ashtanga" + 0.007*"college" + '
  '0.006*"university" + 0.006*"presently" + 0.005*"create" + 0.005*"span" + '
  '0.005*"earn" + 0.005*"additionally"'),
 (13,
  '0.010*"ashtanga" + 0.008*"people" + 0.007*"art" + 0.006*"love" + '
  '0.006*"additionally" + 0.004*"create" + 0.004*"university" + '
  '0.003*"passion" + 0.003*"television" + 0.003*"receive"'),
```

Figure 3: Keywords in each topic

To visualize the model properly, pyLDAvis package is used. We can properly see the relationships between the topics and most relevant terms for each topic visually.

1.3 Gibbs Sampling

1.3.1 Constructing Gibbs sampler

According to the definition of a Gibbs sampler, in each iteration a new value is assigned to variable Z_i by sampling from the conditional distribution.

$$P\left(Z_i \mid z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_r^{(t)}\right)$$

Thus, to assign the value of $\mathrm{L}_1^{(t+1)}$, we need to compute the conditional distribution

$$P\left(\ \mathrm{L}_1 \mid \mathrm{L}_2^{(t)}, \ldots, \mathrm{L}_N^{(t)}, \mathbb{C}, \theta^{(t)}; \mu\right)$$

To assign the value of $\mathrm{L}_2^{(t+1)}$, we need to compute

$$P\left(\ \mathrm{L}_2 \mid \mathrm{L}_1^{(t+1)}, \mathrm{L}_3^{(t)}, \ldots, \mathrm{L}_N^{(t)}, \mathbb{C}, \theta^{(t)}; \mu\right)$$

and so forth for $\mathrm{L}_3^{(t+1)}$ to $\mathrm{L}_N^{(t+1)}$. Similarly for θ,

$$P\left(\theta_0^{(t+1)} \mid \mathrm{L}_1^{(t+1)}, \mathrm{L}_2^{(t+1)}, \ldots, \mathrm{L}_N^{(t+1)}, \mathbb{C}, \theta_1^{(t)}; \mu\right)$$

$$P\left(\theta_1^{(t+1)} \mid \mathrm{L}_1^{(t+1)}, \mathrm{L}_2^{(t+1)}, \ldots, \mathrm{L}_N^{(t+1)}, \mathbb{C}, \theta_0^{(t)}; \mu\right)$$

In the initial iteration, all the information is at hand at this point during this sampling process that includes word count for each document and the labels assigned to each document, current labels and values for respective document and θ.

When we want to sample the new label for document j, we temporarily remove all information about this document from that collection of information. Then we look at the conditional probability that L_j and we sample the new label $\mathrm{L}_j^{(t+1)}$ by choosing randomly according to the relative weight of those two conditional probabilities. Sampling to get the new values $\theta^{(t+1)}$ operates according to the same principal.

Finally, we show how to select all of the new document labels L_j and the new distributions θ during each iteration of the sampler. The conditional probability looks like

$$P\left(\ \mathrm{L}_j \mid \mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \theta_0; \mu\right) = \frac{P\left(\ \mathrm{L}_j, \mathbf{W}_j, \mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \theta; \mu\right)}{P\left(\mathbf{L}^{(-j)}, \mathbb{C}(-j), \theta; \mu\right)} = \frac{P\left(\mathbf{L}, \mathbb{C}, \theta, ; \mu\right)}{P\left(\mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \theta; \mu\right)}$$

where $\mathbf{L}^{(-j)}$ are all the document labels except L_j , and $\mathbb{C}^{(-j)}$ is the set of all documents except $\mathbf{W}_{(j)}$. The distribution is over all possible outcomes for L_j.

Similar methodology is followed to sample for new values of θ. We need to derive an expression for the probability of θ given all other variables.

$$P(\theta \mid \mathbb{C}, \mathbf{L}; \mu) = P(\mathbb{C}, \mathbf{L} \mid \theta) P(\theta \mid \mu)$$

1.4 Gender Debiasing

For gender debiasing, the methodology can be summarized in Figure 4.

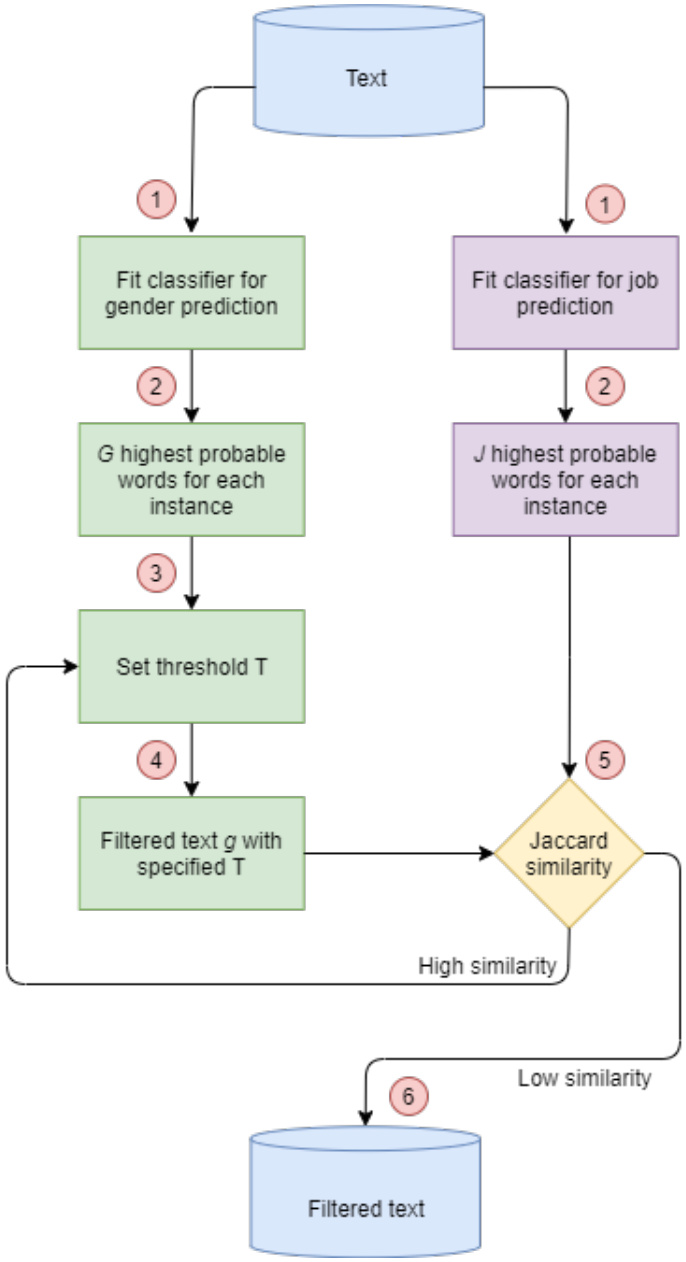

Figure 4: Gender debiasing

First a classifier is fit to predict the gender from text so that LIME can be used to interpret the model to extract the most probable features of set G (words) that impacted during gender classification. These extracted features essentially influenced the model when classifying job category. A subset of features g from G features is then extracted with a threshold, T to extract the highest probable features. The threshold is defined by taking the mean, μ of G with a specified standard deviation, σ to limit number of feature extraction. Finally, the extracted features are filtered out from the original text.

$$T \leq \mu + \sigma$$

However, we need to be careful to eliminate the number of features as we could end up eliminating some words that would have high probability for job classification. In order to avoid elimination of useful features for job classification, LIME is again applied on the job classifier to extract the set of J highest probable features for job classification. Jaccard similarity is checked between g and J. As the Similarity scores increases, it indicates inclusion of influential features in

g for job classification that results in dropping the f1 score. So we have an optimization problem where we want to eliminate as much as gender biased words at the same time retaining the f1 score.

References

[Kapadia,] Kapadia, S. Topic modeling in python: Latent dirichlet allocation (lda). Accessed: 2021-01-28.

[Lee et al., 2018] Lee, J., Kang, J.-H., Jun, S., Lim, H., Jang, D., and Park, S. (2018). Ensemble modeling for sustainable technology transfer. *Sustainability*, 10(7):2278.

[Vu,] Vu, D. Generating wordclouds in python. Accessed: 2021-01-28.