Understanding Gender-coded Wording in Job Postings with Word-vectors and BERT

Stanford CS224N {Custom} Project

Mengmeng Ji
Department of East Asian Languages and Cultures
Stanford University
mmj710@stanford.edu

Abstract

Biased gender-coded words still exist in job advertisements today. These words can strongly influence candidates' perception of the job, discourage diverse candidates from applying and even reduce their sense of belonging to the occupation. Gaucher et al (2011)[1] provides two lists of known gender-coded words in job advertisements. One for masculine and one for feminine coded words. However, these lists are likely incomplete and may miss more subtle or sentence-level gender coding. In this paper I propose that by using word vectors and BERT, we can discover additional gender-coded words and detect gender bias at the grammatical/sentence level.

1 Key Information to include

- Mentor: John Hewitt

2 Introduction

In job postings, biased Gender-coded words can potentially have negative impact on job applicants, such as misleading candidates' perception of the job, discourage diverse candidates from applying and even reduce their sense of belonging to the occupation. Although reducing gender bias has been a heated topic in Natural Language Processing. However, few studies have worked on reducing gender bias in job advertisements,a topic which may contribute greatly to gender equality in working places. A psychological studies paper, (Gaucher et al.2011)[1] points out the existence and negative influence of gender-coded wording in job postings, especially that the masculine wording in job advertisements leads to less job interest and anticipated belongingness among women. This paper points out the importance of reducing gendered wording in job postings. It is crucial to integrate these findings into modern NLP tools. Doing so would help ensure that online job postings do not turn away promising candidates and help reducing the gender gap in the technology industry.
Since its publication, this paper has already inspired two automated tools which have been built for the purpose of detecting and reducing gendered wording in job postings. Texio[1] is a software tool that provides feedback on making the job posting more engaging and inclusive, including detecting gender-coded words. Gender Decoder[2] detects masculine-coded or feminine-coded words based on a list of gender-coded word stems. Both of the tools are developed based on the gender-coded word lists (one masculine-coded and one feminine-coded) in (Gaucher et al.2011)[1]. However, these lists are likely incomplete and may miss more subtle or sentence-level gender coding. In addition, there is still a lot of space for further work to detect gender bias in job advertisements by using machine learning models as well as increasing the general accessibility of such tools.

[1] https://textio.com/blog/welcome-to-the-world-textio-flow/13035173423
[2] http://gender-decoder.katmatfield.com/

To better understand gender-coded wording in job posting and to improve gender bias detection, this project has two tasks: 1) to enrich the existing gender-coded word lists, I use word-vectors to identify additional gender-coded words; 2) to explore a gender bias detection method at grammatical/sentence level, I use Bidirectional Encoder Representations from Transformers (BERT), a sentence level language model to classify job postings based on their gender coded.

3 Related Work

The paper that inspired me to do this project is (Gaucher et al. 2011)[1], which discusses the existence and negative influence of gender-coded words. Gaucher, Friesen and Kay [1] propose that gender-wording in job recruitment materials discourages diverse candidates from applying for the job because it influence their perceived match with the occupation. The paper concludes that masculine wording in job advertisements leads to reduced job interest from women, and a general reduced sense of belongingness of women candidates to the overall profession. In addition, gender-coded words were found to have a far greater impact on women than men i.e. men were not significantly affected by the presence of feminine coded words. This groundbreaking research not only points to a valuable research direction in NLP, but also provides a great starting point for further research by offering lists of gender-coded words. It also shows that there is still significant room for further works to explore detecting gender bias in job advertisements, such as using machine learning models as well as increasing the general accessibility of such tools.
One approach to enhance gender bias detection is to explore the potential existence of additional gender-coded words. The search for such words can, for instance, be accomplished by looking for similar words in the space of word embeddings. Pennington et al.2014 [2] proposes that GloVe vectors outperform other approaches on similarity tasks, which makes them the most appropriate choice for finding additional gender-coded words.
In additional to specific word detection, analyzing how sentences or paragraphs are gender-coded is a more complicated but promising strategy. To experiment with this idea, I construct a dataset of gender-coded job postings labeled based on the existing gender-coded word lists. Then I fine-tune BERT (Devlin et al.2018)[3] as a classifier on this dataset. BERT, as a multi-layer bidirectional Transformer, is pre-trained on masked word prediction and next sentence prediction task, then fine-tune on any NLP dataset of choice. Although BERT has achieved great performance in many NLP tasks[4], its potential in text classification tasks can be tricky, especially when the classification classes are unbalanced or the size of the dataset is limited. Sun et al. 2019[5] provides useful techniques of fine tuning BERT on text classification task. One such trick, which is also applied in this paper, is to take the first 64 tokens and the last 64 tokens of an input text when its length is longer than 128. This was found to give better performance than simply using the first (or last) 128 tokens.

4 Approach

I investigate the task of detecting gender-coded text in four complimentary steps:

1. I use word vector similarity to discover additional gender-coded words in job advertisements. Words are compared using the cosine similarity between their respective word vectors. These comparison are used to construct a list of candidate gender-coded words that are close to the words in genderdecoder's masculine/feminine-coded lists. As a starting point, I use the pre-trained glove vectors. But the language of job-postings is quite specific so I also train GloVe vectors on the Data Analyst job postings dataset. I analyze the resulting lists and select any additional gender-coded words, as well as look for examples of these words being used in the dataset.
2. The next step is to label the job posting dataset using the gender-decoder word lists. This is accomplished using the genderdecoder python package[3]. Each job posting is labeled as "neutral", "masculine", "strongly-masculine", "feminine" or "strongly-feminine". An example is considered neutral if the number of masculine words equals the number of feminine words used. If the number of masculine words is larger than the number of

[3] `https://pypi.org/project/genderdecoder/0.3/`

feminine words then the example is considered masculine coded. While these labels are based on heuristics and may not be perfect, they do provide a good starting point for exploring gender bias in job postings. The gender-decoder package was developed by the same authors as the Gender Decoder website[4].

3. The labeled dataset is used to train a (BERT based) classifier on this dataset. Since BERT does not just focus on individual words but provides a holistic encoding of the phrase or sentence being analyzed, it may be able to generalize beyond word-level gender-bias detection. I work with a smaller pretrained version of BERT referred to as **BERT-small**. This pretrained model is publicly available through tensorflow hub[5]. This version of BERT contains 4 encoder blocks with a hidden size of 512. The multi-head attention uses 8 attention heads. I also apply some fine-tuning tricks from [5] to improve the performance of the classifier.
4. As a final step in my analysis, I compute the gradients of BERT's predictions with respect to the input sentence. This allows me to discover words which sometimes make the job posting more masculine and other times more feminine (depending on their context).

5 Experiments

5.1 Data

5.1.1 Dataset and Preprocessing

I use the Data Analyst Jobs dataset from Kaggle[6]. It contains 2253 job listings for data analyst positions collected from Glassdoor[7]. It contains 15 different fields, including Job Title, Job Description, Salary Estimation, etc. The "Job Description" category contains the most relevant information for this project.
To pre-process the data, I first clean up the data by using the clean-text[8] python package. I lowercase the text, remove all the irrelevant information such as line breaks, emails, urls, phone numbers, currency symbols, etc.

Having cleaned-up the job descriptions I then run the genderdecoder python package[9] to assess each job listing and label them with its gender code. Among the 2253 job listings, 1509 postings are masculine-coded, 178 are strongly masculine-coded, 381 are feminine-coded, 24 are strongly feminine-coded, 161 are neutral.

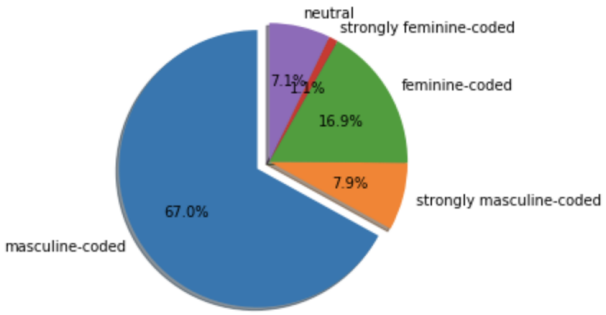

Figure 1: Gender-coded Data Analyst Job Descriptions

[4] http://gender-decoder.katmatfield.com/
[5] https://tfhub.dev/google/small_bert/bert_uncased_L-4_H-512_A-8/1
[6] https://www.kaggle.com/andrewmvd/data-analyst-jobs
[7] www.glassdoor.com
[8] https://pypi.org/project/clean-text/
[9] https://pypi.org/project/genderdecoder/

5.1.2 Creating Dataset for BERT

The genderdecoder package originally provides five labels, as figure 1 shows. Because the dataset is relatively small, I merge "strongly masculine-coded" into "masculine-coded", "strongly feminine-coded" into "feminine-coded" to get three classes.
Then the dataset of 2232 examples is split into a training set of 1992 job descriptions, a development set of 200 samples and a test set of 40 samples. The test set is intentional left very small, so that the results can be analyzed manually.
I then lowercase and tokenize the data using BERT's tokenizer.
The dataset needs to be standardized into the format that is used by the BERT model. It expects a fixed structure of [CLS, Text_a, SEP, Text_b]. In this classification example Text_a is the text of the job description, Text_b is left blank.
For this BERT model, instead of simply using the first 128 tokens of the review as text_a, I use the head+tail truncation methods mentioned in [5]. This involves taking the first 64 and the last 64 tokens of the review. If the review is shorter than 128 tokens the entire text is used, if the review is longer, I throw out the middle tokens and make sure to keep the first and final (head and tail) 64 tokens. The motivation for this is that the head and tail of a job description (or other text) tend to contain the most valuable information [5].
Finally, I write this dataset to the disk as a tfrecord file. By writing it to disk, I ensure that the train/dev/test splits is fixed.

5.2 Evaluation method

The model is evaluated primarily on its ability to correctly identify masculine-coded job postings. Failing to classify a masculine coded job posting as masculine should be heavily penalized i.e. the model should have high recall. On the other hand, the model should also have a high precision on masculine coded job postings as we do not want it to generate too many false positives (which would be tiresome to review). In order to balance the requirement for both high recall and high precision, the main evaluation metric will be the F1 score (harmonic mean of recall and precision) on masculine-coded job postings. The precision and recall of neutral and feminine-coded job postings is of less concern. Previous research ([1]) has shown that feminine-coded job postings do not discourage male candidates from applying.

5.3 Experimental details

5.3.1 Glove vectors

For the first experiment, I use the pre-trained glove-wiki-gigaword-200 GloVe vectors from gensim[10]. I use cosine-similarity between these word-vectors to find the top 50 words and the top 10 word stems that resemble those from genderdecoder's lists (excluding stems that are already in the lists). However, since the word-vectors are not based on job-postings, the results are mostly irrelevant (noise words).
I then train GloVe vectors on the Data Analyst job postings dataset. To do this, I first construct a co-occurrence matrix with a context window size of 4. Based on this co-occurrence matrix, I train GloVe by using the mittens package [11]. I then retrieve the lists of 50 additional gender-coded words and 10 word stems ranked by cosine similarity to the known gender-coded words. The produced lists were mostly all relevant, in contrast to the lists produced with pre-trained word vectors.

5.3.2 BERT

For the BERT experiment, as stated in Approaches section, I use Bert small, which has 4 encoder blocks with a hidden size of 512 and 8 attention heads. The model weights are initialized from Google tfhub checkpoint. I add a single linear classifier layer on top. Before the output layer, I use a dropout layer to reduce over-fitting on this small dataset. To train the model, I use mini batches of batch size 64, and set learning rate of 0.00005. As for the starting point of tuning hyperparameters, I

[10] `https://github.com/RaRe-Technologies/gensim-data`
[11] `https://github.com/roamanalytics/mittens`

use reasonable defaults are provided in the official github classification example.[12]
The initial performance, as the figure shows, is unbalanced because there are much more masculine-coded examples(1493) than feminine-coded(356) or neutral(143) examples. To better train the model, I balance the dataset by repeating the feminine-coded and neutral examples to roughly match with the amount of masculine-coded examples. The balanced training set includes 1493 masculine-coded examples, 1424 feminine-coded and 1430 neutral examples. With balanced training set, the model's performance on development set improves.

5.4 Results

5.4.1 Results of Baseline Model

The word-vector similarity experiment's results on the pre-trained word vecotrs are not ideal. Among the top 10 word stems that are closest to genderdecoder's feminine-coded word lists, 4 are unintelligible, the other 6 are: *link*, respect*, thought*, negoti*, commun*, affect**. Among the top 10 word stems that are closest to genderdecoder's masculine-coded word lists, 4 are unintelligible, the other 6 are: *evalu*, argu*, tenac*, resourc*, assess*, olymp**.
The results obtained using the GloVe vectors trained on the target dataset are much more promising. The top 10 word stems similar to gender-coded word stem list are all intelligible. The additional feminine-coded word stems are: *commun*, execut*, complet*, provid*, assist*, drive*, manag*, innov*, convers*, edit**. The additional masculin-coded word stems are: *use*, demonstr*, offer*, support*, manag*, identifi*, document*, work*, orga*, execut**. In terms of their meanings, part of speech and context, these words are consistent with the original gender-coded word list. Therefore they are eligible as additional gender-coded words.
Moreover, there are some overlapping words between the two lists, such as *execut*, manag**. They can be understood as neutral words because the are shared by both feminine and masculine-coded word lists. However, the index of the shared words in the two lists are different, for example, $execut = femresults[1] = masresults[9]$. This means this word is more likely to be used as a feminine-coded word rather than a masculine-coded word. It is also likely that the gender-coding of these words depends on their context.
Thus, training GloVe vectors on the target dataset has lead to candidate words which are consistent with the existing list and are descriptive of people's behavior or personality.

5.4.2 Results of BERT Model

Because of the imbalance in the dataset distribution, a naive baseline which predicts "masculine-coded" on all examples would yield a precision of 80% and a a corresponding F1 score of 89% (on masculine-coded postings). However, such a baseline would get an F1-score of 0 on feminine and neutral job-postings.
The first table (1) shows the performance of the model on the dev set after training on the unbalanced dataset. The overall accuracy when using original training set is 80%. However, it is not achieved by simply predicting all examples are masculine coded, as the precision on masculine-coded examples is improved to 90%. As previously noted, we want the model to classify masculine-coded postings with high recall, but not generate too many false positives. This initial model gives better precision than the naive baseline. As Table 1 suggests, the precision of masculine-coded examples is 90%, recall is 80% and f1 score is 89%. The results show that the model is doing well identifying masculine-coded job postings while not completely neglecting the feminine and neutral examples.

I also re-trained the model on the balanced version of the dataset where feminine and neutral examples repeat multiple times to match the number of masculine examples. The results are shown in table 2. With the balanced training set, the f1 score of masculine-coded is higher. Additionally, the f1 score of classifying feminine-coded and neutral increased significantly, by 8% and 7% respectively. Thus, balancing the dataset improved the performance on all three classes.

To study the impact of different words on the classification result of the fine-tuned BERT model, I analyze the gradient of the output probabilities with respect to the words in the input. To construct this gradient I take the gradient of the probabilities with respect to the embedding of each word in the

[12] `https://colab.sandbox.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb`

	precision	recall	f1-score	support
masculine-coded	0.90	0.88	0.89	160
feminine coded	0.48	0.52	0.50	28
neutral	0.10	0.12	0.11	12
accuracy			0.80	200
macro avg	0.49	0.51	0.50	200
weighted avg	0.81	0.80	0.80	200

Table 1: BERT Classification quality on dev set after training on unbalanced training set

	precision	recall	f1-score	support
masculine-coded	**0.90**	**0.89**	**0.90**	160
feminine coded	0.55	0.61	0.58	28
neutral	0.20	0.17	0.18	12
accuracy			0.81	200
macro avg	0.55	0.56	0.55	200
weighted avg	0.81	0.81	0.81	200

Table 2: BERT Classification quality on dev set after training on balanced training set

input sentence. If a word occurs more than once in the input, those contributions are summed up. A negative gradient of the masculine classification probability with respect to a given word's embedding vector indicates that removing the word would make the job posting more masculine. In other words, if intensity of the word increases, the job posting would be less masculine. Naturally, words can only be added or removed, not increased in magnitude, so the gradient is only an approximation. Similarly, a positive gradient indicates that a word's presence is directly contributing to the masculine classification. Thus, both negative and positive values of the gradient are of interest. I therefore sort the words in the input depending on the gradient magnitude in order to analyze those that have the largest influence (either positive or negative).

6 Analysis

The results show that some words that are regarded as gender-coded in the previous list are in fact neutral in terms of its meaning or implication. BERT shows that their contribution to gender-bias depends more on how they are used in context rather than their original meaning. For example, *responsible* or *responsibility* have the feminine-coded stem *respons**. Judging from their gradients in test set examples, sometimes they have positive gradients when contribute to a judgement of a text as "masculine-coded" in some cases, while contributing to judging a text as "feminine-coded" or "neutral" in other examples. This proves that by implementing BERT, we can discover that words can be gender-coded depends on how they are used in sentences and context. Similar examples include *analy**, which was regarded as masculine-coded word stem, but also contribute to forming feminine-coded or neutral texts based on different context.

7 Conclusion

My project shows that word vectors and modern NLP models such as BERT can improve the detection and understanding of gender bias in the wording of job postings. GLoVe vectors and cosine similarity allow us to find additional gender-coded words. Meanwhile, experiments with BERT show that gender bias does not only emerge because of the usage of certain words but also from the context in which the words are placed. I find that for some words that are known to be gender-coded, the contribution to the gender-bias (masculine or feminine contribution) is at least partially determined by the context and not just the meaning of the word itself.
Another achievement of this project is the addition of of the gender-coded label to the job posting dataset. Based on an extensive search of public datasets, it appears to be the first dataset of this kind. However, the dataset is relatively small and it relates solely to data analyst job descriptions. This makes it relatively limited in complexity and usefulness for advancing research into gender-coded

job descriptions. For future work, I plan to enlarge the dataset to enable more interesting findings. For example, job descriptions from different occupations, regions, and different periods of time can be included.

References

[1] Danielle Gaucher, Justin Friesen, and Aaron C Kay. Evidence that gendered wording in job advertisements exists and sustains gender inequality. *Journal of personality and social psychology*, 101(1):109, 2011.

[2] Jeffrey Pennington, Richard Socher, and Christopher 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.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning, 2016.

[5] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583, 2019.