ReadMe

Pre-requisites

- The Soft Debias method:
  - For solving of the optimisation problem we used the *cvx solver* . It is a Matlab based package for convex optimisation that conveniently handles Semidefinite programs. The package will have to be downloaded and installed to run the softDebias.m function.
  - To understand the computation of converting the optimisation problem in a format that can be provided as input to the solver, please look at the appendix in the project report (*Section A.2).*
  - Please visit: http://cvxr.com/cvx/download/ to download the solver and,
  - Please visit: http://cvxr.com/cvx/doc/install.html for installation instructions.
  - We have provided a pre-processed transform T in T.mat file for use due to the long runtime of solving. This is specifically used on the Google News Word2Vec set provided by Bolukbasi *et al.*

External Resources

- In order to preserve consistency and have a reliable way to check our results, we imported the dataset used by Bolukbasi et.al for the implementation of their paper - *"Man is to computer programmer as women is to homemaker? Debiasing Word Embeddings,"* . This dataset can be found at https://drive.google.com/drive/folders/0B5vZVlu2WoS5dkRFY19YUXVIU2M .
  - Word vectors dataset for the Hard Debias, Soft Debias and Schmidt Method were taken from the dataset referenced above. Word lists for occupation words, gender-specific words and equalise words can be found within our code folder. The lists can also be found in the Appendix of the project report *(Section A.3, A.4 & A.5)*
  - The textfile w2v_gnews_small.txt must be in folder path to run our code.
- For the Glove Method
  - The dataset used for the GloVe method was the *20 newsgroups* dataset. This dataset is available here: http://qwone.com/~jason/20Newsgroups/ . However, the dataset consists of 20000 newsgroup documents and therefore, more words than our resources could handle. We decided to use a train dataset which can be found here: https://github.com/dheeraj7596/GWBoWV/tree/master/20news/data as *'train_v2.tsv'*
  - The code used to obtain the co-occurrence matrix was sourced from: https://rare-technologies.com/making-sense-of-word2vec/ . We made the following changes to convert the matrix into its dense format and save the matrix in a .mat file.

```
model_glove = glove.Glove(no_components=30, learning_rate=0.05)
model_glove.fit(cooccur.matrix, epochs=1)
final_mat = (cooccur.matrix.todense())

test = filtered_wiki()
for val in word2id:
    print(val)
```

  - To re-embed the co-occurrence matrix into word vectors, we used the following repository: https://github.com/piskvorky/word_embeddings . We made following changes to the file *run_embed.py:*
    - Reduce the feature dimension from 600 to 30.(Can be kept higher for better results if you have more computational power)
    - Reduce the number for words to 6000 (computational efficiency)

- Instead of saving the word embeddings in ***pickle*** format, we chose to write a separate python script to save it in ***.mat*** format which makes performing operations on it simpler.

```
model.word2id = dict((w, v.index) for w, v in model.wv.vocab.iteritems())
            model.id2word = utils.revdict(model.word2id)
            model.word_vectors = model.wv.syn0norm
        wordvecs = model.word_vectors
        words = []
        words[:] = model.wv.vocab
        sio.savemat('model_part.mat', {'wordvecs_part': wordvecs, 'words_part':
words})
```

- The folder for the code contains the co-occurrence matrix in the *coo_matrix.mat* file. The words corresponding to the co-occurrence matrix are present in the same folder as *final_6kwords.mat*.