Bias in Large Language Models

POOMRAPEE CHUTHAMSATID V00942601

PCHUTHAMSATID@UVIC.CA

Introduction

Keys:

- Our study examine 5 models, including OpenAI, Cohere, Google, Microsoft, and Beijing Academy of Artificial Intelligence (BGE), with 2 different classes: Gender and Race.

Purpose:

- Examine bias in word embeddings and their impact on real-world applications to guide the development of mitigation strategies in natural language processing.
- Pave the way for future comprehensive analyses of other demographic axes across different word embedding models

Extension:

- Broaden the scope of word embedding model analysis to five modern word embedding models—OpenAI, Cohere, Google, Microsoft E5, and BGE with two more gender biases but also extend the analysis of racial biases.
- Go beyond theoretical analysis to investigate how word embedding biases in word embeddings manifest in real-world contexts.

Questions

Questions

1. How modern English word embedding models differ in terms of the frequency of gender- and race-associated words

2. What the differences in bias by frequency range and effect size across embedding models

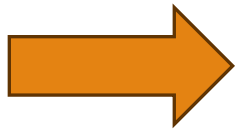
3. What the semantic categories of gender- and race-associated attributes

4. How word embeddings reflect biases present in the tech industry and higher education

Approach: Filter Most Freq-Used Words

most_k_frequency_with_filter.py

```python
# get a list of word to exclude
nltk.download('stopwords')
stop_word_list = stopwords.words('english')
punctuation_list = list(string.punctuation)
number_pattern = re.compile(r'\d')  # Regex pattern to match any digit

def readFrom(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        return file.read()

def is_valid_word(word):
    return word.isalpha() and len(word) > 2 and not number_pattern.search(word)

def process(inputFilename, outputFilename, k, stimuli):
    total = 0
    content = readFrom(inputFilename)

    lines = content.split("\n")

    skip_list = set(stop_word_list + punctuation_list + stimuli)
    print(skip_list)

    with open(outputFilename, 'w', encoding='utf-8') as file:
        # add stimuli seperately
        for line in lines:
            if line:
                word = line.split()[0]
                if word in set(stimuli):
                    total += 1
                    file.write(line + "\n")
```



glove_100000_most_freq_skip.txt

```
he 0.085181 0.50892 -0.08828 -0.39785 0.25251 0.057932 -0.15804 -0.40127 -0.13023 3.9609 -0.1579 0.40404 -0.29674 -0.17273 -0.42699 0.20168 0.16489 -0.027982 0.076
his 0.019097 0.23119 -0.1681 0.1299 0.34298 0.054346 -0.193 -0.39108 -0.11816 3.6969 -0.1359 0.48132 -0.31348 -0.086658 -0.047411 0.29327 0.081209 -0.01393 -0.0473
her 0.14203 0.095083 -0.2681 0.64799 0.1003 -0.10372 -0.18631 -0.39495 -0.27631 3.5805 -0.22571 0.40361 -0.20724 -0.59845 -0.12511 0.35779 -0.10136 0.33431 -0.3168
she 0.23648 0.39091 -0.095802 0.0536 -0.039457 0.094117 -0.12456 -0.50161 -0.14399 3.7937 -0.28044 0.37629 -0.29771 -0.56942 -0.36474 0.22409 -0.063741 0.21193 -0.
him -0.22306 0.15456 -0.32132 0.052292 0.28417 -0.29661 -0.32371 -0.36878 -0.25365 3.7652 -0.11085 0.40548 -0.25184 -0.2527 -0.42596 0.10629 0.052355 -0.14333 -0.0
man -0.1731 0.20663 0.016543 -0.31026 0.019719 0.27791 0.12283 -0.26328 0.12522 3.1894 -0.16291 -0.088759 0.0033067 -0.0029483 -0.34398 0.12779 -0.094536 0.43467 0
black -0.29365 -0.049916 0.096439 -0.089388 0.27109 0.057496 -0.50298 0.11331 -0.19913 1.0869 -0.36474 0.18028 -0.2439 -0.84879 -0.09803 0.22358 0.16649 1.8263 -0.
white -0.39347 -0.061407 0.015231 -0.21578 0.1796 -0.3057 -0.48646 0.0028348 -0.37445 1.2249 -0.20502 0.011611 -0.27524 -0.33221 0.00032765 -0.096056 0.19417 2.017
girl -0.26909 0.25307 -0.57593 0.16235 0.16094 -0.19802 -0.028971 -0.25352 -0.074811 2.1331 -0.4646 0.056153 -0.50651 -0.44885 -0.47379 0.44561 -0.12656 0.6442 0.1
woman 0.025567 0.27885 -0.16992 0.27348 -0.054906 0.26873 0.15479 -0.22401 0.26404 3.2573 -0.36653 0.14961 -0.2687 -0.31193 -0.50554 -0.018622 -0.53002 0.62383 0.2
son -0.10434 -0.050899 -0.57032 0.17441 0.3706 -0.61087 0.44934 -1.0064 0.11231 3.1026 -0.52898 0.93475 0.037713 0.37432 -0.97803 -0.36663 -0.090478 -0.20744 0.101
daughter -0.079268 0.18933 -0.45242 0.37298 0.2592 -0.39937 0.36279 -0.87065 -0.019528 3.1104 -0.36732 0.90774 -0.18414 0.27097 -0.97081 -0.24073 -0.27872 -0.39325
boy -0.27045 0.071814 -0.37339 0.064434 0.20281 -0.19224 -0.044159 -0.50268 0.057192 2.4062 -0.4266 -0.012857 -0.36779 -0.26895 -0.70937 0.32086 0.2402 0.20586 0.3
female -0.19381 0.12637 -0.10258 0.33775 0.31058 -0.051128 0.26628 0.12427 0.31396 2.0901 -0.33471 0.051985 -0.60773 0.147 0.18987 -0.0076918 -0.1114 0.89033 -0.43
brother -0.44628 -0.041053 -0.34353 0.039274 0.47159 -0.3518 0.40668 -0.77796 0.11512 3.0107 -0.016041 0.32272 -0.0087193 -0.078609 -0.5524 0.01316 0.041938 -0.363
male -0.42168 0.040278 -0.11157 0.24424 0.32563 0.025596 0.42151 0.015799 0.36034 1.9198 -0.409 0.174 -0.63391 0.1902 -0.10579 0.039374 0.082213 1.1813 -0.22162 0.
sister 0.003619 -0.13173 -0.47121 0.24497 0.2598 -0.49503 0.53001 -0.63803 -0.071046 2.7561 -0.30906 0.54085 0.032817 -0.35047 -0.4183 0.25369 -0.20419 -0.32522 -0
brown -0.37412 -0.076264 0.10926 0.18662 0.029943 0.1827 -0.63198 0.13306 -0.12898 0.60343 -0.68041 -0.14222 -0.13357 -0.65943 0.052402 0.16745 0.63923 1.768 0.346
asian -0.55917 -0.21003 -0.048223 -0.051157 0.57763 -0.09227 -0.18157 -0.3213 0.060549 0.98301 -0.65208 -0.43544 0.16574 -0.9318 0.08009 0.32869 0.027882 1.1092 0.
american -0.67575 0.29117 0.38272 -0.17098 0.23596 -0.0083299 -0.12957 0.38682 -0.02192 1.4254 -0.72196 -0.321 -0.32117 -0.10871 0.18662 -0.29415 -0.16799 1.4212 0
french -0.29123 -0.18325 -0.35079 -0.14739 0.42011 -0.10591 0.34352 -0.090096 -0.0090257 0.5615 -0.48423 0.1861 -0.18133 -0.3882 -0.0023672 0.12097 -0.12708 1.5471
indian -0.43731 -0.14039 0.42151 -0.014113 0.42082 0.21327 0.20093 -0.33553 0.20936 0.69287 -1.203 -0.11667 -0.1566 -0.65362 0.20088 0.28228 -0.088959 1.1246 0.481
chinese -0.47223 0.055899 -0.31643 -0.4299 0.34976 0.15966 0.55139 -0.52871 0.3236 0.46412 -0.99371 -0.22582 -0.29548 -0.26355 -0.12964 0.14336 -0.22207 1.5319 -0.
japanese -0.48385 -0.22614 -0.20815 -0.18392 0.17217 -0.54053 -0.0014331 -0.098388 -0.38747 0.37382 -0.76111 -0.59807 -0.12819 -0.4746 -0.11626 0.012716 -0.085529
german -0.55301 0.16403 -0.14422 0.53399 -0.073896 -0.1485 0.12619 0.25966 -0.3053 0.4373 -0.83073 -0.066401 -0.21672 -0.42964 0.46147 0.40498 0.52043 1.3168 0.371
hers 0.025954 -0.7399 -0.31839 -0.078759 -0.10531 -0.12984 0.10397 -0.37204 -0.014861 2.0009 0.062099 0.56013 -0.20553 -0.52624 -0.57305 0.10852 0.002467 0.49173 -0
canadian -0.50369 -0.16992 -0.14775 -0.35665 0.39389 -0.022334 0.1261 -0.5295 0.1934 1.2572 -0.81787 0.25549 -0.40385 0.010357 0.66717 -0.24711 0.028154 1.5046 0.1
italian -0.30558 -0.24047 0.0081653 0.22882 0.58773 0.42302 0.47828 -0.11189 -0.12728 0.43914 -0.98992 0.37399 -0.32187 -0.16554 0.16112 -0.11584 -0.034311 1.581 0
british -0.27106 -0.037942 0.30535 -0.24859 -0.084953 -0.79407 -0.14309 -0.36602 -0.099947 0.99531 -0.88803 -0.20284 -0.26612 -0.034553 0.32619 -0.11443 -0.081205
european -0.54638 -0.15723 0.13512 -0.086275 0.52106 -0.46632 0.29003 0.13932 -0.35838 0.72843 -0.99047 -0.26103 0.10687 -0.079908 0.60491 -0.017599 -0.22258 1.223
african -0.409 0.23529 0.14136 -0.38778 0.19759 0.25094 0.00092935 0.31991 0.060913 0.6675 -0.85247 -0.021978 0.054843 -0.070075 0.0033392 0.023436 -0.3393 0.93352
thai -0.20915 0.090433 0.25395 0.26853 0.44573 0.27143 0.15325 -0.7186 0.065312 0.40761 -0.96886 0.078316 0.16007 -0.84179 0.21797 0.39702 0.11227 0.93537 0.067018
korean -0.11477 -0.18642 0.079947 0.094883 0.5213 -0.21756 0.53518 -0.305377 0.22565 1.791 -0.97592 -0.28858 0.11397 -0.63732 -0.16155 -0.26343 0.17286 1.2283 0.133
australian -0.11564 0.27677 0.412 -0.27255 0.36862 -0.1216 0.20164 0.052145 0.10225 0.59862 -0.67828 0.27792 -0.17983 -0.22156 0.31856 -0.033137 -0.37738 0.76524 0
egyptian 0.3404 -0.53233 -0.031703 -0.31297 -0.26512 -0.21159 0.25976 -0.2052 -0.020412 -0.076531 -0.2961 -0.44055 0.24851 0.5444 0.35031 0.18908 -0.20528 0.59241
filipino 0.22998 -0.13363 0.3077 0.22203 0.54962 0.20655 0.77081 -0.28372 0.2576 0.29597 -0.5432 0.29011 0.28705 0.081851 0.15303 0.42065 0.16642 0.38656 0.21453 0
pakistani -0.19035 -0.28643 0.40119 0.3571 0.17854 -0.11601 0.12311 -0.73971 0.41697 -0.041893 -0.83325 0.12684 -0.56496 -0.4273 -0.075705 0.17992 -0.18339 -0.0693
caucasian -0.32428 -0.509 -0.20591 -0.32627 0.32479 -0.18753 -0.14088 0.32481 0.035556 0.20892 -0.12423 0.018874 -0.35108 0.038469 0.43195 0.49879 0.21147 0.2304 0
indonesian -0.024339 -0.1112 0.12712 0.72982 0.1779 0.075746 0.71562 -0.66638 0.098926 -0.32699 -0.9263 0.17699 0.2168 -0.10783 -0.23638 0.36418 -0.034235 0.31501
jamaican 0.10717 -0.19394 0.38236 -0.74854 0.62768 0.41055 0.47443 -0.58862 0.3591 -0.26939 -0.66523 0.30032 -0.080264 -0.28803 0.55446 0.025364 0.22286 -0.28583 0
nigerian -0.34255 -0.26712 0.56534 -0.79585 -0.1028 -0.022187 0.29885 -0.74343 -0.084781 -0.31557 -1.1801 0.34234 0.099648 0.040376 -0.18602 0.30289 -0.040836 -0.4
ethiopian 0.20097 -0.40983 0.031882 -0.10428 0.41072 0.19197 0.24432 -0.3008 0.16173 -0.43912 -0.73485 0.71918 0.13472 -0.054382 -0.1724 0.29412 0.32791 -0.3984 0.
haitian -0.30321 -0.13989 0.23919 -0.14068 0.56817 0.079074 0.32851 0.038427 0.46606 -0.73284 -0.424 -0.21787 0.16938 -0.38643 -0.44891 0.0090003 -0.40979 -0.36155
```

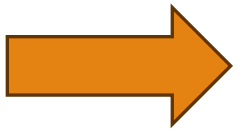
Filter Out By - glove_100000_most_freq_skip.txt:

1. Number
2. Special character
3. Character length of less than 2
4. Stop words and punctuations
5. Append Stimuli (At least 8 per each attribute)

Approach: Extract English Words Only

WordExtractor.py

```python
'''
WordExtractor:
Extract only english words from word embedding files to a new txt file
'''

# Extract data from file
def readFrom(filename: str):
    with open(filename, 'r', encoding='utf-8') as file:
        return file.read()

def writeTo(filename: str, lines: str):
    lines = content.split("\n")
    with open(filename, 'w', encoding='utf-8') as file:
        for line in lines:
            if line:
                word = line.split()[0]
                # if word in word_list:
                file.write(word + "\n")

def writeToEmbeddings(filename: str, content: str):
    lines = content.split("\n")
    with open(filename, 'w', encoding='utf-8') as file:
        for line in lines:
            if line:
                word = line.split()[0]
                # if word in word_list:
                file.write(line + "\n")

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    content = readFrom("D:/Honour Thesis Data/raw/glove_100000_most_freq_skip
```



glove_english_word_100000_most_freq_skip.txt

```
he
his
her
she
him
man
black
white
girl
woman
son
daughter
boy
female
brother
male
sister
brown
asian
american
french
indian
chinese
japanese
german
hers
canadian
italian
british
european
african
thai
korean
australian
egyptian
filipino
pakistani
caucasian
indonesian
jamaican
nigerian
ethiopian
haitian
kenyan
```

Approach: Fetch Embeddings

BGE.py

```python
if __name__ == '__main__':
    STEP = 5000
    word_list = get_word_list(WORDS_FILE)
    all_results = []

    for i in range(0, len(word_list), STEP):
        chunk = word_list[i:i + STEP]

        embeddings = model.encode(chunk,
                                  batch_size=12,
                                  max_length=512,
                                  )['dense_vecs']

        all_results.extend(embeddings)
        print(f"completed: ", i)

    writeTo("D:/Honour_Thesis_Data/BGE/BGE_100000_most_freq_skip.txt", all_results)

    print("Finish most_freq")
```

BGE

MicrosoftEmbeddings.py

```python
if __name__ == '__main__':
    STEP = 5000
    word_list = get_word_list(WORDS_FILE)
    all_results = []

    for i in range(0, len(word_list), STEP):
        chunk = word_list[i:i + STEP]

        chunk = tokenizer(chunk, return_tensors='pt', padding=True, truncation=True)

        # Get the embeddings
        with torch.no_grad():
            outputs = model(**chunk)
            embeddings = outputs.last_hidden_state

        word_embeddings = embeddings.mean(dim=1)

        word_embeddings = [word_embedding.tolist() for word_embedding in word_embeddings]
        all_results.extend(word_embeddings)

        print(f"completed: ", i)

    writeTo("D:/Honour_Thesis_Data/microsoft/microsoft_100000_most_freq_skip.txt", all_results)

    print("Finish most_freq")
```

Microsoft

CohereEmbeddings.py

```python
if __name__ == '__main__':
    STEP = 2500
    model = "embed-english-v3.0"
    input_type = "clustering"
    word_list = get_word_list(WORDS_FILE)

    for i in range(0, len(word_list), STEP):
        chunk = word_list[i:i + STEP]

        response = co.embed(
            model=model,
            texts=chunk,
            input_type=input_type,
            embedding_types=['float']
        ).embeddings

        print(f"completed: ", i)

        writeTo(f"temp/cohere_embeddings/cohere_chunk_skip_{i}.txt", response.float_, chunk)

        time.sleep(65)
        print("Resumed after 65 sec to avoid the request limitation")

    # Write all results to a file once all chunks are processed
    input_directory = "temp/cohere_embeddings"
    output_filename = "D:/Honour_Thesis_Data/cohere/cohere_100000_most_freq_skip.txt"
    num_files = 40
    concatenateFiles(input_directory, output_filename, num_files)

    print("Finish most_freq")
```

Cohere

GoogleEmbeddings.py

```python
if __name__ == "__main__":
    STEP = 250
    word_list = get_word_list(WORDS_FILE)

    for i in range(0, len(word_list), STEP):
        all_results = []
        chunk = word_list[i:i + STEP]

        """Embeds texts with a pre-trained, foundational model."""
        model = TextEmbeddingModel.from_pretrained(MODEL)
        inputs = [TextEmbeddingInput(text, TASK) for text in chunk]
        kwargs = dict(output_dimensionality=DIMENSION) if DIMENSION else {}
        embeddings = model.get_embeddings(inputs, **kwargs)
        [all_results.append(embedding.values) for embedding in embeddings]

        print(f"completed: ", i)

        writeTo(f"temp/google_embeddings/google_chunk_skip_{i}.txt", all_results, chunk)

        time.sleep(15)
        print("Resumed after 15 sec to avoid the request limitation")

    # Write all results to a file once all chunks are processed
    input_directory = "temp/google_embeddings"
    output_filename = "D:/Honour_Thesis_Data/google/google_100000_most_freq_skip.txt"
    num_files = 400
    concatenateFiles(input_directory, output_filename, num_files)

    print("Finish most_freq")
```

Google

OpenAIEmbeddings.py

```python
if __name__ == '__main__':
    STEP = 2000
    word_list = get_word_list(WORDS_FILE)

    all_results = []

    for i in range(0, len(word_list), STEP):
        chunk = word_list[i:i + STEP]

        data = {
            "model": "text-embedding-3-small",
            "input": chunk
        }

        headers = {
            "Content-Type": "application/json",
            "Authorization": f"Bearer {api_key}"
        }

        # Make the POST request
        response = requests.post(url, json=data, headers=headers)

        if response.status_code == 200:
            all_results.extend(response.json()["data"])
        else:
            print(f"Error with chunk {i // STEP + 1}: {response.text}")
            break

    # Write all results to a file once all chunks are processed
    writeTo("D:/Honour_Thesis_Data/openAI/openAI_100000_most_freq_skip.txt", all_results)
```

OpenAI

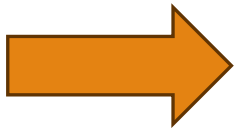
Approach: Compute Gender/Race SC-WEAT

race_SC_WEST_collector.py

```python
import numpy as np
from scipy.stats import norm
import pandas as pd
from os import path

def SC_WEAT(w, A, B, permutations):
    w_normed = w / np.linalg.norm(w)
    A_normed = A / np.linalg.norm(A,axis=-1,keepdims=True)
    B_normed = B / np.linalg.norm(B,axis=-1,keepdims=True)

    A_associations = w_normed @ A_normed.T
    B_associations = w_normed @ B_normed.T
    joint_associations = np.concatenate((A_associations,B_associations),axis=-1)

    test_statistic = np.mean(A_associations) - np.mean(B_associations)
    effect_size = test_statistic / np.std(joint_associations,ddof=1)

    midpoint = len(A)
    sample_distribution = np.array([np.random.permutation(joint_associations) for _ in range(permutations)])
    sample_associations = np.mean(sample_distribution[:,:midpoint],axis=1) - np.mean(sample_distribution[:,midpoint:],
    p_value = 1 - norm.cdf(test_statistic,np.mean(sample_associations),np.std(sample_associations,ddof=1))

    return effect_size, p_value

PERMUTATIONS = 10000
```

BGE_100000_most_freq_skip.txt

```
he -0.009843058 0.011912547 -0.010630585 -0.020679068 0.025292762 -0.022202954 0.00014465557 0.057851877 -0.033272598 -0.016945597 -0.033827852 0.03610311 -0.00155
his -0.024366362 0.023636088 -0.0055576004 0.0033114133 0.00864542 -0.009621155 0.015440252 0.04251365 -0.02576513 -0.001651442 -0.007911447 0.042256474 -0.0222829
her -0.05085539 0.023744775 -0.009161204 0.017153643 -0.00069165137 0.0007735467 -0.008374233 0.048029725 -0.024087137 -0.019253239 0.016731335 0.02756961 -0.00665
she -0.023164514 0.016622378 -0.0142149115 -0.0014279875 0.016335534 0.0051120953 -0.011759172 0.059267152 -0.022265147 -0.023362957 0.0066464767 0.02726629 -0.004
him -0.015600515 0.027411196 0.02332834 -0.004257651 0.0022501058 -0.01900632 0.008135417 0.033999473 -0.035292357 -0.004135884 -0.022481354 0.048481517 -0.0231318
man -0.0069698426 0.037779402 -0.029933427 -0.03144383 0.00247777 0.00540482 0.01425602 0.09212364 -0.024075728 -0.0292885 -0.024802836 0.012189134 -0.029989189 0.
black 0.010044252 0.24086675e-05 -0.0076834615 -0.0115142872 -0.0163269 0.01024806 0.0030273637 0.017571962 -0.002019227 0.017741196 0.026403569 0.022587886 -0.02183
white -0.004495323 0.028197085 -0.019683376 -0.0028184059 -0.007044916 0.013664524 -0.03520654 0.05272645 -0.0005161461 0.023103366 0.03710599 0.05936292 -0.00858
girl -0.05559032 0.009807249 -0.027140541 -0.014373314 0.0010266646 0.008719412 -0.031525023 0.060787547 -0.032244436 -0.0141315395 0.034851484 0.026633933 0.01157
woman -0.018655738 0.022792663 -0.037602548 -0.014867334 -0.02302179 0.020201363 -0.017357226 0.083113775 -0.007001244 -0.04789601 0.023769349 0.03730692 0.0039586
son 0.009027074 0.032651268 -0.023786629 -0.018022874 -0.0067714746 -0.026581258 -0.0005390688 0.08102337 0.0049292627 -0.044939443 0.036230065 0.04764845 -0.05269
daughter -0.010306914 0.033900585 -0.02870858 0.007041408 -0.0557733 0.035465546 0.001198934 0.0735478 0.001173639 -0.034910258 0.051958818 0.026153868 -0.01829657
boy -0.0115512945 0.027859367 -0.003315251 -0.0070938114 0.0064185136 0.0064308876 0.008242706 0.05571794 -0.011902489 -0.0145111213 -0.00046763723 0.057610437 0.00
female -0.05721689 0.01182378 -0.04179501 -0.01603585 -0.0138873765 0.010024216 -0.01841371 0.08027002 -0.0034275 -0.035150006 0.022998143 0.009344744 -0.009586388
brother -0.001895327 0.049692504 -0.00488821075 0.015874244 -0.018604001 -0.027376195 0.031364724 0.05249363 0.032344133 -0.047050416 -0.0127108002 0.01445697 25 -0.
male -0.006540634 0.04430576 -0.03205376 5 -0.013717482 0.008434906 0.013156017 0.004328107 0.06613994 -0.022307856 -0.0172709 -0.053621005 0.016481867 -0.03508684
sister -0.01019742 0.05690981 -0.0038518924 0.024037993 -0.009773784 -0.005696313 0.010984503 0.08801713 0.023637272 -0.032235168 0.022172892 0.010880737 -0.022966
brown 0.022222469 0.049672466 -0.0043864283 -0.0063141435 -0.064200655 -0.041201536 -0.015341946 0.0146073615 -0.005273176 -0.018320968 -0.020639075 0.02977809 0.0
asian 0.038221836 -0.032542355 -0.021036172 0.015903793 -0.041798875 0.0024744489 -0.019971514 0.034296278 -0.032275442 0.002160117 0.047064632 0.028770417 0.00143
american 0.022404581 0.0031304155 -0.024398452 0.019377327 -0.015010053 0.008980044 0.016551673 0.06696 -0.006126401 -0.00042978412 0.025242051 0.022386028 0.01827
french -0.014102734 0.03077161 -0.033833656 -0.0087327 -0.0058310195 0.025382932 0.03824311 0.027486967 -0.0013469943 -0.008762327 -0.01770902 0.03403527 0.0106680
indian 0.015264683 0.028347842 -0.005283056 0.020942923 -0.044722177 -0.001915568 -0.017999595 0.041125357 -0.036066573 -0.032127097 0.018236635 0.006371234 -0.010
chinese -0.015682943 0.02080758 -0.03729429 0.003597267 -0.012521127 0.04440437 0.0058401157 0.076024294 -0.025364242 -0.00994091 0.004291999 0.04920967 0.022701
japanese 0.003671518 -0.0067323926 -0.015457888 0.022293553 -0.01394926 0.020033065 0.024532266 0.040033378 -0.015675684 -0.031324945 0.008586337 0.03957112 -0.017
german -0.03717758 0.012503683 -0.009570018 -0.020475548 -0.027941812 0.0062041353 0.009431545 0.03982228 -0.001678796 -0.0015954392 0.009235592 0.02425509 -0.0496
hers -0.04841891 0.016828965 -0.016982354 0.040736474 -0.009714688 0.0009916571 -0.020122563 0.017441437 -0.022343185 -0.023785252 -0.012761632 0.032535326 -0.0119
canadian 0.0005883841 0.021475319 -0.028990641 -0.046849333 0.014656802 0.0018865735 0.026975313 0.0562077 0.00874665 0.005573923 0.018331569 0.040721424 0.0456998
italian -0.0025208113 0.028438076 -0.007321941 0.032402195 -0.020876724 0.00573845 95 -0.012363688 0.05113818 -0.02759093 -0.028303513 0.015624384 0.052583534 -0.005
british -0.013089506 0.034533456 -0.008183056 -0.037507914 -0.0131961545 -0.007943909 0.017462412 0.018476836 -0.008638039 -0.030507565 0.0074860677 0.04257792 0.0
```



AutoSave Off — BGE_gend

File Home Insert Page Layout Formulas D

Paste — Tahoma 11 — Clipboard — Font

A1 — word

	A	B	C	D
1	word	female_effect_size	p_value	
2	he	-0.943677605	0.970500617	
3	his	-0.909747302	0.965617251	
4	her	0.915395145	0.033609327	
5	she	0.947423343	0.029639476	
6	him	-0.867682702	0.960340342	
7	man	-1.090637079	0.986541925	
8	black	-0.781083833	0.93772196	
9	white	-0.093449019	0.576187764	
10	girl	1.26733101	0.006045478	
11	woman	1.397305849	0.002559069	
12	son	-1.070543755	0.983709045	
13	daughter	1.272994386	0.005618401	
14	boy	-1.028133197	0.980950154	
15	female	1.450262242	0.001777771	
16	brother	-0.752125782	0.935435309	
17	male	-0.934933936	0.970373242	
18	sister	0.87881339	0.036901378	
19	brown	-1.008360691	0.977386787	
20	asian	-0.380471963	0.78570984	
21	american	-0.677240924	0.909893783	
22	french	-0.464742735	0.821419611	
23	indian	-0.630329924	0.892628966	
24	chinese	0.178253665	0.36218094	
25	japanese	-0.601035724	0.883621357	
26	german	-0.032106458	0.52187447	
27	hers	1.076191584	0.015081363	
28	canadian	-0.378910441	0.773015185	
29	italian	-0.523560812	0.852101726	
30	british	-0.660623805	0.907858355	

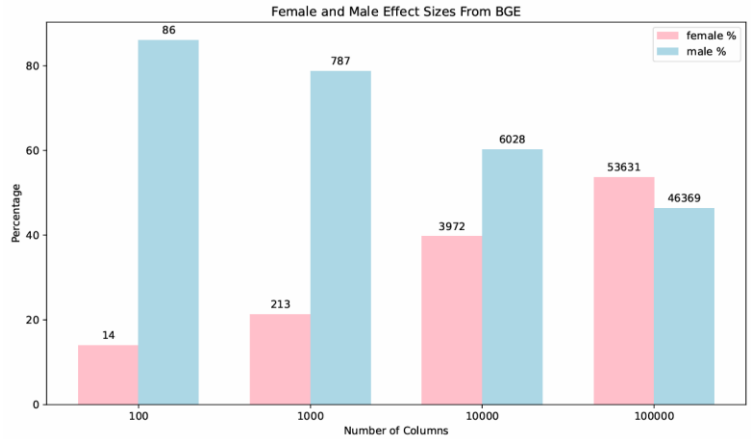
BGE_gender_100000_most_frequenc

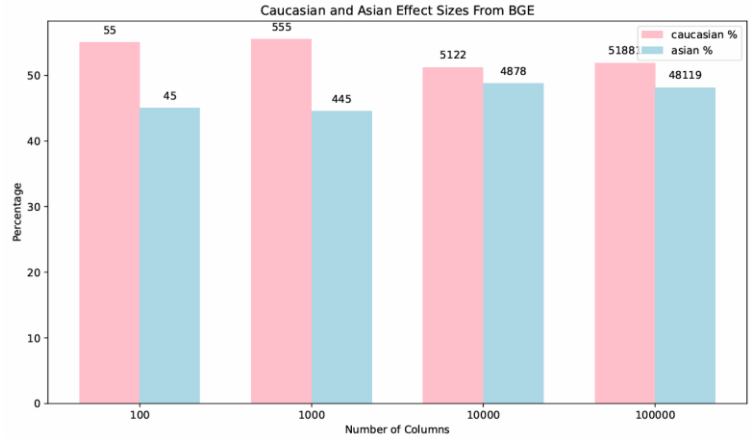
Approach: Find Freq Word Associations

association_most_frequency_words_plot.py

```python
def plot(input, output, num_columns_list, name, groups):
    print(f"\n == {input} == \n")
    data = pd.read_csv(input)

    first_group_percentages = []
    second_group_percentages = []
    first_group_counts = []
    second_group_counts = []

    for num_columns in num_columns_list:
        n_data = data[:num_columns]
        # Filter rows where {first_group}_effect_size is positive and {second_group}_effect_size is negative
        first_group_data = n_data[n_data[f'{groups[0]}_effect_size'] > 0]
        second_group_data = n_data[n_data[f'{groups[0]}_effect_size'] < 0]

        # Count the number of occurrences for each word
        total_count = len(first_group_data) + len(second_group_data)
        first_group_count = len(first_group_data)
        second_group_count = len(second_group_data)

        print("Number of N:", num_columns)
        print("Total:", total_count)
        print(f"{groups[0].title()}:", first_group_count)
        print(f"{groups[1].title()}:", second_group_count)
```

	A	B	C	D
1	word	female_effect_size	p_value	
2	he	-0.943677605	0.970500617	
3	his	-0.909747302	0.965617251	
4	her	0.915395145	0.033609327	
5	she	0.947423343	0.029639476	
6	him	-0.867682702	0.960340342	
7	man	-1.090637079	0.986541925	
8	black	-0.781083833	0.93772196	
9	white	-0.093449019	0.576187764	
10	girl	1.26733101	0.006045478	
11	woman	1.397305849	0.002559069	
12	son	-1.070543755	0.983709045	
13	daughter	1.272994386	0.005618401	
14	boy	-1.028133197	0.980950154	
15	female	1.450262242	0.001777771	
16	brother	-0.752125782	0.935435309	
17	male	-0.934933936	0.970373242	
18	sister	0.87881339	0.036901378	





Approach: Find Freq Range and Effect Size

frequency_range_and_effect_size_plot.py

```python
def plot(input, output, num_columns_list, groups):
    print(f"\n == {input} == \n")
    data = pd.read_csv(input, na_values=None, keep_default_na=False)

    effect_size_floors = [0, .2, .5, .8]

    # for 100, 1000, 10000 N-words
    es_list = []

    for ceiling in num_columns_list:
        head_df = data.head(ceiling)
        ceiling_counts = [ceiling]

        for es in effect_size_floors:
            es_df = head_df.loc[head_df[f'{groups[0]}_effect_size'] >= es]
            es_quantity = len(es_df.index.tolist())
            ceiling_counts.append(es_quantity)

        for es in effect_size_floors:
            es_df = head_df.loc[head_df[f'{groups[0]}_effect_size'] <= -es]
            es_quantity = len(es_df.index.tolist())
            ceiling_counts.append(es_quantity)
```

	A	B	C	D
1	word	female_effect_size	p_value	
2	he	-0.943677605	0.970500617	
3	his	-0.909747302	0.965617251	
4	her	0.915395145	0.033609327	
5	she	0.947423343	0.029639476	
6	him	-0.867682702	0.960340342	
7	man	-1.090637079	0.986541925	
8	black	-0.781083833	0.93772196	
9	white	-0.093449019	0.576187764	
10	girl	1.26733101	0.006045478	
11	woman	1.397305849	0.002559069	
12	son	-1.070543755	0.983709045	
13	daughter	1.272994386	0.005618401	
14	boy	-1.028133197	0.980950154	
15	female	1.450262242	0.001777771	
16	brother	-0.752125782	0.935435309	
17	male	-0.934933936	0.970373242	
18	sister	0.87881339	0.036901378	

B	C	D	E	F	G	H	I	J
num_words	female_0	female_0.2	female_0.5	female_0.8	male_0	male_0.2	male_0.5	male_0.8
100	14	11	8	8	86	80	60	30
1000	213	145	66	30	787	667	444	159
10000	3972	2904	1614	815	6028	4811	2813	1151
100000	53631	42980	28055	15825	46369	35333	20293	8998

B	C	D	E	F	G	H	I	J
num_words	caucasian_0	caucasian_0.2	caucasian_0.5	caucasian_0.8	asian_0	asian_0.2	asian_0.5	asian_0.8
100	55	26	2	0	45	20	11	0
1000	555	210	24	1	445	139	22	0
10000	5122	2519	587	94	4878	2293	487	74
100000	51881	31694	11091	2600	48119	28248	9534	2241

Approach: Plot Cluster with K-Mean & TSNE

clustering_of_words_plot.py

```python
# K-Means clustering and transformed coordinates
NUM_CLUSTERS = 11
kmeans_group1 = KMeans(n_clusters=NUM_CLUSTERS, random_state=0, algorithm='elkan', init='k-means++', max_iter=1000,
                       n_init=100).fit(target_data_group1)
kmeans_group1_transform = KMeans(n_clusters=NUM_CLUSTERS, random_state=0, algorithm='elkan', init='k-means++',
                                 max_iter=1000, n_init=100).fit_transform(target_data_group1)

kmeans_group2 = KMeans(n_clusters=NUM_CLUSTERS, random_state=0, algorithm='elkan', init='k-means++', max_iter=1000,
                       n_init=100).fit(target_data_group2)
kmeans_group2_transform = KMeans(n_clusters=NUM_CLUSTERS, random_state=0, algorithm='elkan', init='k-means++',
                                 max_iter=1000, n_init=100).fit_transform(target_data_group2)

# T-SNE coordinates
reduced_dims_group1 = TSNE().fit_transform(kmeans_group1_transform.squeeze())
tsne_df_group1 = pd.DataFrame(reduced_dims_group1, index=top_group1_words, columns=['x', 'y'])
tsne_df_group1['word'] = top_group1_words
tsne_df_group1['cluster'] = kmeans_group1.labels_
tsne_df_group1.to_csv(path.join(dir, f'tsne_clusters_{groups[0]}_over_{groups[1]}_1k_{NUM_CLUSTERS}.csv'))

tsne_group1_x = tsne_df_group1['x'].tolist()
tsne_group1_y = tsne_df_group1['y'].tolist()

reduced_dims_group2 = TSNE().fit_transform(kmeans_group2_transform.squeeze())
tsne_df_group2 = pd.DataFrame(reduced_dims_group2, index=top_group2_words, columns=['x', 'y'])
tsne_df_group2['word'] = top_group2_words
tsne_df_group2['cluster'] = kmeans_group2.labels_
tsne_df_group2.to_csv(path.join(dir, f'tsne_clusters_{groups[1]}_over_{groups[0]}_1k_{NUM_CLUSTERS}.csv'))
```

asian_over_black_clusters_11 - Notepad

File Edit Format View Help

```
Cluster 0:add, Add, added, addition, additional, extra, increase, plus
Cluster 1:agree, allow, allows, ask, asked, available, back, became, become, began, believe, bring, broug
Cluster 2:account, address, age, air, application, area, art, baby, band, base, bit, board, body, box, bc
sional, program, property, quality, range, rate, result, road, room, sales, school, screen, season, secti
Cluster 3:About, After, All, Also, And, Are, Back, Business, But, Comment, Even, First, For, From, Have,
Cluster 4:activities, books, cases, changes, children, Comments, comments, companies, conditions, costs,
Cluster 5:America, american, American, asian, australian, british, canadian, China, chinese, English, eur
Cluster 6:amazing, bad, beautiful, better, big, Big, cheap, clean, clear, cool, difficult, early, easy, e
Cluster 7:actually, almost, along, already, always, AND, another, anyone, anything, cannot, certain, comm
Cluster 8:access, action, answer, attention, bed, blood, call, care, change, check, choice, click, collec
Cluster 9:Air, Apple, Apr, April, Art, Beach, Best, Blue, Board, Book, California, Can, Car, Center, Chec
Cluster 10:according, across, ago, also, among, amount, around, article, average, based, behind, best, bl
```

Select 1,000 most frequently used and associated word

- Effect Size is equal or more than 0.5
- P-Value is equal or less than 0.05
- Select the top most 1,000 word by frequently-used

Approach: Assign Clusters' Topic

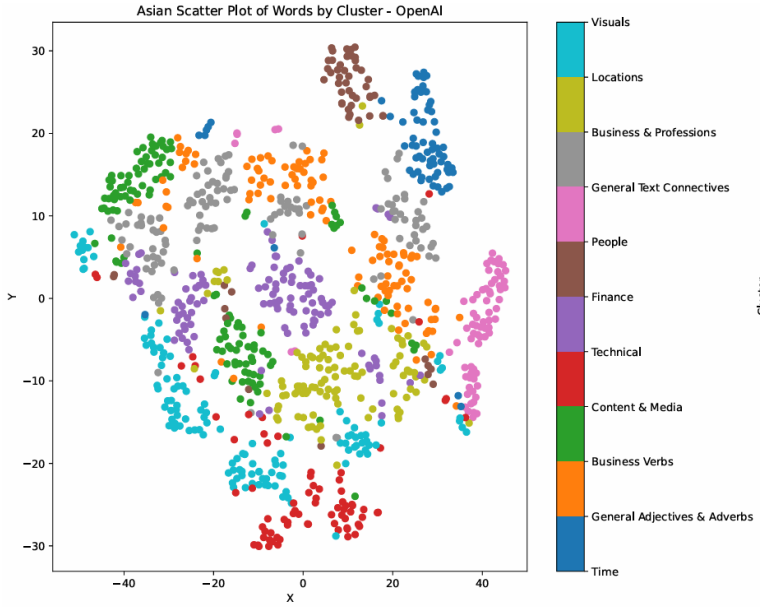
asian_over_black_clusters_11 - Notepad

File Edit Format View Help

```
Cluster 0:add, Add, added, addition, additional, extra, increase, plus
Cluster 1:agree, allow, allows, ask, asked, available, back, became, become, began, believe, bring, broug
Cluster 2:account, address, age, air, application, area, art, baby, band, base, bit, board, body, box, bc
sional, program, property, quality, range, rate, result, road, room, sales, school, screen, season, secti
Cluster 3:About, After, All, Also, And, Are, Back, Business, But, Comment, Even, First, For, From, Have,
Cluster 4:activities, books, cases, changes, children, Comments, comments, companies, conditions, costs,
Cluster 5:America, american, American, asian, australian, british, canadian, China, chinese, English, eur
Cluster 6:amazing, bad, beautiful, better, big, Big, cheap, clean, clear, cool, difficult, early, easy, e
Cluster 7:actually, almost, along, already, always, AND, another, anyone, anything, cannot, certain, comm
Cluster 8:access, action, answer, attention, bed, blood, call, care, change, check, choice, click, collec
Cluster 9:Air, Apple, Apr, April, Art, Beach, Best, Blue, Board, Book, California, Can, Car, Center, Chec
Cluster 10:according, across, ago, also, among, amount, around, article, average, based, behind, best, bl
```

AI-Generated Prompts:

- For each cluster of words, assign an appropriate unique concept such as Sports, Health and Relationships, Female Names, or Engineering and Electronics. Output in json e.g {"0":title}




```
{
  "0": "Time",
  "1": "General Adjectives & Adverbs",
  "2": "Business Verbs",
  "3": "Content & Media",
  "4": "Technical",
  "5": "Finance",
  "6": "People",
  "7": "General Text Connectives",
  "8": "Business & Professions",
  "9": "Locations",
  "10": "Visuals"
}
```

Approach: Find Big Tech Association

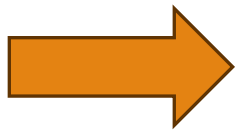
big_tech_associations.py

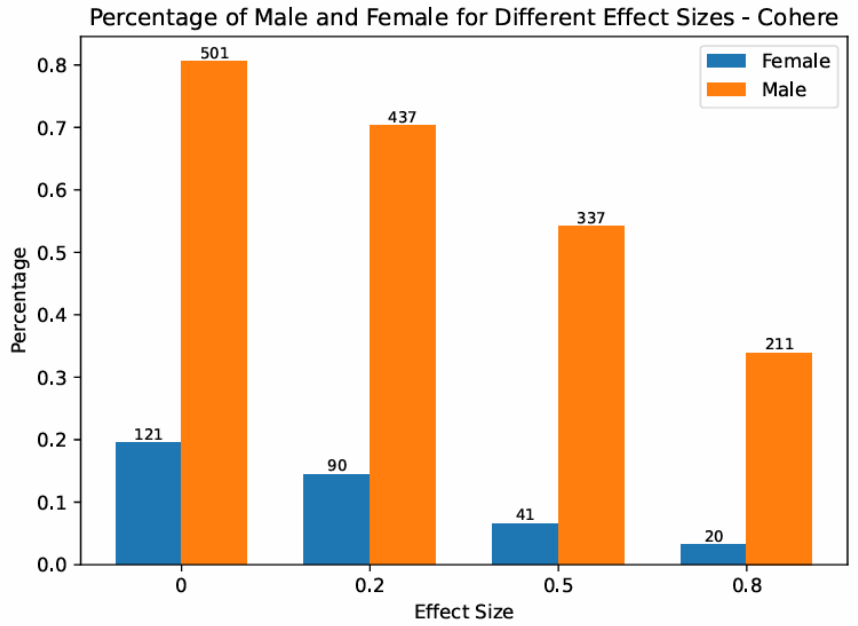
```
def process(top_100k_embeddings, result):
    print(top_100k_embeddings)
    embedding_df = pd.read_csv(top_100k_embeddings, sep=' ', header=None, index_col=0, na_values=None, keep_de

    # Get mean cosine similarities with Big Tech words

    big_tech_words = ['Google', 'Amazon', 'Facebook', 'Microsoft', 'Apple', 'Nvidia', 'Intel', 'IBM', 'Huawei'
                      'Samsung', 'Uber', 'Alibaba']

    big_tech_embs = embedding_df.loc[[word for word in big_tech_words if word in embedding_df.index]].to_numpy
    big_tech_normed = big_tech_embs / np.linalg.norm(big_tech_embs, axis=-1, keepdims=True)

    all_embs = embedding_df.to_numpy()
    all_embs_normed = all_embs / np.linalg.norm(all_embs, axis=-1, keepdims=True)

    associations = all_embs_normed @ big_tech_normed.T
    means = np.mean(associations, axis=1)

    # Write dataframe to file

    big_tech_df = pd.DataFrame(means, index=embedding_df.index.tolist(), columns=['big_tech_es'])
    largest = big_tech_df.nlargest(10000, 'big_tech_es')

    largest.to_csv(result)
```

+

big_tech_analysis.py

```
from matplotlib import pyplot as plt

def SC_WEAT(w, A, B, permutations):
    w_normed = w / np.linalg.norm(w)
    A_normed = A / np.linalg.norm(A, axis=-1, keepdims=True)
    B_normed = B / np.linalg.norm(B, axis=-1, keepdims=True)

    A_associations = w_normed @ A_normed.T
    B_associations = w_normed @ B_normed.T
    joint_associations = np.concatenate((A_associations, B_associations), axis=-1)

    test_statistic = np.mean(A_associations) - np.mean(B_associations)
    effect_size = test_statistic / np.std(joint_associations, ddof=1)

    midpoint = len(A)
    sample_distribution = np.array([np.random.permutation(joint_associations) for _ in range(permutations)])
    sample_associations = np.mean(sample_distribution[:, :midpoint], axis=1) - np.mean(sample_distribution[:, midpoint:
    p_value = 1 - norm.cdf(test_statistic, np.mean(sample_associations), np.std(sample_associations, ddof=1))

    return effect_size, p_value

def process(top_100k_embeddings, largestBigTechs, output_weats, output_bigtechs, pdf, group1_stimuli, group2_stimuli
    print(top_100k_embeddings)
    embedding_df = pd.read_csv(top_100k_embeddings, sep=' ', header=None, index_col=0, na_values=None, keep_default_
```

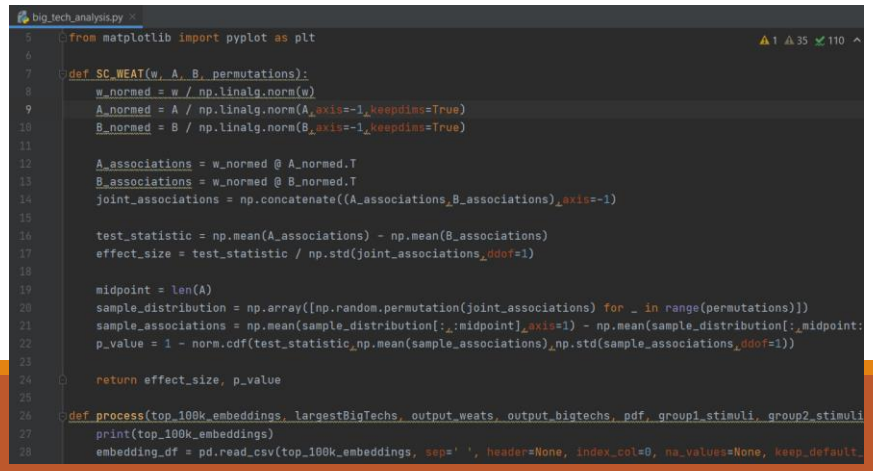




Approach: Find Higher Education Association

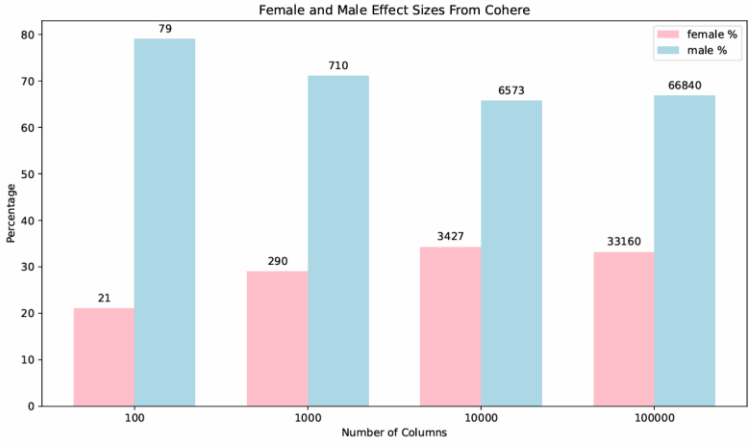
top_50_universities.txt

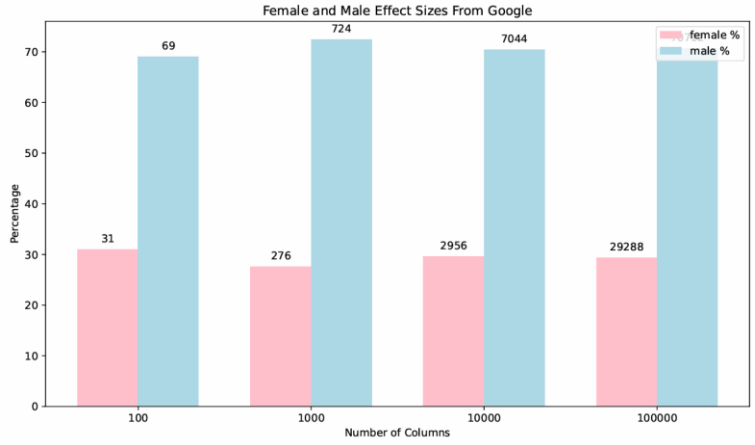
```
University of Oxford
Stanford University
Massachusetts Institute of Technology
Harvard University
University of Cambridge
Princeton University
California Institute of Technology
Imperial College London
University of California, Berkeley
Yale University
ETH Zurich
Tsinghua University
The University of Chicago
Peking University
Johns Hopkins University
University of Pennsylvania
```

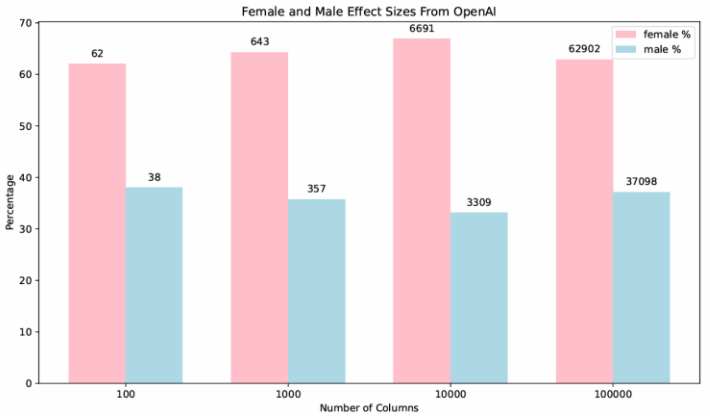


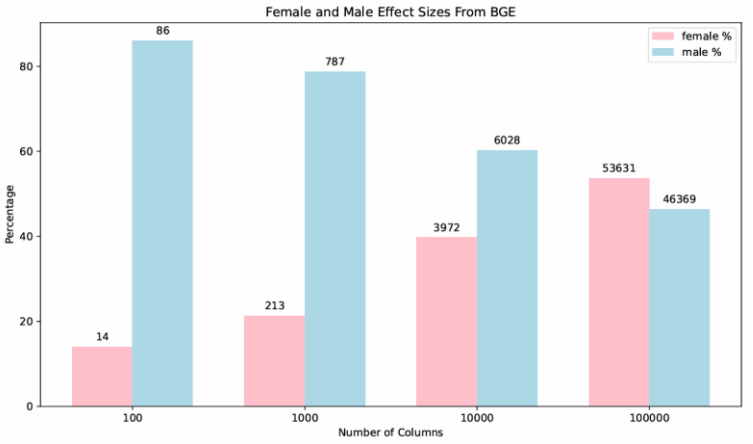
```python
from matplotlib import pyplot as plt


def SC_WEAT(w, A, B, permutations):
    w_normed = w / np.linalg.norm(w)
    A_normed = A / np.linalg.norm(A, axis=-1, keepdims=True)
    B_normed = B / np.linalg.norm(B, axis=-1, keepdims=True)

    A_associations = w_normed @ A_normed.T
    B_associations = w_normed @ B_normed.T
    joint_associations = np.concatenate((A_associations, B_associations), axis=-1)

    test_statistic = np.mean(A_associations) - np.mean(B_associations)
    effect_size = test_statistic / np.std(joint_associations, ddof=1)

    midpoint = len(A)
    sample_distribution = np.array([np.random.permutation(joint_associations) for _ in range(permutations)])
    sample_associations = np.mean(sample_distribution[:, :midpoint], axis=1) - np.mean(sample_distribution[:, midpoint:
    p_value = 1 - norm.cdf(test_statistic, np.mean(sample_associations), np.std(sample_associations, ddof=1))

    return effect_size, p_value

def process(top_100k_embeddings, largestBigTechs, output_weats, output_bigtechs, pdf, group1_stimuli, group2_stimuli
    print(top_100k_embeddings)
    embedding_df = pd.read_csv(top_100k_embeddings, sep=' ', header=None, index_col=0, na_values=None, keep_default_
```

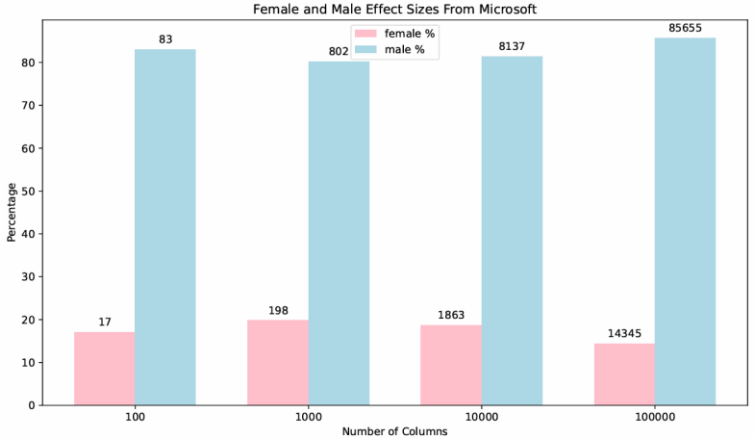
Results: Gender Freq Word Associations





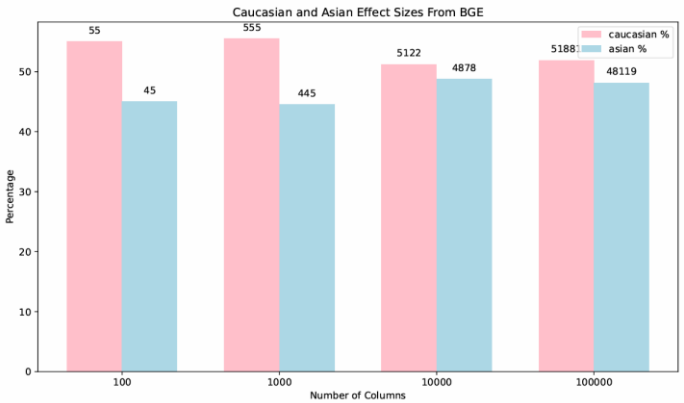


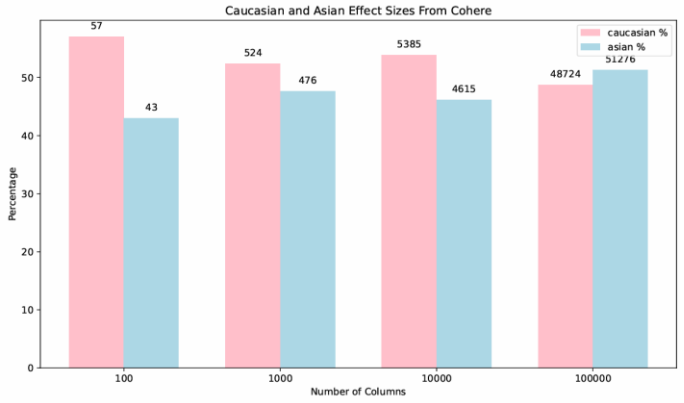


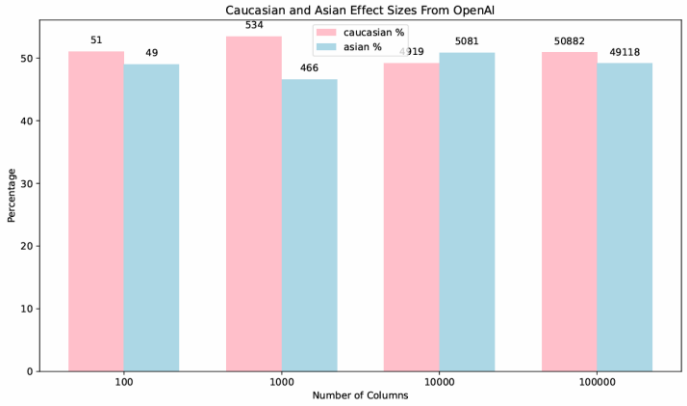


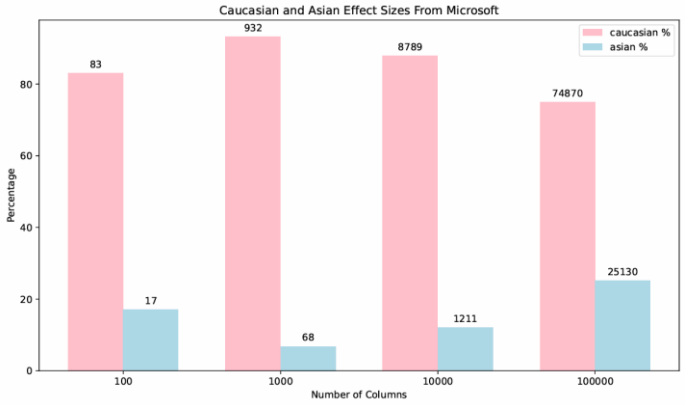
4 out of 5 models indicate a more association with males rather than females

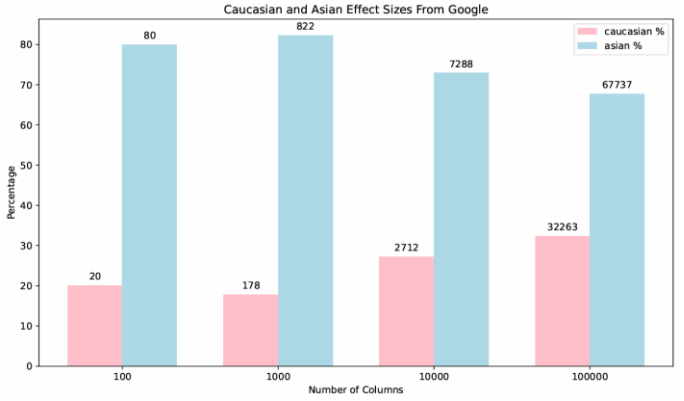
Results: Race Freq Word Associations I







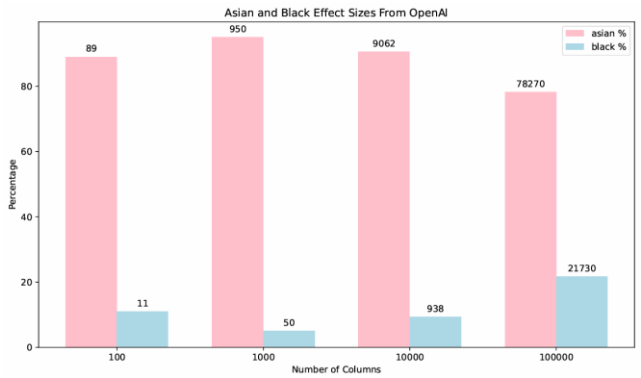


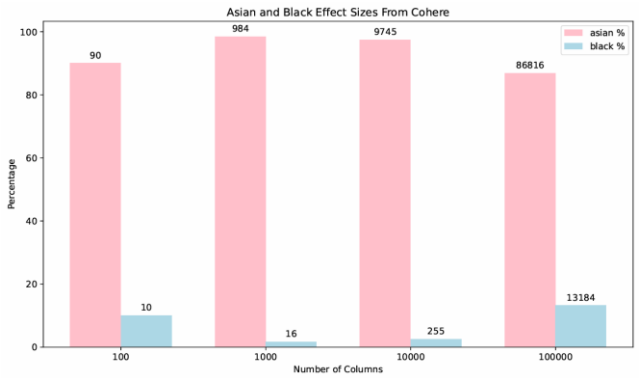


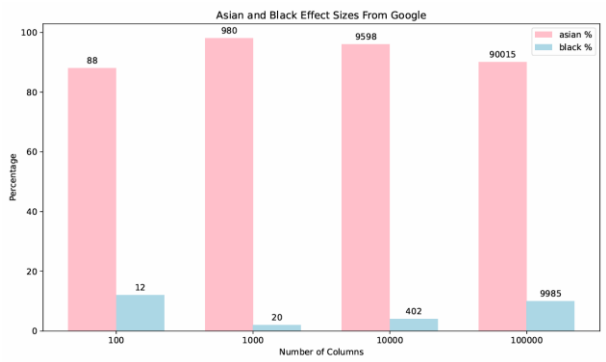
3 out of 5 models indicate an equal association between Caucasians and Asians.

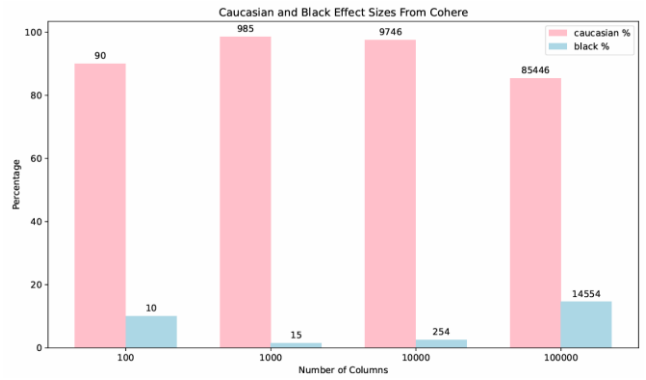
While the other two report otherwise.

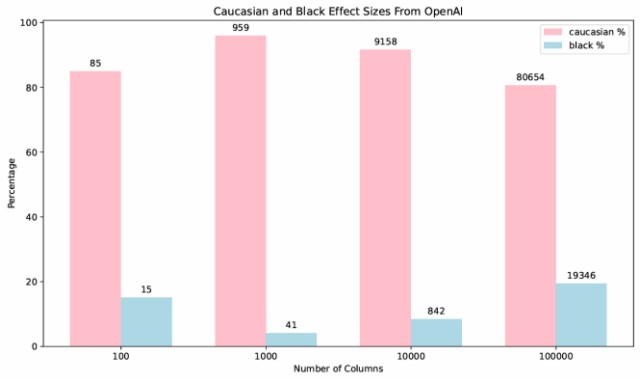
Results: Race Freq Word Associations II











All 5 models indicate that Blacks are underrepresented in every pairwise comparison

Result: Gender Freq Range and Effect Size

num_words	female_0	female_0.2	female_0.5	female_0.8	male_0	male_0.2	male_0.5	male_0.8
100	21	16	10	10	79	66	28	16
1000	290	177	86	36	710	569	289	112
10000	3427	2325	1148	539	6573	5240	3144	1545
100000	33160	24315	14378	8604	66840	56283	39141	23695

Cohere

Google

num_words	female_0	female_0.2	female_0.5	female_0.8	male_0	male_0.2	male_0.5	male_0.8
100	31	22	12	8	69	62	51	37
1000	276	182	102	49	724	633	466	307
10000	2956	2188	1271	671	7044	6140	4582	2896
100000	29288	22778	14951	9275	70712	63081	49655	34546

num_words	female_0	female_0.2	female_0.5	female_0.8	male_0	male_0.2	male_0.5	male_0.8
100	17	10	8	4	83	66	28	3
1000	198	87	36	11	802	599	276	86
10000	1863	990	396	175	8137	6536	3471	1332
100000	14345	8553	4705	2895	85655	75282	50949	26358

Microsoft

3 of out 5 models indicate a higher number of strong-associated (0.8) words with males rather than females

Result: Race Freq Range and Effect Size I

num_words	caucasian_0	caucasian_0.2	caucasian_0.5	caucasian_0.8	asian_0	asian_0.2	asian_0.5	asian_0.8
100	57	31	8	1	43	25	13	6
1000	524	258	56	6	476	211	46	13
10000	5385	3274	1060	202	4615	2643	870	212
100000	48724	32613	13457	3624	51276	35780	17645	7109

Cohere

Google

num_words	caucasian_0	caucasian_0.2	caucasian_0.5	caucasian_0.8	asian_0	asian_0.2	asian_0.5	asian_0.8
100	20	13	9	2	80	72	52	38
1000	178	109	41	8	822	724	504	267
10000	2712	1746	749	238	7288	6091	4093	2095
100000	32263	22406	11439	4485	67737	56365	37510	19465

num_words	caucasian_0	caucasian_0.2	caucasian_0.5	caucasian_0.8	asian_0	asian_0.2	asian_0.5	asian_0.8
100	51	36	16	2	49	44	27	9
1000	534	372	168	44	466	323	135	46
10000	4919	3471	1637	514	5081	3654	1769	645
100000	50882	38086	20833	7961	49118	36798	20833	9147

OpenAI

3 of out 5 models indicate a higher number of strong-associated (0.8) words with Asians rather than Caucasians

Result: Race Freq Range and Effect Size II

num_words	asian_0	asian_0.2	asian_0.5	asian_0.8	black_0	black_0.2	black_0.5	black_0.8
100	91	90	88	86	9	7	3	0
1000	990	985	973	947	10	8	3	0
10000	9868	9715	9299	8310	132	59	16	5
100000	92803	87553	75710	57440	7197	3862	1171	270

Race-Associated (Asian vs Black) by Range and Effect Size - BGE Model

num_words	caucasian_0	caucasian_0.2	caucasian_0.5	caucasian_0.8	black_0	black_0.2	black_0.5	black_0.8
100	91	90	89	83	9	8	4	0
1000	990	989	982	941	10	8	4	0
10000	9938	9837	9455	7963	62	36	15	1
100000	95832	91344	77736	52707	4168	1879	544	134

Race-Associated (Caucasian vs Black) by Range and Effect Size - BGE Model

All model indicate that Blacks always has a lower number of strong-associated words compared to other attributes (Caucasians and Asians)

These four tables are an example where Blacks are underrepresented compared to Caucasian and Asians – Generated by BGE and OpenAI

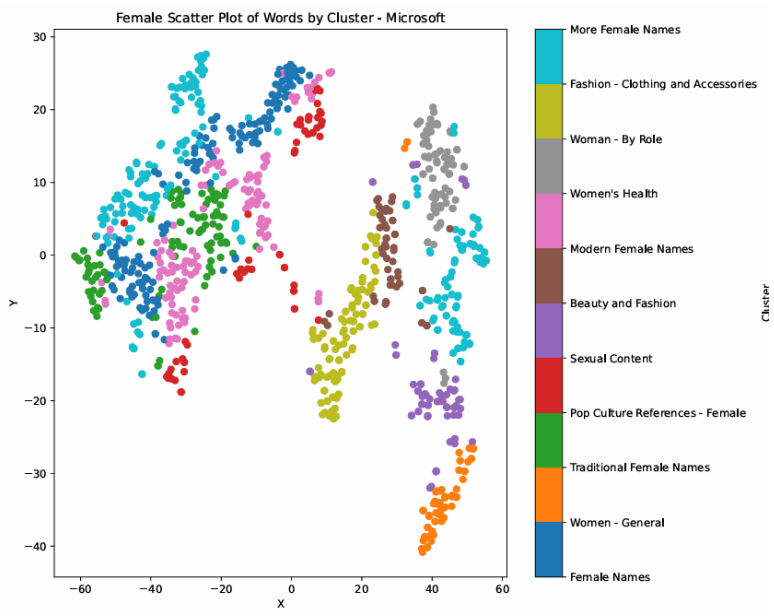
num_words	asian_0	asian_0.2	asian_0.5	asian_0.8	black_0	black_0.2	black_0.5	black_0.8
100	89	86	68	54	11	11	9	5
1000	950	897	730	540	50	32	17	7
10000	9062	8430	6858	4743	938	563	236	100
100000	78270	69567	53272	34445	21730	15070	7779	3404

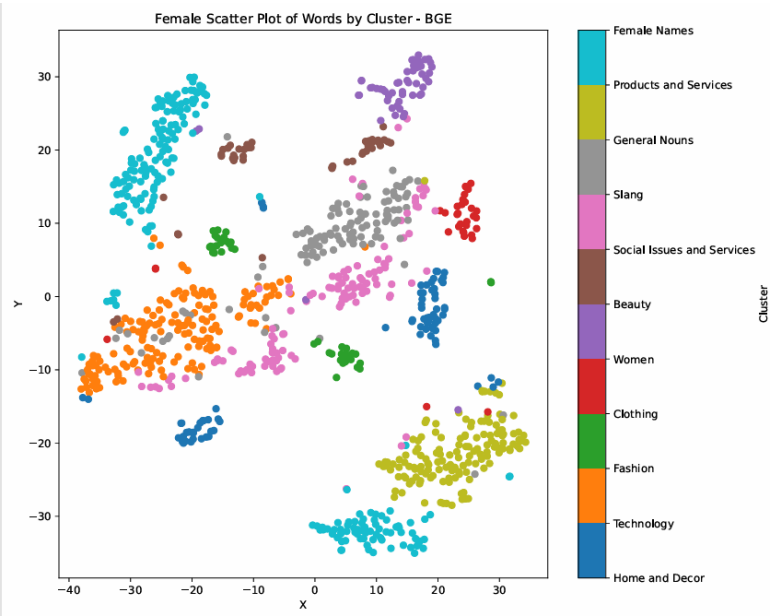
Race-Associated (Asian vs Black) by Range and Effect Size - OpenAI Model

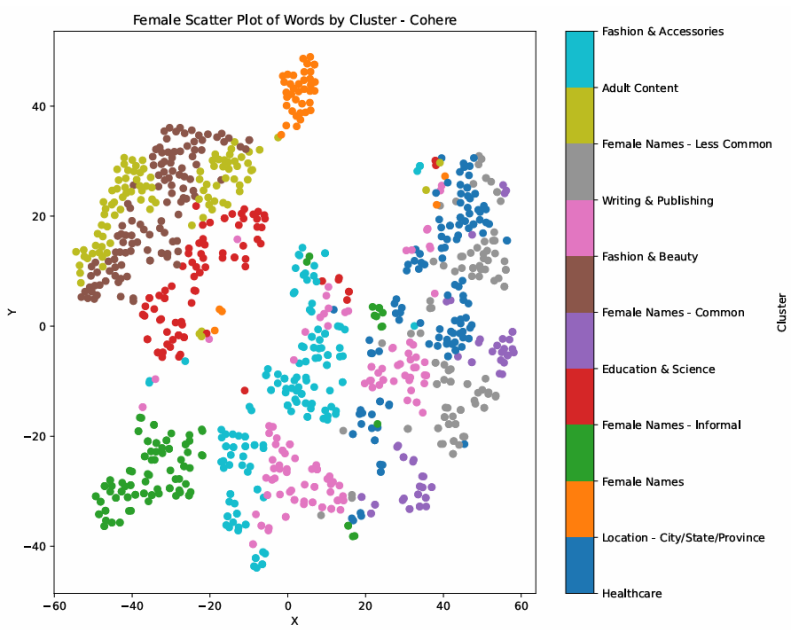
num_words	caucasian_0	caucasian_0.2	caucasian_0.5	caucasian_0.8	black_0	black_0.2	black_0.5	black_0.8
100	85	82	70	49	15	12	9	9
1000	959	928	785	538	41	26	11	9
10000	9158	8587	7007	4688	842	482	177	67
100000	80654	72159	54904	33649	19346	12889	6604	3028

Race-Associated (Caucasian vs Black) by Range and Effect Size - OpenAI Model

Result: Female Semantic Clustering

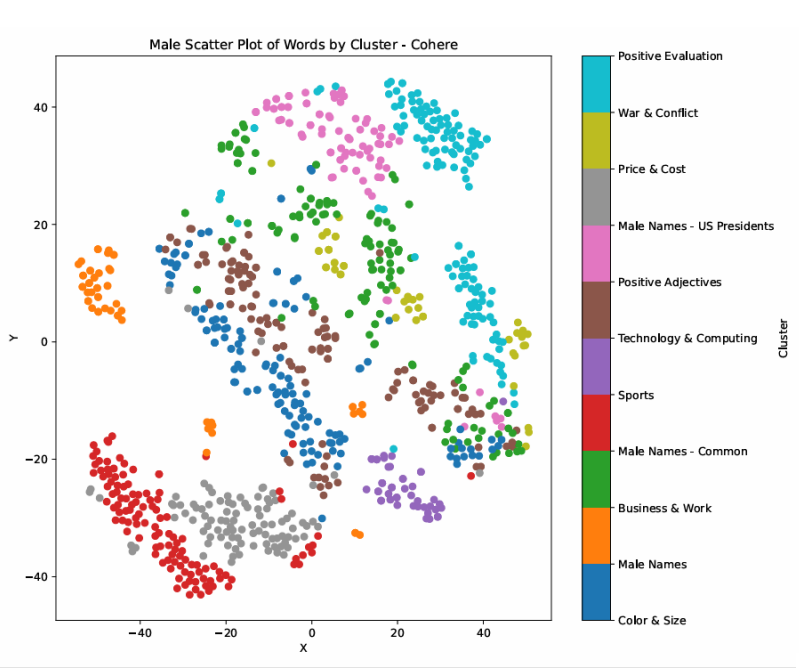


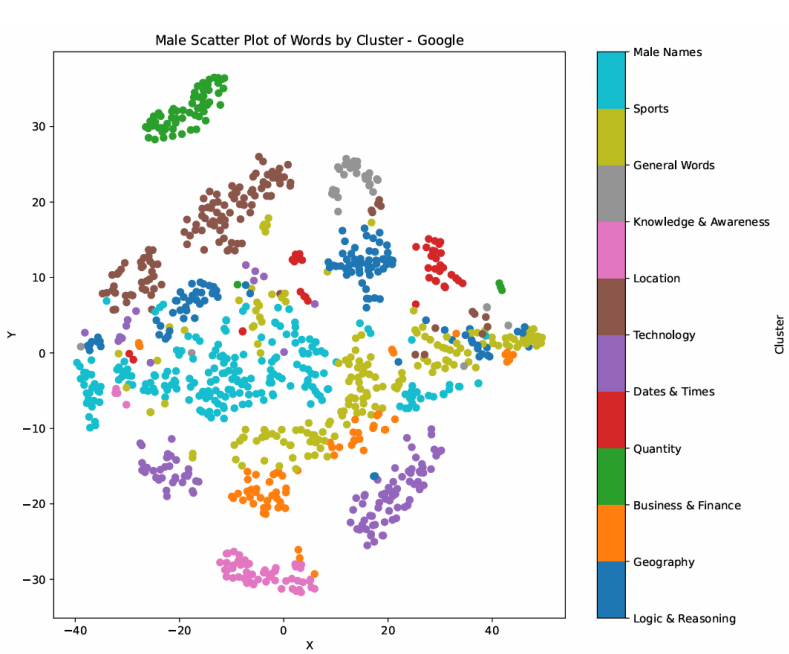


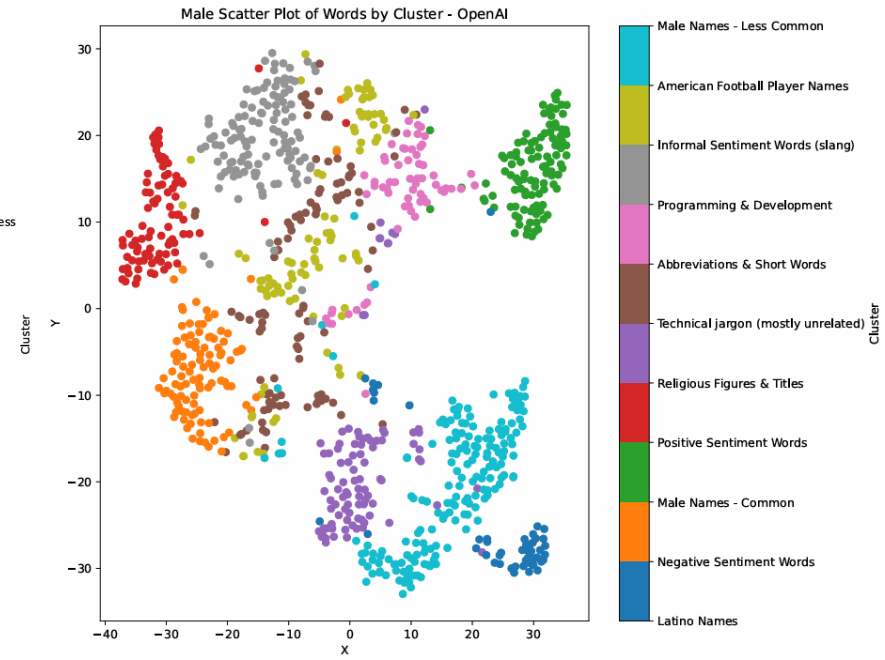


Common female-associated clusters are related to healthcare, home decor, beauty, fashion, and sexual content.

Result: Male Semantic Clustering

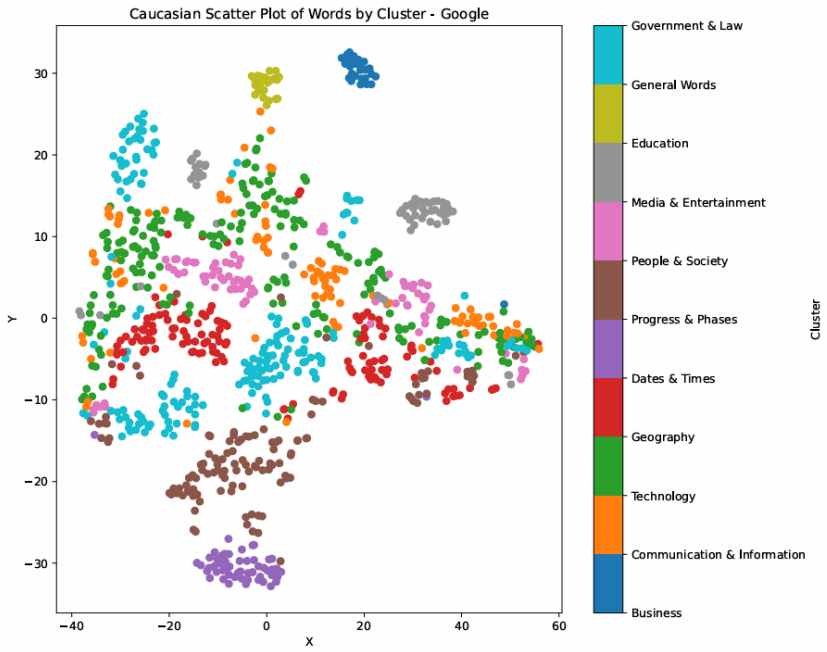


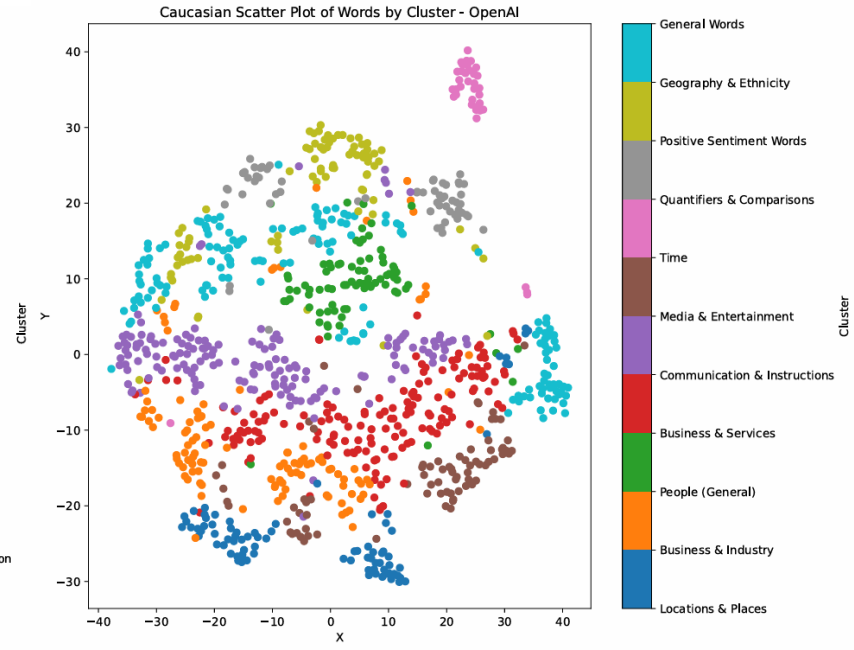


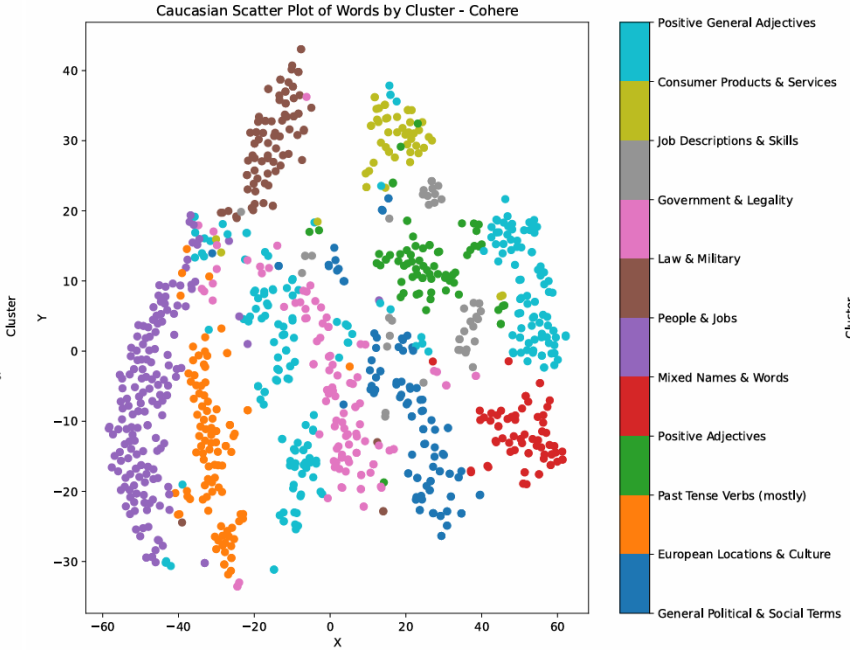


Male-associated clusters commonly focus on technology, sports, business, and sentiment words

Result: Caucasian Semantic Clustering

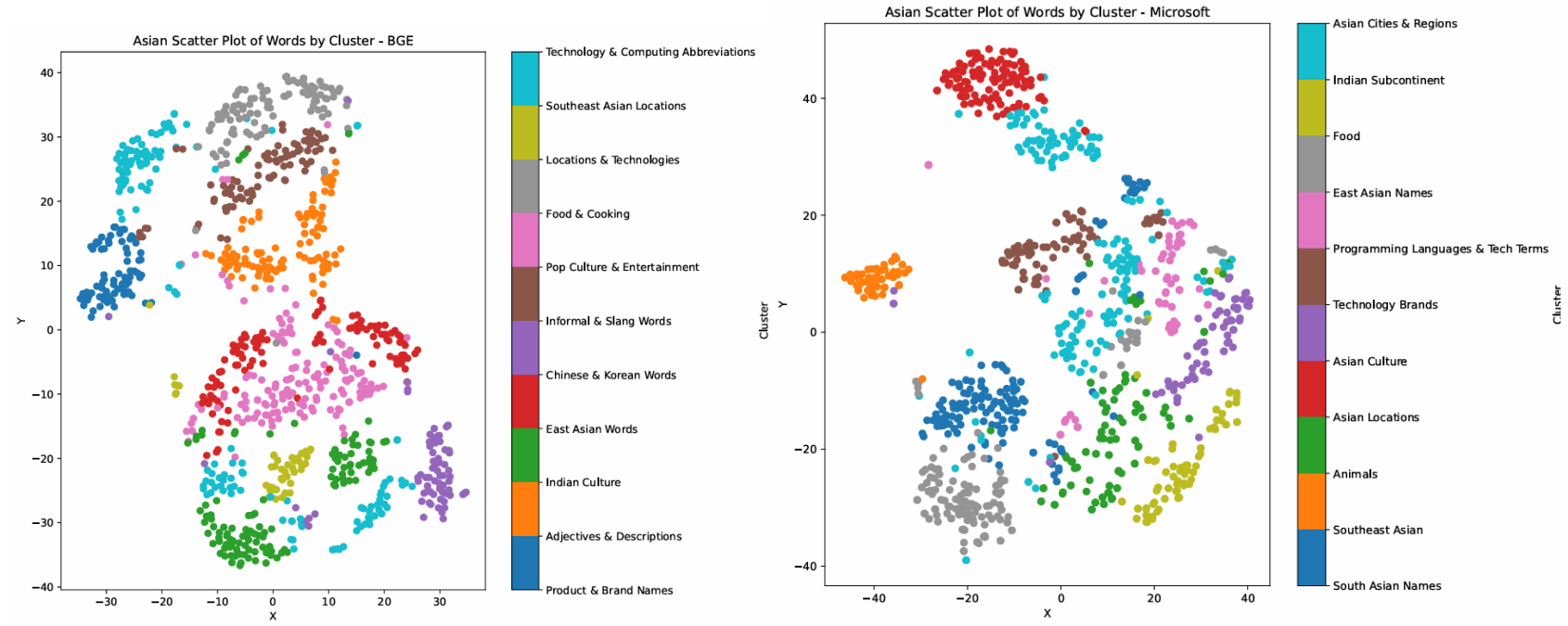


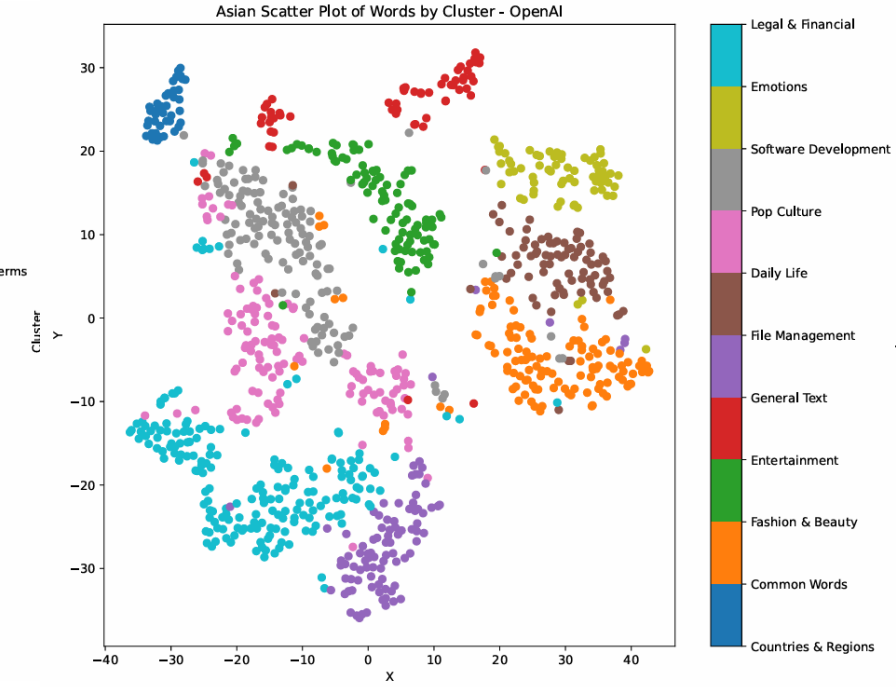




Common Caucasian-associated clusters include business, people & society, education, media, and technology

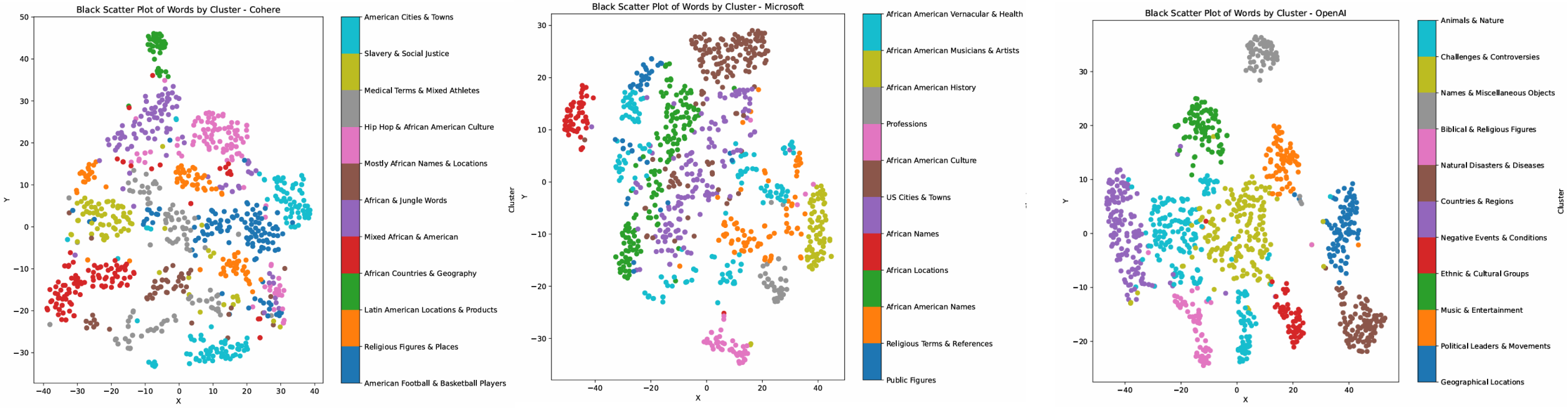
Result: Asian Semantic Clustering





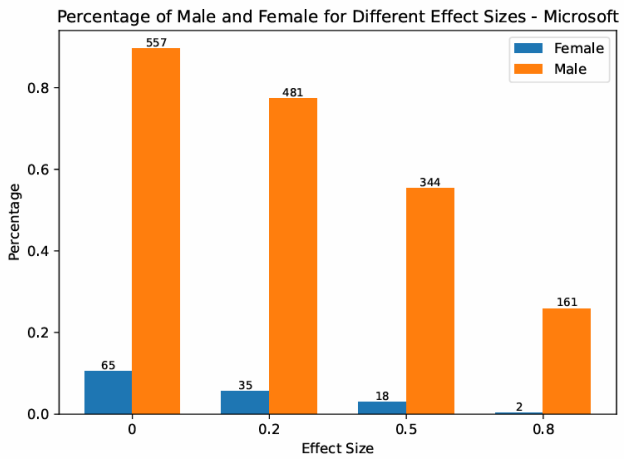
Asian-associated clusters frequently relate to business, software engineering, technology, entertainment, and food and culture

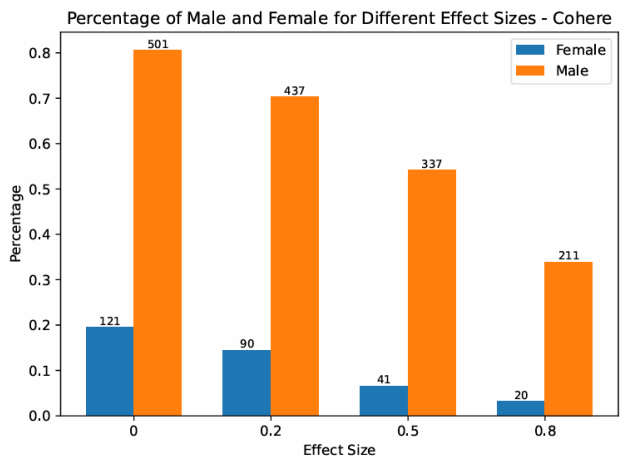
Result: Black Semantic Clustering

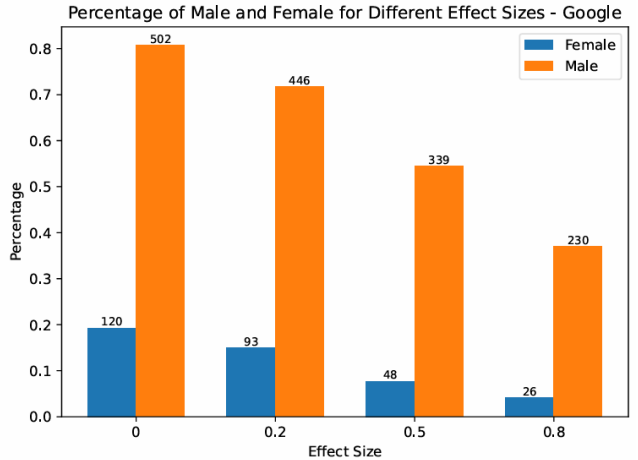


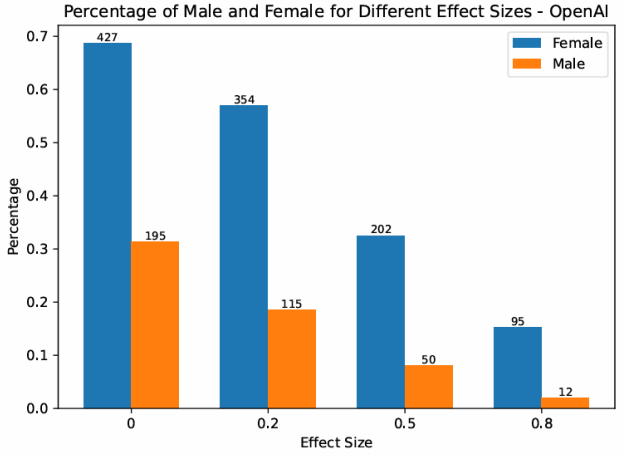
Black-associated clusters typically focus on religion, music, athletes and public figures, wild animals, and ethnicity

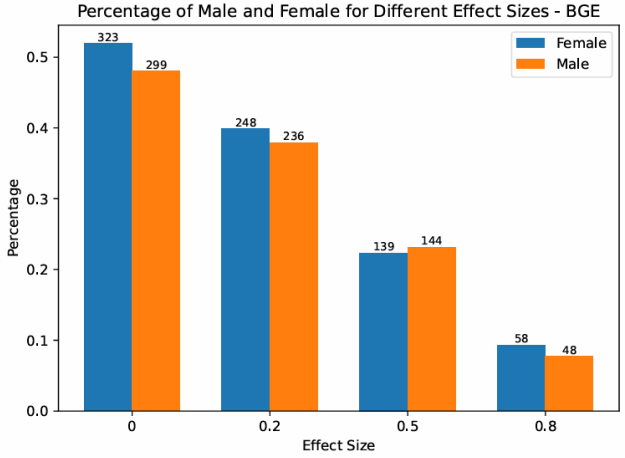
Result: Gender Big Tech Association







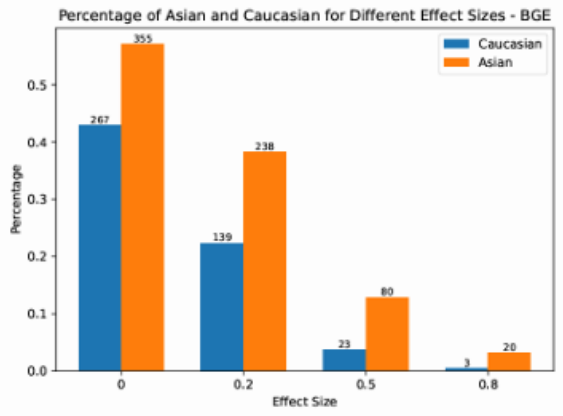


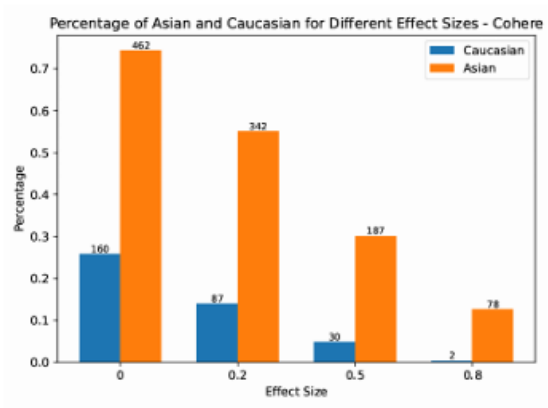


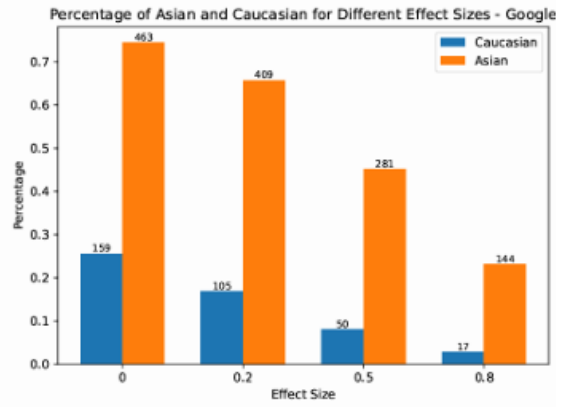
3 out of 5 models indicate a strong relation between male target words and Big Tech words.

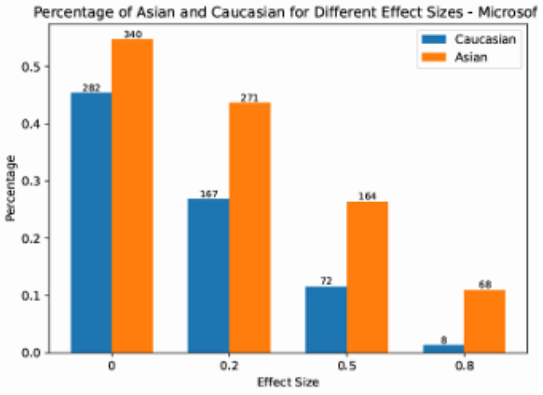
BGE shows minimal bias

Result: Race Big Tech Association

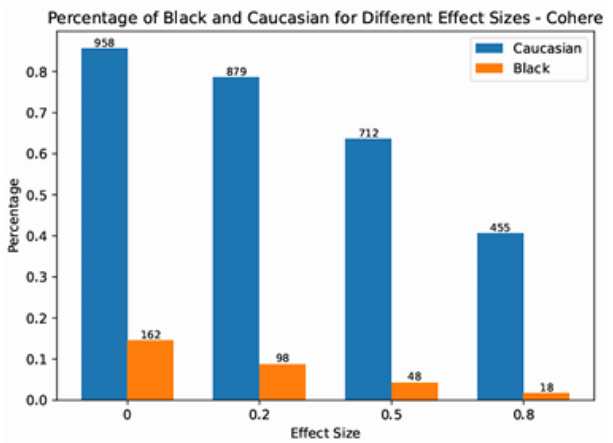




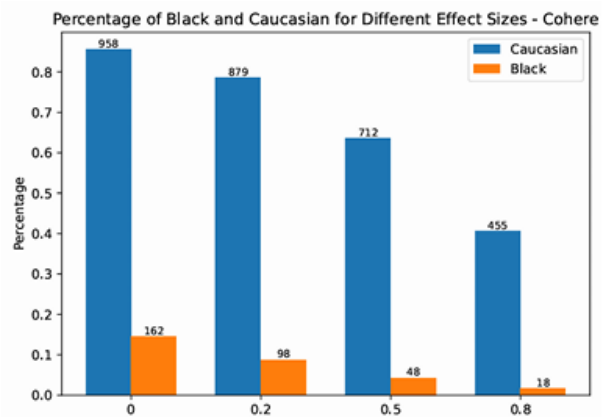




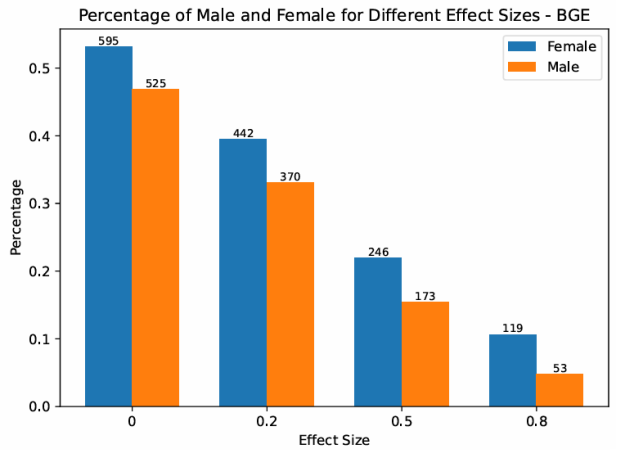
4 out of 5 indicate more Asian words are associated with Big tech words

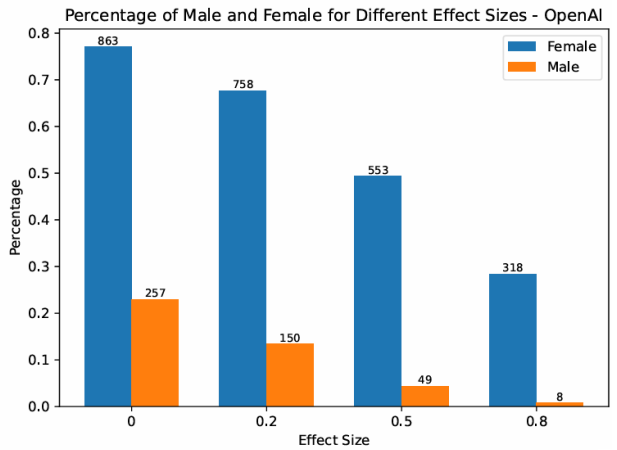


All models demonstrate that Blacks are under represented in the Big Tech domain

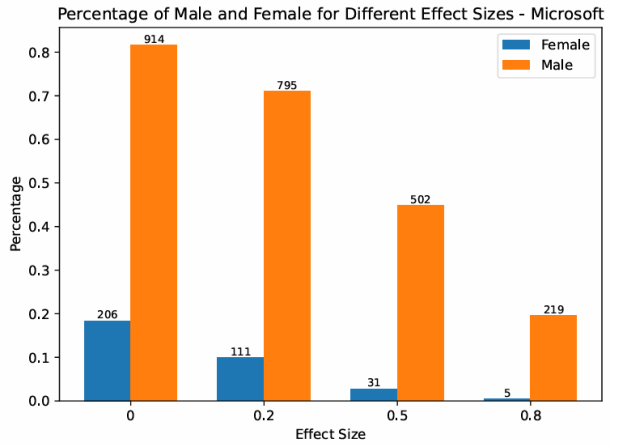


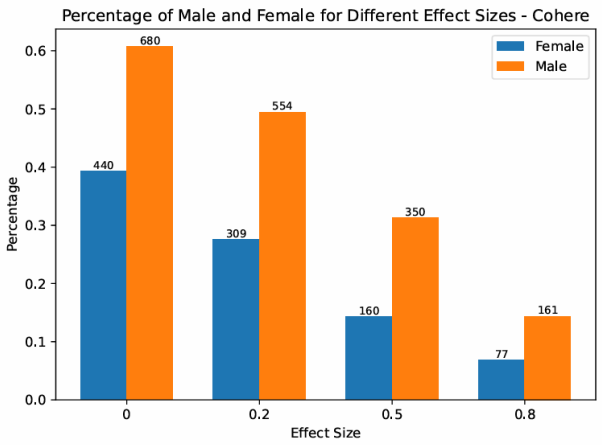
Result: Gender Higher Education Association

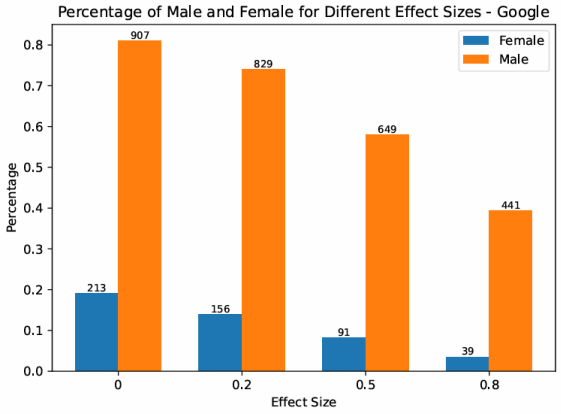




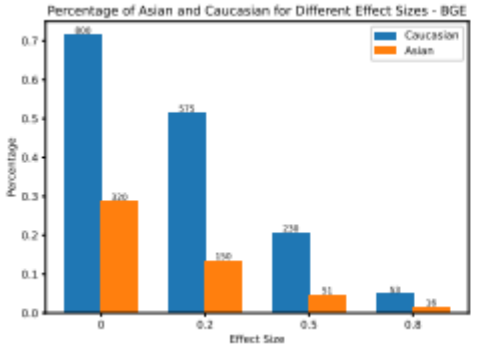
3 out of 5 models indicate a strong relation between male target words and Big Tech words.

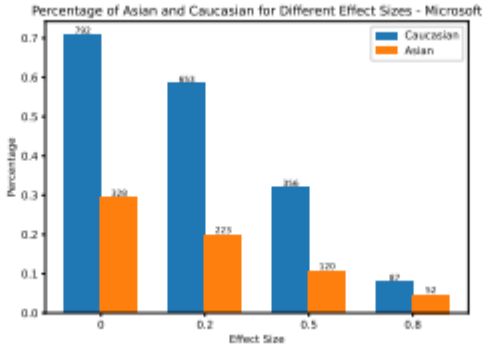


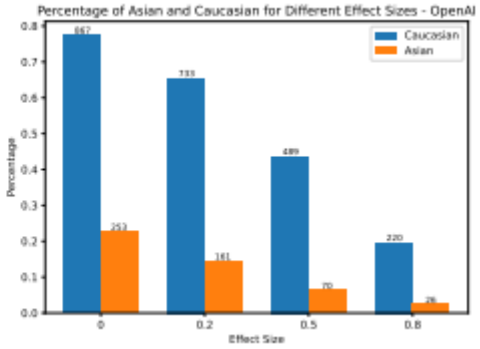




Result: Race Higher Education Association

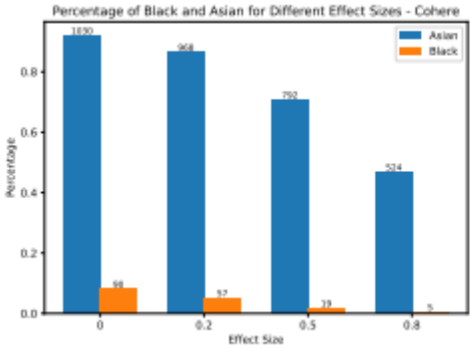


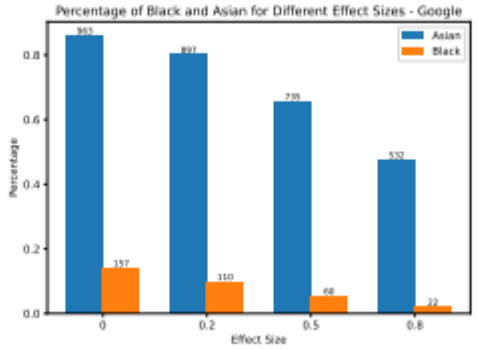


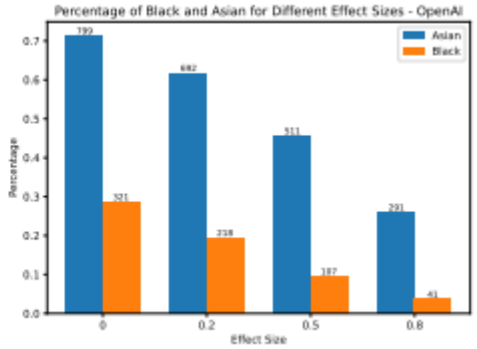


3 out of 5 indicate that more Caucasian words are associated with Higher Education words

4 out of 5 indicate more Asian words are more associated with Big tech words







Discussion

- Have attempted to improve the representation of women in their training data
- Need further development to mitigate persistent biases among different racial groups
- Suggests ongoing challenges in eliminating persistent social stereotypes from embedding models.
- Mirror the pattern of the finding from the previous research
- Pattern that social biases may limit the opportunities for individuals from different demographic groups