06-GIGANT_CHN_w2v

April 7, 2025

[12]:

```
import gensim
import pandas as pd
import spacy
import numpy as np
from wefe.word_embedding_model import WordEmbeddingModel
from wefe.metrics import RIPA
from wefe.query import Query
import re
from gensim.models import KeyedVectors
import logging
import time
import matplotlib.pyplot as plt
import seaborn as sns
```

[10]:

```
def cosine_similarity(v1: np.ndarray, v2: np.ndarray) -> float:
    return np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))

def compute_bias(adj, male_terms, female_terms, model):
    male_mean = np.mean([cosine_similarity(model[adj], model[m]) for m in␣
 ↪male_terms if m in model])
    female_mean = np.mean([cosine_similarity(model[adj], model[f]) for f in␣
 ↪female_terms if f in model])
    return male_mean - female_mean

# NIEUWE Functie: voeg een permutatietest toe voor statistische significantie
def compute_bias_with_pvalue(adj, male_terms, female_terms, model,␣
 ↪permutations=1000, seed=42):
    np.random.seed(seed)

    # echte bias-score berekenen
    real_bias = compute_bias(adj, male_terms, female_terms, model)

    combined_terms = male_terms + female_terms
    num_male = len(male_terms)

    extreme_count = 0
```

```
    for _ in range(permutations):
        np.random.shuffle(combined_terms)
        permuted_male = combined_terms[:num_male]
        permuted_female = combined_terms[num_male:]

        # bereken bias-score onder permutatie
        permuted_bias = compute_bias(adj, permuted_male, permuted_female, model)

        # kijk of permuted bias extremer is dan echte bias
        if abs(permuted_bias) >= abs(real_bias):
            extreme_count += 1

    # Bereken de p-waarde:
    p_value = extreme_count / permutations

    return real_bias, p_value
```

```
[ ]: # Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(message)s'
)

start_time = time.time()

logging.info("Loading Word2Vec model from file...")
model_path = "/Users/matthijstentije/University/MSc_Data-Science/Thesis/
 ↪MSc_Data_Science_Thesis/data/sonar-320.txt"

try:
    model = KeyedVectors.load_word2vec_format(model_path, binary=False)
    logging.info("Model loaded successfully.")
except Exception as e:
    logging.error(f"Failed to load model: {e}")
    raise

elapsed_time = time.time() - start_time
logging.info(f"Model loading took {elapsed_time:.2f} seconds.")

# Check a word
word = "taal"
logging.info(f"Finding most similar words to: '{word}'")

try:
    similar_words = model.most_similar(word)
    logging.info("Similar words found:")
    for w, score in similar_words:
```

```python
        print(f"{w}: {score:.4f}")
except KeyError:
    logging.warning(f"Word '{word}' not found in the vocabulary.")
```

```
2025-04-07 09:45:38,976 [INFO] Loading Word2Vec model from file…
2025-04-07 09:45:38,977 [INFO] loading projection weights from
/Users/matthijstentije/University/MSc_Data-
Science/Thesis/MSc_Data_Science_Thesis/data/sonar-320.txt
2025-04-07 09:47:16,550 [INFO] KeyedVectors lifecycle event {'msg': 'loaded
(626711, 320) matrix of type float32 from
/Users/matthijstentije/University/MSc_Data-
Science/Thesis/MSc_Data_Science_Thesis/data/sonar-320.txt', 'binary': False,
'encoding': 'utf8', 'datetime': '2025-04-07T09:47:16.550052', 'gensim': '4.3.3',
'python': '3.12.1 (v3.12.1:2305ca5144, Dec  7 2023, 17:23:38) [Clang 13.0.0
(clang-1300.0.29.30)]', 'platform': 'macOS-15.3.2-arm64-arm-64bit', 'event':
'load_word2vec_format'}
2025-04-07 09:47:16,560 [INFO] Model loaded successfully.
2025-04-07 09:47:16,564 [INFO] Model loading took 97.59 seconds.
2025-04-07 09:47:16,565 [INFO] Finding most similar words to: 'taal'
2025-04-07 09:47:17,743 [INFO] Similar words found:
```

```
moedertaal: 0.7225
talen: 0.6961
ladino: 0.6668
nederlands: 0.6628
landstaal: 0.6611
engels: 0.6596
kiswahili: 0.6552
papiaments: 0.6535
berbertaal: 0.6488
papiamento: 0.6475
```

[8]:
```python
# Load spaCy Dutch model
nlp = spacy.load('nl_core_news_lg')

def extract_adjectives_from_csv(file_path):
    """
    Leest een CSV-bestand in, parse elke phrase, en retourneert unieke␣
 ↪gelemmatiseerde bijvoeglijke naamwoorden.
    """
    logging.info(f"Loading CSV file: {file_path}")
    try:
        df = pd.read_csv(file_path, delimiter=';', usecols=[0],␣
 ↪names=["Group"], header=0)
        logging.info(f"CSV loaded successfully with shape: {df.shape}")
    except Exception as e:
        logging.error(f"Failed to load CSV file: {e}")
        raise
```

```python
    df.dropna(subset=["Group"], inplace=True)
    logging.info(f"Dropped NaN rows. Remaining phrases: {len(df)}")

    adjectives = []

    logging.info("Starting POS tagging and lemmatization...")
    for idx, phrase in enumerate(df["Group"]):
        doc = nlp(phrase)
        for token in doc:
            if token.pos_ == "ADJ" and token.is_alpha:
                adjectives.append(token.lemma_.lower())
        if idx % 1000 == 0 and idx > 0:
            logging.info(f"Processed {idx} phrases...")

    unique_adjectives = list(dict.fromkeys(adjectives))
    logging.info(f"Extracted {len(unique_adjectives)} unique adjectives.")
    return unique_adjectives

# ---- Use the function ----
csv_file_path = "/Users/matthijstentije/University/MSc_Data-Science/Thesis/
↪MSc_Data_Science_Thesis/data/Corpus_Hedendaags_Nederlands_Adjectives.csv"
result_adjectives = extract_adjectives_from_csv(csv_file_path)

# Log basic stats
print(f"\n Total unique lemmas: {len(result_adjectives)}")

# Words missing from Word2Vec model
missing_words = [word for word in result_adjectives if word not in model]
print(f" Total missing words from embedding model: {len(missing_words)}")
print("Sample missing words:", missing_words[:10])

# Words that exist in the model
filtered_lemmas = [word for word in result_adjectives if word in model]

# Exclude adjectives containing target words
target_words = [
    "man", "kerel", "jongen", "vader", "zoon", "vent", "gast", "meneer", "opa",␣
↪"oom",
    "vrouw", "dame", "meisje", "moeder", "dochter", "tante", "oma", "mevrouw",␣
↪"meid",
]

filtered_adjectives = [
    adj for adj in filtered_lemmas
    if not any(target_word in adj for target_word in target_words)
]
```

```
print(f"Remaining adjectives after filtering: {len(filtered_adjectives)}")
```

```
2025-04-07 09:52:06,433 [INFO] Loading CSV file:
/Users/matthijstentije/University/MSc_Data-Science/Thesis/MSc_Data_Science_Thesi
s/data/Corpus_Hedendaags_Nederlands_Adjectives.csv
2025-04-07 09:52:06,453 [INFO] CSV loaded successfully with shape: (19242, 1)
2025-04-07 09:52:06,463 [INFO] Dropped NaN rows. Remaining phrases: 19239
2025-04-07 09:52:06,464 [INFO] Starting POS tagging and lemmatization…
2025-04-07 09:52:09,651 [INFO] Processed 1000 phrases…
2025-04-07 09:52:12,766 [INFO] Processed 2000 phrases…
2025-04-07 09:52:15,956 [INFO] Processed 3000 phrases…
2025-04-07 09:52:18,848 [INFO] Processed 4000 phrases…
2025-04-07 09:52:21,834 [INFO] Processed 5000 phrases…
2025-04-07 09:52:24,879 [INFO] Processed 6000 phrases…
2025-04-07 09:52:27,998 [INFO] Processed 7000 phrases…
2025-04-07 09:52:31,075 [INFO] Processed 8000 phrases…
2025-04-07 09:52:34,417 [INFO] Processed 9000 phrases…
2025-04-07 09:52:37,728 [INFO] Processed 10000 phrases…
2025-04-07 09:52:40,841 [INFO] Processed 11000 phrases…
2025-04-07 09:52:44,102 [INFO] Processed 12000 phrases…
2025-04-07 09:52:47,070 [INFO] Processed 13000 phrases…
2025-04-07 09:52:49,769 [INFO] Processed 14000 phrases…
2025-04-07 09:52:52,176 [INFO] Processed 15000 phrases…
2025-04-07 09:52:54,765 [INFO] Processed 16000 phrases…
2025-04-07 09:52:57,332 [INFO] Processed 17000 phrases…
2025-04-07 09:52:59,967 [INFO] Processed 18000 phrases…
2025-04-07 09:53:02,691 [INFO] Processed 19000 phrases…
2025-04-07 09:53:03,341 [INFO] Extracted 2938 unique adjectives.


 Total unique lemmas: 2938
 Total missing words from embedding model: 161
Sample missing words: ['polygaum', 'wereldwijaz', 'grooot', 'ongerussen',
'mannek', 'alleenstaan', 'slapap', 'kieen', 'querulante', 'milieubewu']
 Remaining adjectives after filtering: 2753
```

[9]:
```
MALE_WORDS = [
     "man", "kerel", "jongen", "vader", "zoon", "vent", "gast", "meneer",␣
  ↪"opa", "oom",
]
FEMALE_WORDS = [
    "vrouw", "dame", "meisje", "moeder", "dochter", "tante", "oma", "mevrouw",␣
  ↪"meid",
]
```

[11]:
```
logging.info("Starting bias computation for filtered adjectives...")
```

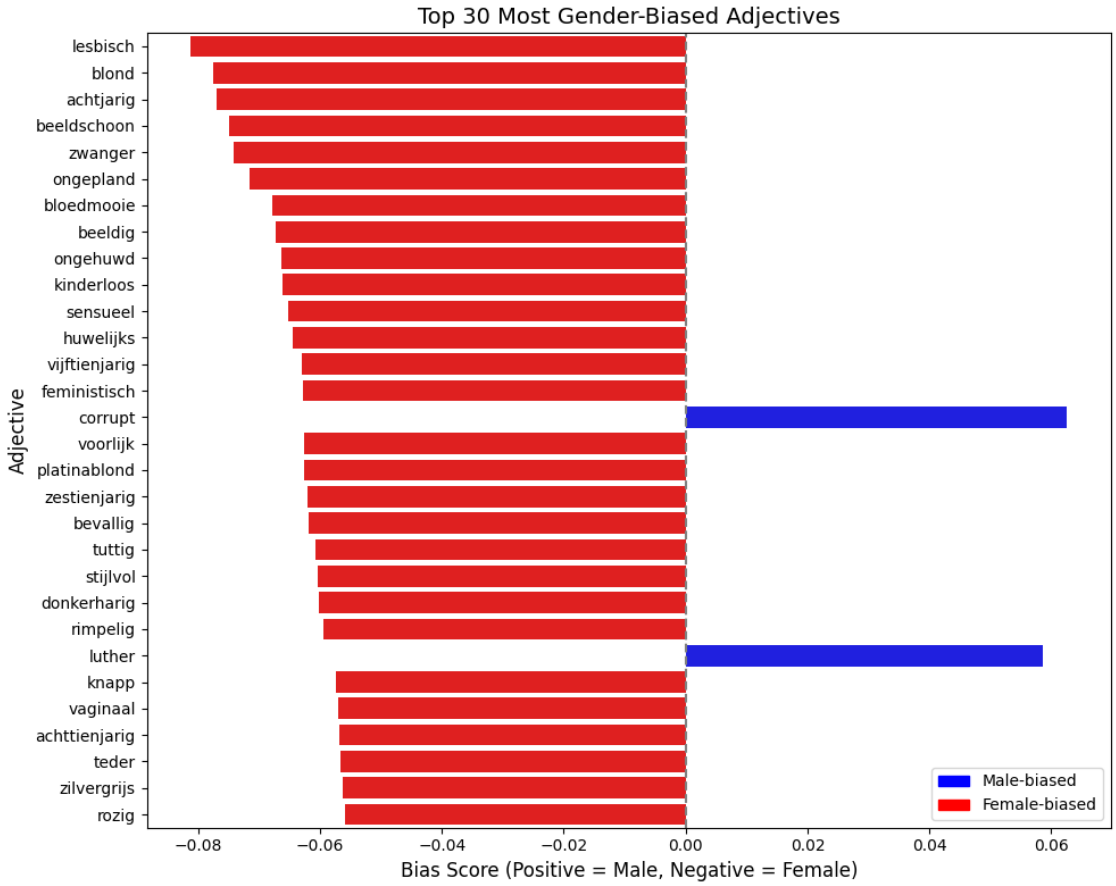
```python
results = []
skipped = []

for i, adj in enumerate(filtered_adjectives):
    if adj in model:
        try:
            bias, pval = compute_bias_with_pvalue(adj, MALE_WORDS,␣
↪FEMALE_WORDS, model)
            results.append({'word': adj, 'bias': bias, 'p_value': pval})
        except Exception as e:
            logging.warning(f"Error processing '{adj}': {e}")
            skipped.append(adj)
    else:
        skipped.append(adj)

logging.info(f"Finished computing bias for {len(results)} adjectives.")
logging.info(f"Skipped {len(skipped)} words (not in model or error during␣
↪computation).")

# Convert results to DataFrame
df_bias_pval = pd.DataFrame(results)

# Sort by p-value
df_bias_pval_sorted = df_bias_pval.sort_values(by='p_value', ascending=True)

# Show top entries
print(df_bias_pval_sorted.head(10))
```

```
2025-04-07 09:53:59,026 [INFO] Starting bias computation for filtered
adjectives…
2025-04-07 09:59:01,339 [INFO] Finished computing bias for 2753 adjectives.
2025-04-07 09:59:01,340 [INFO] Skipped 0 words (not in model or error during
computation).

                 word      bias  p_value
1543            knapp -0.057418    0.001
1291        statutair  0.046310    0.002
1962          indisch -0.042601    0.003
1194  maatschappelijk -0.042173    0.003
928             lamme  0.040604    0.006
1473           luther  0.058629    0.006
79        alleenstaand -0.047796    0.006
731           lesbisch -0.081243    0.006
1439            zedig -0.046326    0.007
170           corrupt  0.062605    0.008
```

[13]:

```python
#  'df_bias_pval' containing:
# columns: ['word', 'bias', 'p_value']
```

```python
# First, get the top 30 adjectives based on absolute bias
df_bias_pval['abs_bias'] = df_bias_pval['bias'].abs()
top30_bias = df_bias_pval.sort_values(by='abs_bias', ascending=False).head(30)

# Add a column to specify gender-bias direction (for coloring)
top30_bias['gender'] = top30_bias['bias'].apply(lambda x: 'male' if x > 0 else␣
 ↪'female')

# Define colors: blue for male, red for female
palette = {'male': 'blue', 'female': 'red'}

# Plotting
plt.figure(figsize=(10, 8))
sns.barplot(
    x='bias',
    y='word',
    data=top30_bias,
    hue='gender',
    palette=palette,
    dodge=False
)

plt.title('Top 30 Most Gender-Biased Adjectives', fontsize=14)
plt.xlabel('Bias Score (Positive = Male, Negative = Female)', fontsize=12)
plt.ylabel('Adjective', fontsize=12)
plt.axvline(0, color='grey', linestyle='--')

# Adding legend manually
from matplotlib.patches import Patch
legend_handles = [Patch(color='blue', label='Male-biased'),
                  Patch(color='red', label='Female-biased')]
plt.legend(handles=legend_handles, loc='lower right')

plt.tight_layout()
plt.show()
```



[14]:
```
def compute_individual_bias(
    adjectives,
    male_terms,
    female_terms,
    model,
    exclude_substrings=True
):
    """
    Berekent voor elk bijvoeglijk naamwoord (ADJ) het 'bias'-verschil
    tussen gemiddelde cosinesim. met mannelijke termen en
    gemiddelde cosinesim. met vrouwelijke termen.

    Parameters
    ----------
    adjectives : list of str
        De lijst van te analyseren bijvoeglijke naamwoorden (al schoongemaakt/
↪gelmmatized).
    male_terms : list of str
```

```
        Woorden die 'mannelijkheid' representeren (bijv.␣
↪['man','jongen','vader',...]).
    female_terms : list of str
        Woorden die 'vrouwelijkheid' representeren (bijv.␣
↪['vrouw','meisje','dame',...]).
    model : dict-like of {str -> np.ndarray} of a KeyedVectors-like object
        Jouw embeddingmodel, waarbij je `word in model` kunt checken en␣
↪`model[word]`
        de vector geeft.
    exclude_substrings : bool, default=True
        Of we woorden moeten overslaan die 'male_terms' of 'female_terms'
        als substring bevatten (bijv. "mannenachtig" bevat "man").

    Returns
    -------
    pd.DataFrame
        DataFrame met kolommen:
        ['word', 'male_mean', 'female_mean', 'bias_value'].
        Waar 'bias_value' = male_mean - female_mean.
        Rijen waar niet genoeg info (embeddings) beschikbaar was, worden␣
↪overgeslagen.
    """

    # 1) Optioneel substring-filter:
    if exclude_substrings:
        all_target_words = set(male_terms + female_terms)
        def has_target_substring(adj):
            return any(tw in adj for tw in all_target_words)
        adjectives = [adj for adj in adjectives if not␣
↪has_target_substring(adj)]

    records = []

    # 2) Loop over elke adjective
    for adj in adjectives:
        # Check of het adj in het model zit
        if adj not in model:
            continue

        adj_vec = model[adj]

        # Verzamel cosines met mannelijke termen
        male_sims = []
        for m in male_terms:
            if m in model:
                male_sims.append(cosine_similarity(adj_vec, model[m]))
```

```
        # Verzamel cosines met vrouwelijke termen
        female_sims = []
        for f in female_terms:
            if f in model:
                female_sims.append(cosine_similarity(adj_vec, model[f]))

        # Check of we beide lijsten niet leeg zijn
        if len(male_sims) == 0 or len(female_sims) == 0:
            # Dan kunnen we geen bias berekenen
            continue

        # Gemiddelde cosines
        male_mean = np.mean(male_sims)
        female_mean = np.mean(female_sims)

        # Bias = verschil man - vrouw
        bias_value = male_mean - female_mean

        records.append({
            "word": adj,
            "male_mean": male_mean,
            "female_mean": female_mean,
            "bias_value": bias_value
        })

    # Omzetten naar DataFrame
    df_bias = pd.DataFrame(records)

    return df_bias

individual_bias = compute_individual_bias(adjectives=filtered_adjectives,␣
 ↪male_terms=MALE_WORDS, female_terms=FEMALE_WORDS, model=model)
print(f"Totaal aantal berekende bias-waarden: {len(individual_bias)}")

    # Sorteer oplopend op 'bias_value'
df_sorted = individual_bias.sort_values("bias_value", ascending=True)

    # Selecteer de 30 laagste scores
lowest_30 = df_sorted.head(30)

    # Selecteer de 30 hoogste scores
highest_30 = df_sorted.tail(30)

print("=== 30 Laagste Scores (bias_value) ===")
for _, row in lowest_30.iterrows():
    print(f"{row['word']}: {row['bias_value']:.4f} "
```

```
            f"(male_mean={row['male_mean']:.4f},␣
↪female_mean={row['female_mean']:.4f})")

print("\n=== 30 Hoogste Scores (bias_value) ===")
for _, row in highest_30.iloc[::-1].iterrows():
    print(f"{row['word']}: {row['bias_value']:.4f} "
        f"(male_mean={row['male_mean']:.4f}, female_mean={row['female_mean']:.
↪4f})")

df_sorted_male = individual_bias.sort_values("male_mean", ascending=False)
top_30_male_mean = df_sorted_male.head(30)

print("\n=== 30 Hoogste male_mean (onafhankelijk van female_mean) ===")
for _, row in top_30_male_mean.iterrows():
    print(f"{row['word']} - male_mean={row['male_mean']:.4f}, "
          f"female_mean={row['female_mean']:.4f}, bias={row['bias_value']:.4f}")

# En idem voor female_mean
df_sorted_female = individual_bias.sort_values("female_mean", ascending=False)
top_30_female_mean = df_sorted_female.head(30)

print("\n=== 30 Hoogste female_mean (onafhankelijk van male_mean) ===")
for _, row in top_30_female_mean.iterrows():
    print(f"{row['word']} - female_mean={row['female_mean']:.4f}, "
          f"male_mean={row['male_mean']:.4f}, bias={row['bias_value']:.4f}")
```

```
Totaal aantal berekende bias-waarden: 2753
=== 30 Laagste Scores (bias_value) ===
lesbisch: -0.0812 (male_mean=0.3926, female_mean=0.4738)
blond: -0.0776 (male_mean=0.3504, female_mean=0.4280)
achtjarig: -0.0769 (male_mean=0.3788, female_mean=0.4558)
beeldschoon: -0.0749 (male_mean=0.3908, female_mean=0.4657)
zwanger: -0.0741 (male_mean=0.4026, female_mean=0.4768)
ongepland: -0.0716 (male_mean=0.2737, female_mean=0.3454)
bloedmooie: -0.0678 (male_mean=0.4200, female_mean=0.4879)
beeldig: -0.0673 (male_mean=0.3034, female_mean=0.3707)
ongehuwd: -0.0663 (male_mean=0.2719, female_mean=0.3382)
kinderloos: -0.0661 (male_mean=0.3426, female_mean=0.4087)
sensueel: -0.0653 (male_mean=0.2770, female_mean=0.3423)
huwelijks: -0.0644 (male_mean=0.1827, female_mean=0.2471)
vijftienjarig: -0.0630 (male_mean=0.2719, female_mean=0.3349)
feministisch: -0.0628 (male_mean=0.2635, female_mean=0.3262)
voorlijk: -0.0625 (male_mean=0.3400, female_mean=0.4025)
platinablond: -0.0625 (male_mean=0.3645, female_mean=0.4270)
zestienjarig: -0.0620 (male_mean=0.3717, female_mean=0.4337)
bevallig: -0.0619 (male_mean=0.3235, female_mean=0.3854)
tuttig: -0.0607 (male_mean=0.2930, female_mean=0.3537)
```

```
stijlvol: -0.0603 (male_mean=0.2799, female_mean=0.3403)
donkerharig: -0.0603 (male_mean=0.3877, female_mean=0.4480)
rimpelig: -0.0594 (male_mean=0.3540, female_mean=0.4134)
knapp: -0.0574 (male_mean=0.1880, female_mean=0.2454)
vaginaal: -0.0570 (male_mean=0.2629, female_mean=0.3199)
achttienjarig: -0.0569 (male_mean=0.3969, female_mean=0.4537)
teder: -0.0566 (male_mean=0.3585, female_mean=0.4151)
zilvergrijs: -0.0563 (male_mean=0.2495, female_mean=0.3058)
rozig: -0.0559 (male_mean=0.2854, female_mean=0.3412)
goudblonde: -0.0556 (male_mean=0.3148, female_mean=0.3704)
spichtig: -0.0554 (male_mean=0.3296, female_mean=0.3850)

=== 30 Hoogste Scores (bias_value) ===
corrupt: 0.0626 (male_mean=0.3008, female_mean=0.2382)
luther: 0.0586 (male_mean=0.3164, female_mean=0.2578)
bekwaam: 0.0536 (male_mean=0.2447, female_mean=0.1911)
plaatsvervangend: 0.0525 (male_mean=0.2351, female_mean=0.1826)
onoverwinnelijk: 0.0517 (male_mean=0.3017, female_mean=0.2500)
incompetent: 0.0512 (male_mean=0.2729, female_mean=0.2217)
misdadig: 0.0507 (male_mean=0.2738, female_mean=0.2231)
impopulair: 0.0507 (male_mean=0.2088, female_mean=0.1581)
geniaal: 0.0496 (male_mean=0.3306, female_mean=0.2810)
vooraanstaand: 0.0494 (male_mean=0.2725, female_mean=0.2231)
sadistisch: 0.0483 (male_mean=0.3525, female_mean=0.3042)
capabel: 0.0483 (male_mean=0.2420, female_mean=0.1937)
actief: 0.0478 (male_mean=0.2211, female_mean=0.1733)
lucratief: 0.0476 (male_mean=0.2225, female_mean=0.1749)
goddeloos: 0.0473 (male_mean=0.3236, female_mean=0.2763)
voortvluchtig: 0.0466 (male_mean=0.2394, female_mean=0.1928)
planmatig: 0.0466 (male_mean=0.1842, female_mean=0.1376)
statutair: 0.0463 (male_mean=0.1488, female_mean=0.1025)
steenrijk: 0.0463 (male_mean=0.3640, female_mean=0.3177)
immoreel: 0.0455 (male_mean=0.2649, female_mean=0.2194)
gewetenloos: 0.0454 (male_mean=0.3343, female_mean=0.2889)
schatrijk: 0.0453 (male_mean=0.3760, female_mean=0.3307)
islamistisch: 0.0449 (male_mean=0.1330, female_mean=0.0880)
overmoedig: 0.0448 (male_mean=0.2759, female_mean=0.2311)
rebel: 0.0447 (male_mean=0.3707, female_mean=0.3259)
crimineel: 0.0446 (male_mean=0.4098, female_mean=0.3652)
maffiose: 0.0439 (male_mean=0.2840, female_mean=0.2401)
knettergekke: 0.0433 (male_mean=0.3033, female_mean=0.2600)
onverbeterlijk: 0.0433 (male_mean=0.3364, female_mean=0.2931)
operationeel: 0.0428 (male_mean=0.1122, female_mean=0.0694)

=== 30 Hoogste male_mean (onafhankelijk van female_mean) ===
grofgebekt - male_mean=0.5118, female_mean=0.4958, bias=0.0161
lief - male_mean=0.5115, female_mean=0.5455, bias=-0.0341
eigenlijk - male_mean=0.4962, female_mean=0.4870, bias=0.0093
```

```
drankzuchtig - male_mean=0.4936, female_mean=0.4769, bias=0.0167
goeiig - male_mean=0.4914, female_mean=0.4845, bias=0.0069
hengstig - male_mean=0.4907, female_mean=0.5027, bias=-0.0120
seksverslaafde - male_mean=0.4879, female_mean=0.4778, bias=0.0101
waarschijnlijk - male_mean=0.4864, female_mean=0.4693, bias=0.0170
precies - male_mean=0.4863, female_mean=0.4581, bias=0.0282
gewoon - male_mean=0.4833, female_mean=0.4814, bias=0.0019
streberig - male_mean=0.4825, female_mean=0.4722, bias=0.0104
echt - male_mean=0.4792, female_mean=0.4667, bias=0.0125
dood - male_mean=0.4782, female_mean=0.4417, bias=0.0365
toevallig - male_mean=0.4738, female_mean=0.4622, bias=0.0117
geile - male_mean=0.4731, female_mean=0.4730, bias=0.0001
stronteigenwijs - male_mean=0.4690, female_mean=0.4618, bias=0.0071
jong - male_mean=0.4650, female_mean=0.4874, bias=-0.0224
ouderloos - male_mean=0.4644, female_mean=0.5037, bias=-0.0392
zeker - male_mean=0.4637, female_mean=0.4502, bias=0.0135
gierigste - male_mean=0.4627, female_mean=0.4581, bias=0.0046
ziek - male_mean=0.4607, female_mean=0.4626, bias=-0.0018
werkelijk - male_mean=0.4606, female_mean=0.4400, bias=0.0206
moe - male_mean=0.4600, female_mean=0.4599, bias=0.0001
sukkelig - male_mean=0.4591, female_mean=0.4356, bias=0.0235
schurkachtig - male_mean=0.4589, female_mean=0.4368, bias=0.0221
gelukkig - male_mean=0.4588, female_mean=0.4722, bias=-0.0134
imbeciel - male_mean=0.4574, female_mean=0.4289, bias=0.0285
vlinderachtig - male_mean=0.4552, female_mean=0.4669, bias=-0.0118
brave - male_mean=0.4542, female_mean=0.4350, bias=0.0191
natuurlijk - male_mean=0.4536, female_mean=0.4561, bias=-0.0025

=== 30 Hoogste female_mean (onafhankelijk van male_mean) ===
lief - female_mean=0.5455, male_mean=0.5115, bias=-0.0341
ouderloos - female_mean=0.5037, male_mean=0.4644, bias=-0.0392
hengstig - female_mean=0.5027, male_mean=0.4907, bias=-0.0120
grofgebekt - female_mean=0.4958, male_mean=0.5118, bias=0.0161
bloedmooie - female_mean=0.4879, male_mean=0.4200, bias=-0.0678
jong - female_mean=0.4874, male_mean=0.4650, bias=-0.0224
eigenlijk - female_mean=0.4870, male_mean=0.4962, bias=0.0093
goeiig - female_mean=0.4845, male_mean=0.4914, bias=0.0069
transseksueel - female_mean=0.4827, male_mean=0.4401, bias=-0.0426
gewoon - female_mean=0.4814, male_mean=0.4833, bias=0.0019
seksverslaafde - female_mean=0.4778, male_mean=0.4879, bias=0.0101
drankzuchtig - female_mean=0.4769, male_mean=0.4936, bias=0.0167
zwanger - female_mean=0.4768, male_mean=0.4026, bias=-0.0741
lesbisch - female_mean=0.4738, male_mean=0.3926, bias=-0.0812
geile - female_mean=0.4730, male_mean=0.4731, bias=0.0001
gelukkig - female_mean=0.4722, male_mean=0.4588, bias=-0.0134
streberig - female_mean=0.4722, male_mean=0.4825, bias=0.0104
schattig - female_mean=0.4694, male_mean=0.4348, bias=-0.0346
waarschijnlijk - female_mean=0.4693, male_mean=0.4864, bias=0.0170
```

```
vlinderachtig - female_mean=0.4669, male_mean=0.4552, bias=-0.0118
echt - female_mean=0.4667, male_mean=0.4792, bias=0.0125
beeldschoon - female_mean=0.4657, male_mean=0.3908, bias=-0.0749
verliefd - female_mean=0.4655, male_mean=0.4429, bias=-0.0225
ziek - female_mean=0.4626, male_mean=0.4607, bias=-0.0018
toevallig - female_mean=0.4622, male_mean=0.4738, bias=0.0117
stronteigenwijs - female_mean=0.4618, male_mean=0.4690, bias=0.0071
buitenechtelijk - female_mean=0.4609, male_mean=0.4268, bias=-0.0340
zwartharig - female_mean=0.4599, male_mean=0.4114, bias=-0.0485
moe - female_mean=0.4599, male_mean=0.4600, bias=0.0001
leeftijdsloos - female_mean=0.4597, male_mean=0.4455, bias=-0.0142
```

[15]:
```
individual_bias_dict = individual_bias.set_index("word")["bias_value"].to_dict()
search_words = ["sterk", "zacht", "moedig", "emotioneel", "dominant",
                "zorgzaam", "aardig", "knap", "schattig"]

print("Bias scores voor specifieke woorden:")
for word in search_words:
    # Let op: als je bij de filtering alles lowercase hebt gemaakt, doe je hier␣
 ↪ook word.lower()
    w_lower = word.lower()
    bias_value = individual_bias_dict.get(w_lower)

    if bias_value is not None:
        print(f"{word}: {bias_value:.3f}")
    else:
        print(f"{word}: Niet gevonden in df_bias (of model).")
```

```
Bias scores voor specifieke woorden:
sterk: 0.014
zacht: -0.037
moedig: -0.022
emotioneel: -0.033
dominant: 0.019
zorgzaam: -0.028
aardig: 0.002
knap: -0.009
schattig: -0.035
```

0.1 RIPA

[ ]:
```
from wefe.word_embedding_model import WordEmbeddingModel

class GensimDutchEmbeddingModel(WordEmbeddingModel):
    def __init__(self, keyed_vectors):
        super().__init__(wv=keyed_vectors)
wrapped_model = GensimDutchEmbeddingModel(model)
```

```python
# Step 5: Test with a word
word = "taal"
logging.info(f"Finding most similar words to: '{word}'")

try:
    similar_words = model.most_similar(word)
    logging.info("Similar words found:")
    for w, score in similar_words:
        print(f"{w}: {score:.4f}")
except KeyError:
    logging.warning(f"Word '{word}' not found in the vocabulary.")
```

```
2025-04-07 10:10:23,661 [INFO] Finding most similar words to: 'taal'
2025-04-07 10:10:23,783 [INFO] Similar words found:
```

```
moedertaal: 0.7225
talen: 0.6961
ladino: 0.6668
nederlands: 0.6628
landstaal: 0.6611
engels: 0.6596
kiswahili: 0.6552
papiaments: 0.6535
berbertaal: 0.6488
papiamento: 0.6475
```

[27]:
```python
print(f"Number of adjectives (filtered_lemmas): {len(filtered_adjectives)}")

# Define the query
query = Query(
    target_sets=[
        ["man", "kerel", "jongen", "vader", "zoon", "vent", "meneer", "opa",␣
 ↪"oom"],
        ["vrouw", "dame", "meisje", "moeder", "dochter", "tante", "oma",␣
 ↪"mevrouw", "meid"]],
    attribute_sets=[filtered_adjectives],  # Ensure it's a list of lists
    target_sets_names=["Male Terms", "Female Terms"],
    attribute_sets_names=["Adjectives"],
)

ripa = RIPA()
result = ripa.run_query(query, wrapped_model)
```

```
Number of adjectives (filtered_lemmas): 2753
```

[28]:
```python
# 'result["word_values"]' {woord: {'mean': x, 'std': y}, ...}
df_ripa = pd.DataFrame({
    'Word': result["word_values"].keys(),
```

```
    'Mean Score': [val['mean'] for val in result["word_values"].values()],
    'Std Dev': [val['std'] for val in result["word_values"].values()],
})

# Sorteer op Mean Score (die RIPA per woord toekent) en bekijk
df_ripa = df_ripa.sort_values(by="Mean Score", ascending=False).
 ↪reset_index(drop=True)
for word in search_words:
    if word in result["word_values"]:
        mean_val = result["word_values"][word]["mean"]
        std_val = result["word_values"][word]["std"]
        print(f"{word}: Mean={mean_val:.3f}, Std={std_val:.3f}")
    else:
        print(f"{word}: niet gevonden in RIPA-result.")
```

```
sterk: Mean=0.019, Std=0.029
zacht: Mean=-0.057, Std=0.078
moedig: Mean=-0.031, Std=0.052
emotioneel: Mean=-0.046, Std=0.064
dominant: Mean=0.007, Std=0.051
zorgzaam: Mean=-0.043, Std=0.072
aardig: Mean=-0.007, Std=0.074
knap: Mean=-0.024, Std=0.095
schattig: Mean=-0.059, Std=0.097
```

[29]:
```
# 1) Voeg een Z-score-kolom toe:
mean_of_scores = df_ripa["Mean Score"].mean()
std_of_scores = df_ripa["Mean Score"].std()

df_ripa["Z-Score"] = (df_ripa["Mean Score"] - mean_of_scores) / std_of_scores

# 2) Als je daarna wilt sorteren op Z-Score (hoog -> laag), doe je:
df_ripa = df_ripa.sort_values("Z-Score", ascending=False).reset_index(drop=True)

# 3) Print de eerste rijen om te zien hoe de Z-scores eruitzien:
print(df_ripa.head(10))
print(df_ripa.tail(10))
```

```
                Word  Mean Score   Std Dev   Z-Score
0             luther    0.093703  0.054713  3.776857
1            corrupt    0.077202  0.069673  3.176023
2               dood    0.070344  0.080848  2.926319
3   onoverwinnelijk    0.069247  0.061142  2.886385
4          goddeloos    0.068410  0.077281  2.855910
5         impopulair    0.065271  0.067593  2.741602
6        gewetenloos    0.063345  0.091916  2.671476
7   plaatsvervangend    0.062924  0.059071  2.656163
8        incompetent    0.062248  0.075782  2.631555
```

```
9          steenrijk    0.061982  0.068094  2.621870
                  Word  Mean Score   Std Dev   Z-Score
2743         stijlvol   -0.097000  0.074878 -3.166799
2744     platinablond   -0.097472  0.123362 -3.183998
2745          beeldig   -0.098894  0.086156 -3.235759
2746        bloedmooie   -0.099148  0.087792 -3.245038
2747         ongepland   -0.103742  0.123087 -3.412308
2748         achtjarig   -0.104324  0.122664 -3.433492
2749        beeldschoon   -0.107545  0.117354 -3.550768
2750             blond   -0.107563  0.105564 -3.551410
2751           zwanger   -0.111246  0.118781 -3.685519
2752          lesbisch   -0.124354  0.124735 -4.162793
```

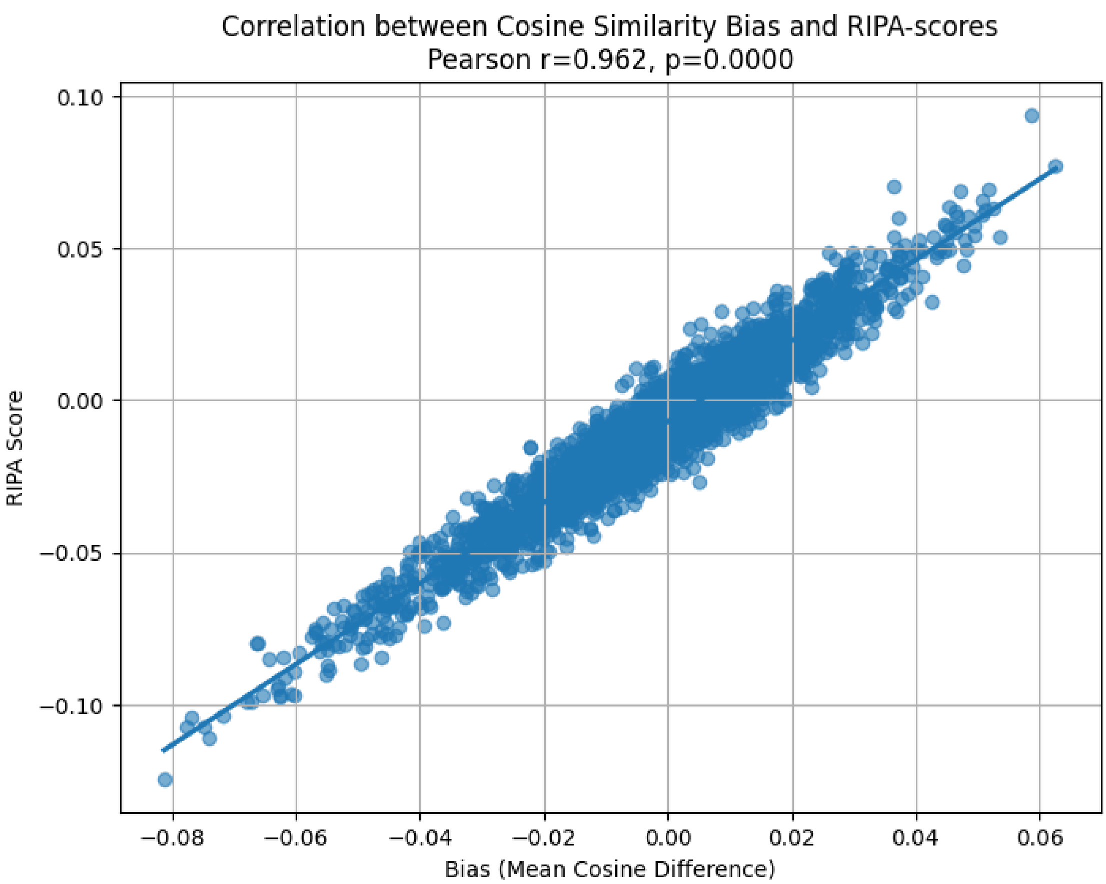
[30]:
```python
df_combined = pd.merge(
    df_bias_pval,
    df_ripa[['Word', 'Mean Score']],
    left_on='word',
    right_on='Word',
    how='inner'
).rename(columns={'Mean Score': 'RIPA_score'})
df_combined['cosine_bias_z'] = (df_combined['bias'] - df_combined['bias'].
 ↪mean()) / df_combined['bias'].std()
df_combined['ripa_z'] = (df_combined['RIPA_score'] - df_combined['RIPA_score'].
 ↪mean()) / df_combined['RIPA_score'].std()
# verwijder overtollige kolom
df_combined.drop('Word', axis=1, inplace=True)

# voorbeeld inspectie
print(df_combined.head())
```

```
         word      bias  p_value  abs_bias  RIPA_score  cosine_bias_z  \
0       groot  0.012608    0.631  0.012608    0.019918       0.741863
1      vreemd  0.015225    0.513  0.015225    0.010203       0.873263
2    prachtig -0.026836    0.243  0.026836   -0.048913      -1.239154
3  onschuldig  0.011254    0.617  0.011254    0.010799       0.673814
4     angstig -0.004337    0.883  0.004337   -0.010091      -0.109181

     ripa_z
0  1.090263
1  0.736532
2 -1.415915
3  0.758234
4 -0.002398
```

[32]:
```python
import scipy.stats as stats


# Bereken Pearson correlatie
```

```
corr, p_corr = stats.pearsonr(df_combined['bias'], df_combined['RIPA_score'])
print(f"Correlation between Cosine Similarity Bias and RIPA: r = {corr:.3f}, p␣
↪= {p_corr:.4f}")

# Scatterplot ter visualisatie:
plt.figure(figsize=(8, 6))
sns.regplot(x='bias', y='RIPA_score', data=df_combined, scatter_kws={'alpha':0.
↪6})
plt.title(f"Correlation between Cosine Similarity Bias and RIPA-scores\nPearson␣
↪r={corr:.3f}, p={p_corr:.4f}")
plt.xlabel("Bias (Mean Cosine Difference)")
plt.ylabel("RIPA Score")
plt.grid(True)
plt.show()
```

```
Correlation between Cosine Similarity Bias and RIPA: r = 0.962, p = 0.0000
```



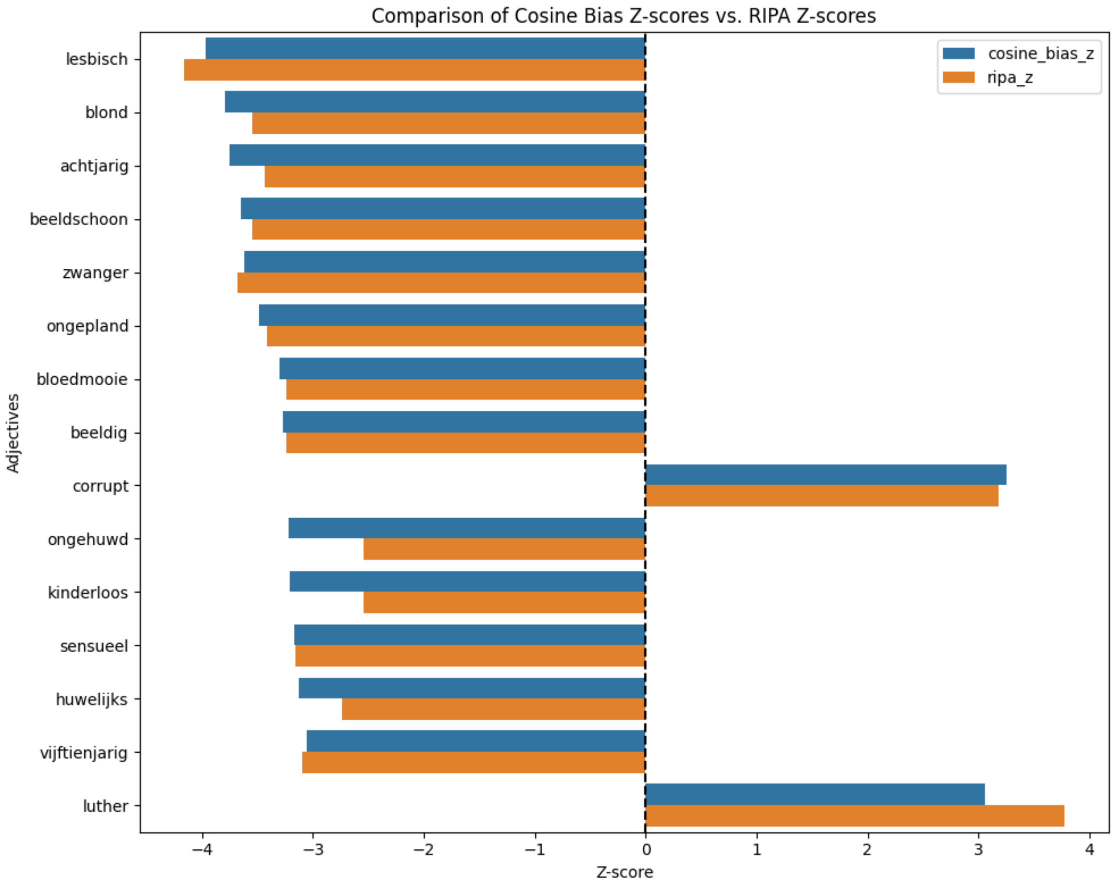
[33]:
```
# Top 10 most similar scores between the two metrics
df_combined['score_diff'] = abs(df_combined['bias'] - df_combined['RIPA_score'])
```

```python
consistent_words = df_combined.sort_values('score_diff').head(10)

print("Top 10 most similar scores between the two metrics:")
print(consistent_words[['word', 'bias', 'RIPA_score', 'score_diff']])

# Top 10 least similar scores between the two metrics
inconsistent_words = df_combined.sort_values('score_diff', ascending=False).
 ↪head(10)

print("\nTop 10 least similar scores between the two metrics")
print(inconsistent_words[['word', 'bias', 'RIPA_score', 'score_diff']])
```

```
Top 10 most similar scores between the two metrics:
             word      bias  RIPA_score    score_diff
95      onbevoegd  0.019433    0.019432  7.431954e-07
1150   economisch  0.004967    0.004955  1.216820e-05
2386     onwettig  0.026596    0.026582  1.367368e-05
488     excessief  0.004790    0.004775  1.441641e-05
2502  grootmoedig  0.016608    0.016589  1.959689e-05
431     roekeloos  0.017667    0.017689  2.154335e-05
78      straatarm  0.001493    0.001515  2.160238e-05
2350        lompe  0.018716    0.018689  2.625771e-05
1105       geldig  0.020263    0.020297  3.317744e-05
1668   inspireren  0.021009    0.021046  3.699772e-05

Top 10 least similar scores between the two metrics
              word      bias  RIPA_score  score_diff
731       lesbisch -0.081243   -0.124354    0.043111
564      halfnaakt -0.046169   -0.084281    0.038111
1171        hitsig -0.049474   -0.086785    0.037311
2145       zwanger -0.074148   -0.111246    0.037098
386         intiem -0.036313   -0.073215    0.036902
2408      stijlvol -0.060330   -0.097000    0.036669
1018        tuttig -0.060731   -0.096252    0.035521
2422    glamoureus -0.055189   -0.090350    0.035161
1473        luther  0.058629    0.093703    0.035075
2625  platinablond -0.062529   -0.097472    0.034943
```

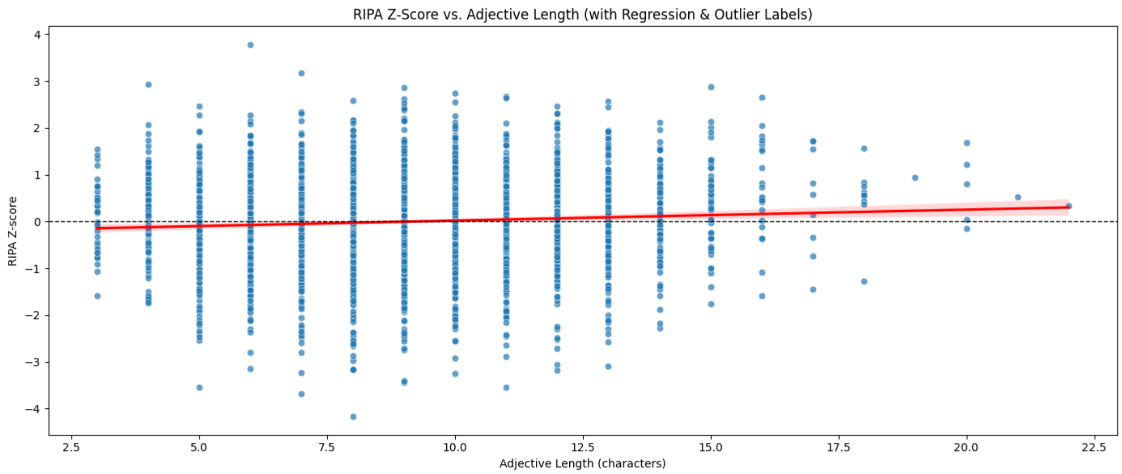
[34]:
```python
# 1) Pick the top 15 based on the absolute cosine z-score (or however you want␣
 ↪to define "top")
top15_z = (
    df_combined
    .reindex(df_combined['cosine_bias_z'].abs().sort_values(ascending=False).
 ↪index)
    .head(15)
)
```

```python
# 2) Melt the dataframe so that `cosine_bias_z` and `ripa_z` become one column,
#    and we can use "Metric" as a hue in the barplot
df_melted = top15_z.melt(
    id_vars='word',
    value_vars=['cosine_bias_z', 'ripa_z'],
    var_name='Metric',
    value_name='Z_score'
)

# 3) Create a grouped bar chart
plt.figure(figsize=(10, 8))
sns.barplot(
    data=df_melted,
    y='word',
    x='Z_score',
    hue='Metric',
    orient='h'  # horizontal bars
)

# 4) Draw a reference line at zero
plt.axvline(0, color='black', linestyle='--')

# 5) Labeling
plt.xlabel('Z-score')
plt.ylabel('Adjectives')
plt.title('Comparison of Cosine Bias Z-scores vs. RIPA Z-scores')

# 6) Layout
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

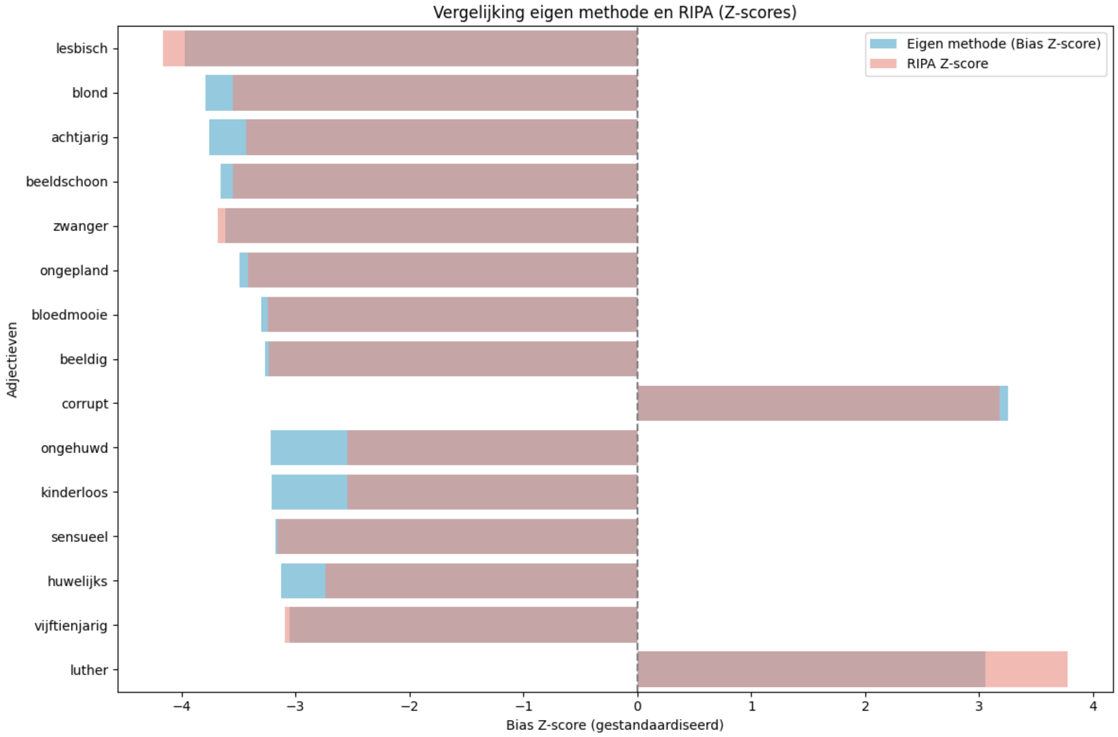


[35]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Make sure 'adjective_length' column exists
df_combined['adjective_length'] = df_combined['word'].str.len()

fig, ax = plt.subplots(figsize=(14, 6))

# --- 1) Scatter Plot ---
sns.scatterplot(
    data=df_combined,
    x='adjective_length',
    y='ripa_z',
    alpha=0.7,
    ax=ax
)

# --- 2) Regression Line ---
```

```
sns.regplot(
    data=df_combined,
    x='adjective_length',
    y='ripa_z',
    scatter=False,    # we already have scatter from above
    line_kws={'color': 'red'},
    ax=ax
)

# --- 3) Reference line at y=0 ---
ax.axhline(0, color='black', linestyle='--', linewidth=1)

# --- 4) Outlier Annotation ---
# Filter rows where |ripa_z| > 4.5 (adjust threshold as desired)
threshold = 4.5
df_outliers = df_combined[df_combined['ripa_z'].abs() > threshold]

# Annotate each outlier
for i, row in df_outliers.iterrows():
    ax.annotate(
        text=row['word'],
        xy=(row['adjective_length'], row['ripa_z']),
        xytext=(5,5),  # offsets text a bit from the point
        textcoords='offset points',
        arrowprops=dict(arrowstyle='->', color='gray', lw=0.5),
        fontsize=9
    )

# --- 5) Axis Labels, Title, Layout ---
ax.set_xlabel('Adjective Length (characters)')
ax.set_ylabel('RIPA Z-score')
ax.set_title('RIPA Z-Score vs. Adjective Length (with Regression & Outlier␣
 ↪Labels)')
plt.tight_layout()
plt.show()
```



[37]:
```
# Kies top 15 adjectieven met hoogste absolute bias_z (van je eigen methode)
top15_z = df_combined.reindex(df_combined['cosine_bias_z'].abs().
    ↪sort_values(ascending=False).index).head(15)

plt.figure(figsize=(12, 8))

# Plot eigen methode
sns.barplot(x='cosine_bias_z', y='word', data=top15_z, color='skyblue',␣
    ↪label='Eigen methode (Bias Z-score)')

# Overlay RIPA Z-scores
sns.barplot(x='ripa_z', y='word', data=top15_z, color='salmon', alpha=0.6,␣
    ↪label='RIPA Z-score')

# Visualisatie details
plt.axvline(0, color='grey', linestyle='--')
plt.xlabel('Bias Z-score (gestandaardiseerd)')
plt.ylabel('Adjectieven')
plt.title('Vergelijking eigen methode en RIPA (Z-scores)')
plt.legend()
plt.tight_layout()
plt.show()
```



Vergelijking eigen methode en RIPA (Z-scores)
Eigen methode (Bias Z-score)
RIPA Z-score
lesbisch
blond
achtjarig
beeldschoon
zwanger
ongepland
bloedmooie
beeldig
corrupt
ongehuwd
kinderloos
sensueel
huwelijks
vijftienjarig
luther
Adjectieven
−4
−3
−2
−1
0
1
2
3
4
Bias Z-score (gestandaardiseerd)