SemAxis is a method for scoring terms along a user-defined axis (e.g., positive-negative, concrete-abstract, hot-cold), which can be used for a range of empirical questions (for one example, see Kozlowski et al. 2019). In this homework, you'll implement SemAxis using word representations from Glove, and use it to explore corpus-specific conceptual associations.

Before running, install gensim with:

```
conda install gensim
```

In [5]:

```
conda install gensim
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying with flexi
ble solve.
Solving environment: ...working... failed with repodata from current_repodata.json, will
retry with next repodata source.
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\12062\Anaconda3\envs\anlp

  added / updated specs:
    - gensim


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    boto3-1.18.21              |     pyhd3eb1b0_0          70 KB
    botocore-1.21.21           |     pyhd3eb1b0_1         3.8 MB
    bz2file-0.98               |    py38haa95532_1         246 KB
    cython-0.29.23             |    py38hd77b12b_0         1.7 MB
    gensim-4.0.1               |    py38hd77b12b_0        18.2 MB
    jmespath-0.10.0            |     pyhd3eb1b0_0          22 KB
    openssl-1.1.1l             |       h2bbff1b_0         4.8 MB
    s3transfer-0.5.0           |     pyhd3eb1b0_0          57 KB
    smart_open-1.9.0           |              py_0          56 KB
    ------------------------------------------------------------
                                           Total:        29.0 MB

The following NEW packages will be INSTALLED:

  boto               pkgs/main/win-64::boto-2.49.0-py38_0
  boto3              pkgs/main/noarch::boto3-1.18.21-pyhd3eb1b0_0
  botocore           pkgs/main/noarch::botocore-1.21.21-pyhd3eb1b0_1
  bz2file            pkgs/main/win-64::bz2file-0.98-py38haa95532_1
  cython             pkgs/main/win-64::cython-0.29.23-py38hd77b12b_0
  gensim             pkgs/main/win-64::gensim-4.0.1-py38hd77b12b_0
  jmespath           pkgs/main/noarch::jmespath-0.10.0-pyhd3eb1b0_0
  s3transfer         pkgs/main/noarch::s3transfer-0.5.0-pyhd3eb1b0_0
  smart_open         pkgs/main/noarch::smart_open-1.9.0-py_0

The following packages will be UPDATED:

  openssl                                 1.1.1k-h2bbff1b_0 --> 1.1.1l-h2bbff1b_0
```

```
Downloading and Extracting Packages

Note: you may need to restart the kernel to use updated packages.
cython-0.29.23       | 1.7 MB    |            |   0%
cython-0.29.23       | 1.7 MB    | 3          |   4%
cython-0.29.23       | 1.7 MB    | ##8        |  29%
cython-0.29.23       | 1.7 MB    | #####      |  50%
cython-0.29.23       | 1.7 MB    | ########3  |  83%

cython-0.29.23       | 1.7 MB    | ########## | 100%

boto3-1.18.21        | 70 KB     |            |   0%
boto3-1.18.21        | 70 KB     | ########## | 100%
boto3-1.18.21        | 70 KB     | ########## | 100%

botocore-1.21.21     | 3.8 MB    |            |   0%
botocore-1.21.21     | 3.8 MB    | 8          |   9%
botocore-1.21.21     | 3.8 MB    | ##4        |  24%
botocore-1.21.21     | 3.8 MB    | ####       |  41%
botocore-1.21.21     | 3.8 MB    | #####5     |  56%
botocore-1.21.21     | 3.8 MB    | #######3   |  74%
botocore-1.21.21     | 3.8 MB    | #########2 |  92%
botocore-1.21.21     | 3.8 MB    | ########## | 100%

smart_open-1.9.0     | 56 KB     |            |   0%
smart_open-1.9.0     | 56 KB     | ##8        |  28%
smart_open-1.9.0     | 56 KB     | ########## | 100%
smart_open-1.9.0     | 56 KB     | ########## | 100%

jmespath-0.10.0      | 22 KB     |            |   0%
jmespath-0.10.0      | 22 KB     | ########## | 100%
jmespath-0.10.0      | 22 KB     | ########## | 100%

s3transfer-0.5.0     | 57 KB     |            |   0%
s3transfer-0.5.0     | 57 KB     | ########## | 100%
s3transfer-0.5.0     | 57 KB     | ########## | 100%

gensim-4.0.1         | 18.2 MB   |            |   0%
gensim-4.0.1         | 18.2 MB   | 1          |   1%
gensim-4.0.1         | 18.2 MB   | 4          |   5%
gensim-4.0.1         | 18.2 MB   | 8          |   9%
gensim-4.0.1         | 18.2 MB   | #2         |  12%
gensim-4.0.1         | 18.2 MB   | #5         |  16%
gensim-4.0.1         | 18.2 MB   | #9         |  19%
gensim-4.0.1         | 18.2 MB   | ##2        |  22%
gensim-4.0.1         | 18.2 MB   | ##5        |  25%
gensim-4.0.1         | 18.2 MB   | ##9        |  29%
gensim-4.0.1         | 18.2 MB   | ###2       |  33%
gensim-4.0.1         | 18.2 MB   | ###5       |  36%
gensim-4.0.1         | 18.2 MB   | ###9       |  40%
gensim-4.0.1         | 18.2 MB   | ####3      |  43%
gensim-4.0.1         | 18.2 MB   | ####6      |  47%
gensim-4.0.1         | 18.2 MB   | #####      |  50%
gensim-4.0.1         | 18.2 MB   | #####3     |  54%
gensim-4.0.1         | 18.2 MB   | #####7     |  57%
gensim-4.0.1         | 18.2 MB   | ######     |  61%
gensim-4.0.1         | 18.2 MB   | ######3    |  64%
gensim-4.0.1         | 18.2 MB   | ######7    |  67%
```

```
gensim-4.0.1         | 18.2 MB   | #######    |  71%
gensim-4.0.1         | 18.2 MB   | #######4   |  74%
gensim-4.0.1         | 18.2 MB   | #######8   |  78%
gensim-4.0.1         | 18.2 MB   | ########1  |  81%
gensim-4.0.1         | 18.2 MB   | ########5  |  85%
gensim-4.0.1         | 18.2 MB   | ########8  |  89%
gensim-4.0.1         | 18.2 MB   | #########2 |  93%
gensim-4.0.1         | 18.2 MB   | #########5 |  96%
gensim-4.0.1         | 18.2 MB   | #########9 |  99%
gensim-4.0.1         | 18.2 MB   | ########## | 100%

bz2file-0.98         | 246 KB    |            |   0%
bz2file-0.98         | 246 KB    | ########## | 100%
bz2file-0.98         | 246 KB    | ########## | 100%

openssl-1.1.1l       | 4.8 MB    |            |   0%
openssl-1.1.1l       | 4.8 MB    | 7          |   8%
openssl-1.1.1l       | 4.8 MB    | #8         |  18%
openssl-1.1.1l       | 4.8 MB    | ##9        |  30%
openssl-1.1.1l       | 4.8 MB    | ####       |  41%
openssl-1.1.1l       | 4.8 MB    | #####1     |  51%
openssl-1.1.1l       | 4.8 MB    | ######4    |  64%
openssl-1.1.1l       | 4.8 MB    | #######5   |  75%
openssl-1.1.1l       | 4.8 MB    | ########6  |  86%
openssl-1.1.1l       | 4.8 MB    | #########7 |  97%
openssl-1.1.1l       | 4.8 MB    | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done
```

In [6]:

```python
import re
from gensim.models import KeyedVectors
from gensim.scripts.glove2word2vec import glove2word2vec
import numpy as np
import numpy.linalg as LA
```

```
C:\Users\12062\Anaconda3\envs\anlp\lib\site-packages\gensim\similarities\__init__.py:15:
UserWarning: The gensim.similarities.levenshtein submodule is disabled, because the opti
onal Levenshtein package <https://pypi.org/project/python-Levenshtein/> is unavailable.
Install Levenhstein (e.g. `pip install python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
```

In this homework, we'll be working with pre-trained word embeddings using the `gensim` library, which provides a number of functions for accessing representations for individual words and comparing them. The representations we'll use come from Glove, which are trained on web data from the Common Crawl corpus.

In [7]:

```python
# First we have to convert the Glove format into w2v format; this creates a new file
glove_file="../data/glove.6B.100d.100K.txt"
glove_in_w2v_format="../data/glove.6B.100d.100K.w2v.txt"
_ = glove2word2vec(glove_file, glove_in_w2v_format)
```

```
<ipython-input-7-86a990e6d4aa>:4: DeprecationWarning: Call to deprecated `glove2word2vec
` (KeyedVectors.load_word2vec_format(.., binary=False, no_header=True) loads GLoVE text
vectors.).
  _ = glove2word2vec(glove_file, glove_in_w2v_format)
```

In [8]:

```
glove = KeyedVectors.load_word2vec_format("../data/glove.6B.100d.100K.w2v.txt", binary=
```

In [9]:

```
good_vector=glove["good"]
```

Functions useful for the first question include the following:

In [16]:

```
# access the representation for a single word
great_vector=glove["great"]

# use numpy to average multiple vector representations together
vecs_to_average=[good_vector, great_vector]
average=np.mean(vecs_to_average, axis=0)
# calculate the cosine similariy between two vectors
cosine_similarity=glove.cosine_similarities(good_vector, [great_vector])

print(good_vector.shape, great_vector.shape, average.shape, cosine_similarity)
```

```
(100,) (100,) (100,) [0.7592798]
```

Q1. Read the SemAxis paper and implement the SemAxis method described in sections 3.1.2 and 3.1.3. Given a set of word embeddings for positive terms $S^+ = \{v_1^+, \ldots v_n^+\}$ and embeddings for negative terms $S^- = \{v_1^-, \ldots v_n^-\}$ that define the endpoints of the axis, your output should be a single real-value score for an input word w with word representation v_w:

$$score(w)_{\mathbf{V}_{\text{axis}}} = \cos(v_w, \mathbf{V}_{\text{axis}})$$

Where:

$$\mathbf{V}^+ = \frac{1}{n}\sum_{1}^{n} v_i^+$$

$$\mathbf{V}^- = \frac{1}{m}\sum_{1}^{m} v_i^-$$

$$\mathbf{V}_{\text{axis}} = \mathbf{V}^+ - \mathbf{V}^-$$

In [52]:

```
def get_semaxis_score(vectors, positive_terms=None, negative_terms=None, target_word=No
    V_plus = []
    V_neg = []

    # your code here
    for p_word in positive_terms:
        V_plus.append(vectors[p_word])

    for n_word in negative_terms:
        V_neg.append(vectors[n_word])

    average_p = np.mean(V_plus, axis=0)
    average_n = np.mean(V_neg, axis=0)

    V_axis = np.subtract(average_p , average_n)

    # calculate the cosine similariy between two vectors
    score = vectors.cosine_similarities(vectors[target_word], [V_axis])
```

```python
    return score[0]
```

In [53]:

```python
# should be 0.342
get_semaxis_score(glove, positive_terms=["woman", "women"], negative_terms=["man", "men
```

Out[53]:

```
0.3424988
```

Now let's score a set of target terms along that axis

In [54]:

```python
def score_list_of_targets(vectors, positive_terms=None, negative_terms=None, target_wor
    scores=[]
    for target in target_words:
        scores.append((get_semaxis_score(vectors, positive_terms, negative_terms, targe

    for k,v in reversed(sorted(scores)):
        print("%.3f\t%s" % (k,v))
```

In [55]:

```python
targets=["doctor", "nurse", "actor", "actress", "mechanic", "librarian", "architect", "
```

In [56]:

```python
score_list_of_targets(glove, positive_terms=["woman", "women"], negative_terms=["man",
```

```
0.342   actress
0.294   nurse
0.219   librarian
0.106   doctor
0.024   actor
0.003   chef
-0.019  cook
-0.075  architect
-0.153  magician
-0.194  mechanic
```

Q2: Define your own concept axis by selecting a set of positive and negative terms and illustrate its utility by scoring a set of 10 target terms (as we did above).

In [58]:

```python
positive_terms=['patience', 'love', 'knowledge', 'perseverance', 'faith', 'hope']
negative_terms=['pride', 'greed', 'wrath', 'envy', 'lust', 'gluttony', 'sloth']
targets=['jealous', 'control', 'playful', 'competitive', 'humble', 'admire', 'frustrati

score_list_of_targets(glove, positive_terms=positive_terms, negative_terms=negative_ter
```

```
0.493   sharing
0.330   control
0.303   mentor
0.280   competitive
0.264   humble
0.199   admire
0.127   frustration
0.048   playful
0.001   anger
-0.087  jealous
```

Q3: Let's assume now that you're able to score all words in a vocabulary along several conceptual dimensions (like the one you've defined) for a given set of word embeddings trained on a dataset. What could you do with that score? Brainstorm possible applications.

Incentivize positive word use in a community sharing environment, or parental control on a children gaming platform, banning words that are scored lower at a certain threashold, and rewardc positive attitude and word choices.