AYDIN ADNAN MENDERES UNIVERSITY

ENGINEERING FACULTY
COMPUTER SCIENCE ENGINEERING DEPARTMENT

Finding Mean Values from 6 Word2Vec Models

CSE431 – Natural Language Processing with Machine Learning 2023/2024

Burak TÜZEL
Talha Alper ASAV

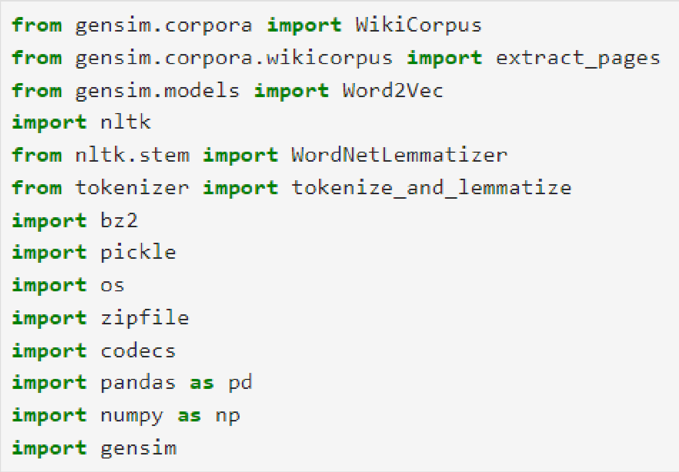
Lecturer:

Asst. Prof. Dr. Fatih SOYGAZİ

Finding Mean Values from 6 Word2Vec Models

Installing necessary environment and importing the libraries:

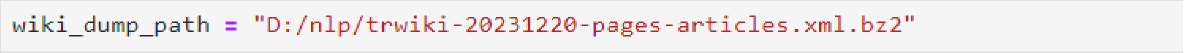
1- Install Jupyter Notebook



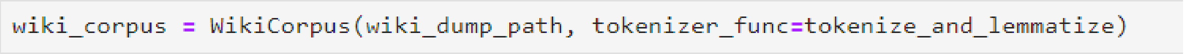
```
from gensim.corpora import WikiCorpus
from gensim.corpora.wikicorpus import extract_pages
from gensim.models import Word2Vec
import nltk
from nltk.stem import WordNetLemmatizer
from tokenizer import tokenize_and_lemmatize
import bz2
import pickle
import os
import zipfile
import codecs
import pandas as pd
import numpy as np
import gensim
```

Downloading the Latest Wiki Dump and Training Our Model:

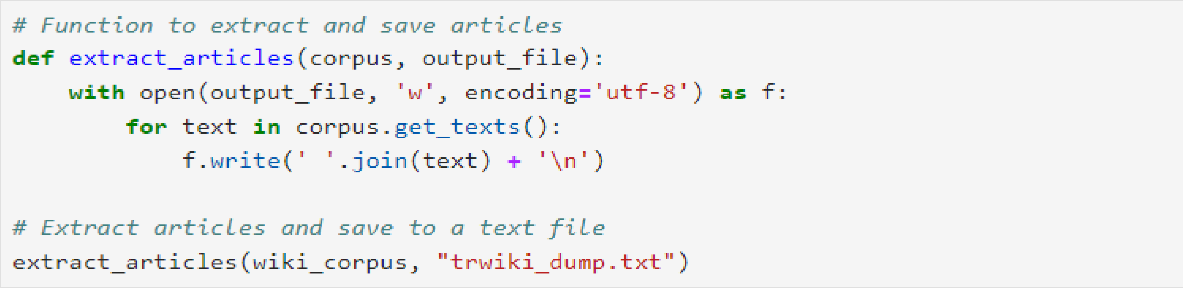
Here is the wiki dump link to download latest dump: https://dumps.wikimedia.org/trwiki/



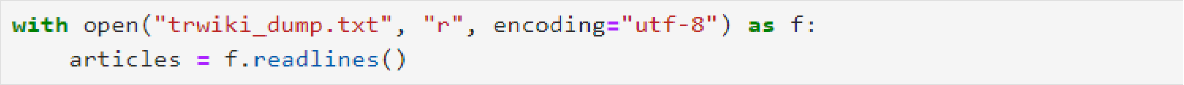
```
wiki_dump_path = "D:/nlp/trwiki-20231220-pages-articles.xml.bz2"
```



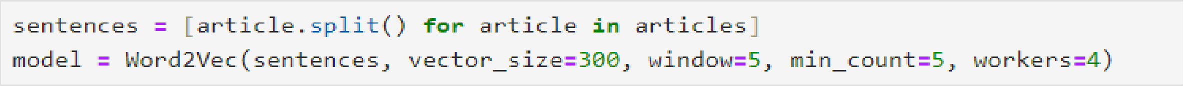
```
wiki_corpus = WikiCorpus(wiki_dump_path, tokenizer_func=tokenize_and_lemmatize)
```



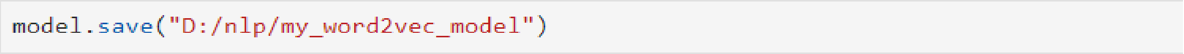
```
# Function to extract and save articles
def extract_articles(corpus, output_file):
    with open(output_file, 'w', encoding='utf-8') as f:
        for text in corpus.get_texts():
            f.write(' '.join(text) + '\n')

# Extract articles and save to a text file
extract_articles(wiki_corpus, "trwiki_dump.txt")
```



```
with open("trwiki_dump.txt", "r", encoding="utf-8") as f:
    articles = f.readlines()
```



```
sentences = [article.split() for article in articles]
model = Word2Vec(sentences, vector_size=300, window=5, min_count=5, workers=4)
```

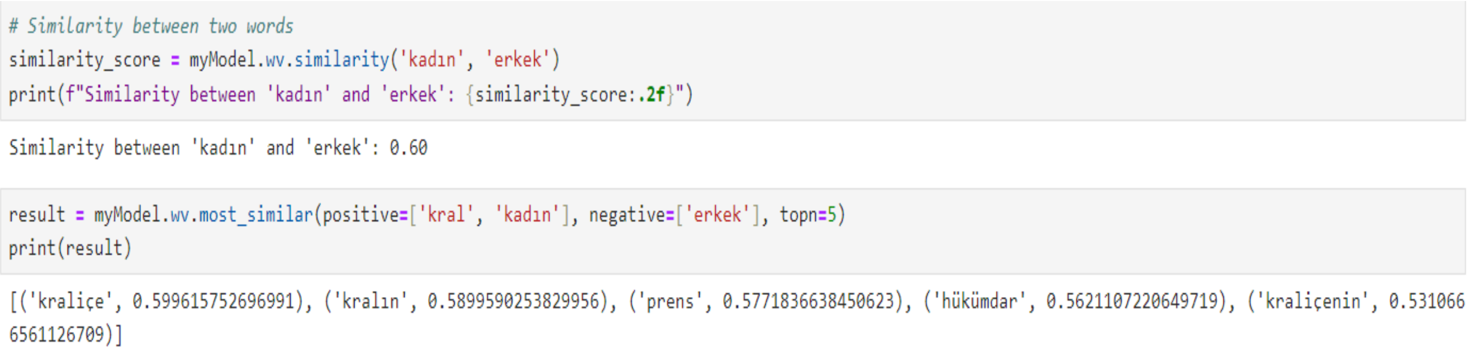


```
model.save("D:/nlp/my_word2vec_model")
```

At the end we saved our model for later. So we can use it later on without training again.

Here Is Some Operations We Did to Test Our Model:

In here we tried different ways to test our model. It can find for example (king+women)-man = queen



```
# Similarity between two words
similarity_score = myModel.wv.similarity('kadın', 'erkek')
print(f"Similarity between 'kadın' and 'erkek': {similarity_score:.2f}")
```

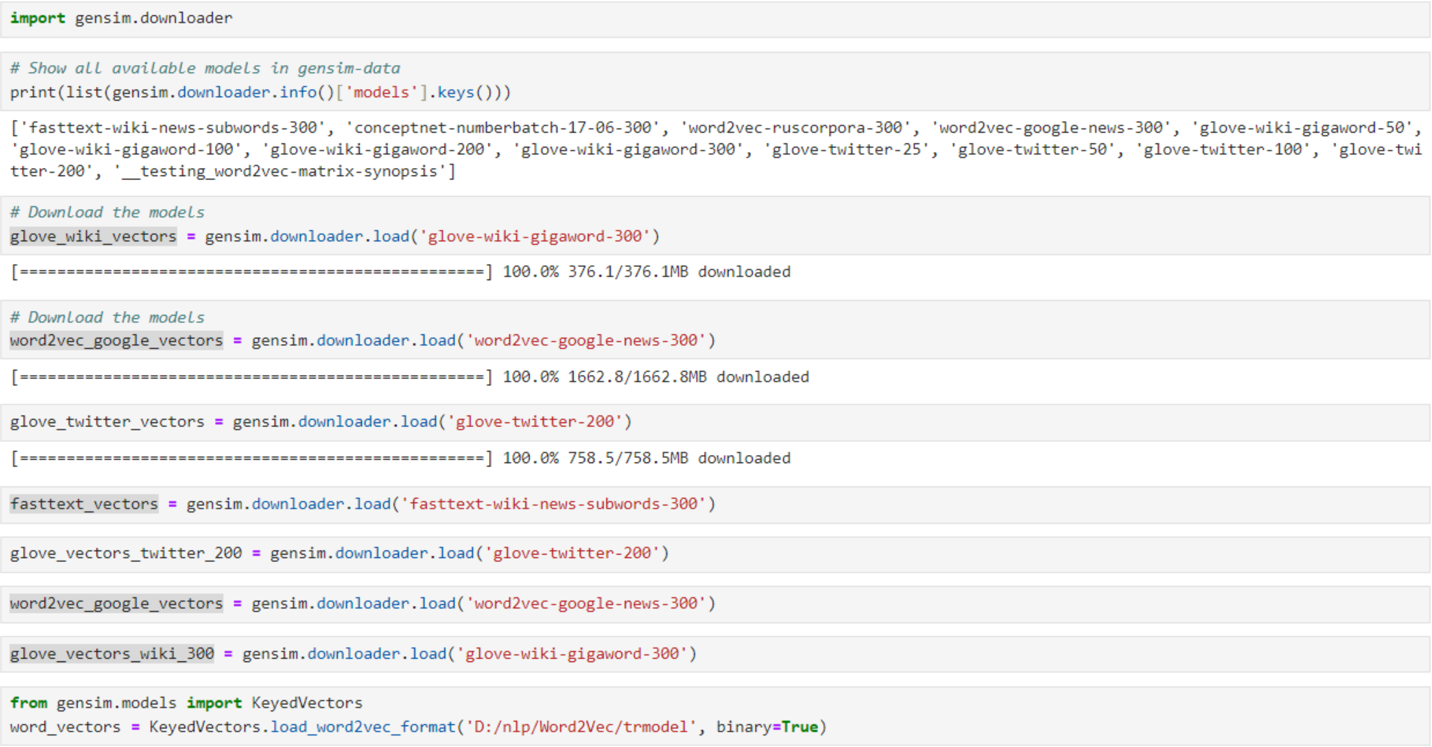
```
Similarity between 'kadın' and 'erkek': 0.60
```

```
result = myModel.wv.most_similar(positive=['kral', 'kadın'], negative=['erkek'], topn=5)
print(result)
```

```
[('kraliçe', 0.599615752696991), ('kralın', 0.5899590253829956), ('prens', 0.5771836638450623), ('hükümdar', 0.5621107220649719), ('kraliçenin', 0.531066
6561126709)]
```

Loading Other 5 Models We Are Going to Use:

Here we loaded the “2018 trmodel” and other 4 that comes with gensim library.



```
import gensim.downloader
```

```
# Show all available models in gensim-data
print(list(gensim.downloader.info()['models'].keys()))
```

```
['fasttext-wiki-news-subwords-300', 'conceptnet-numberbatch-17-06-300', 'word2vec-ruscorpora-300', 'word2vec-google-news-300', 'glove-wiki-gigaword-50',
'glove-wiki-gigaword-100', 'glove-wiki-gigaword-200', 'glove-wiki-gigaword-300', 'glove-twitter-25', 'glove-twitter-50', 'glove-twitter-100', 'glove-twi
tter-200', '__testing_word2vec-matrix-synopsis']
```

```
# Download the models
glove_wiki_vectors = gensim.downloader.load('glove-wiki-gigaword-300')
```

```
[==================================================] 100.0% 376.1/376.1MB downloaded
```

```
# Download the models
word2vec_google_vectors = gensim.downloader.load('word2vec-google-news-300')
```

```
[==================================================] 100.0% 1662.8/1662.8MB downloaded
```

```
glove_twitter_vectors = gensim.downloader.load('glove-twitter-200')
```

```
[==================================================] 100.0% 758.5/758.5MB downloaded
```

```
fasttext_vectors = gensim.downloader.load('fasttext-wiki-news-subwords-300')
```

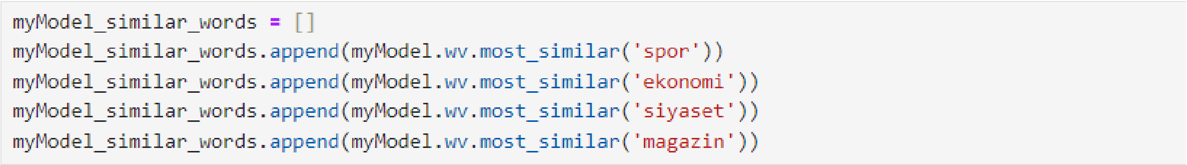
```
glove_vectors_twitter_200 = gensim.downloader.load('glove-twitter-200')
```

```
word2vec_google_vectors = gensim.downloader.load('word2vec-google-news-300')
```

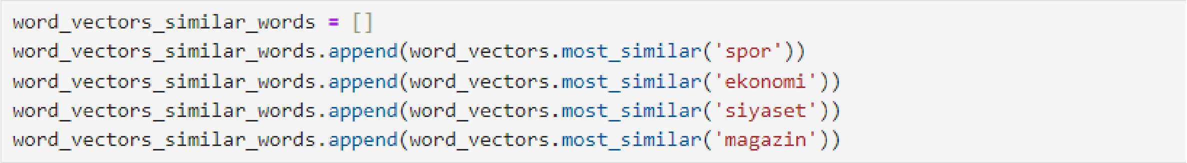
```
glove_vectors_wiki_300 = gensim.downloader.load('glove-wiki-gigaword-300')
```

```
from gensim.models import KeyedVectors
word_vectors = KeyedVectors.load_word2vec_format('D:/nlp/Word2Vec/trmodel', binary=True)
```

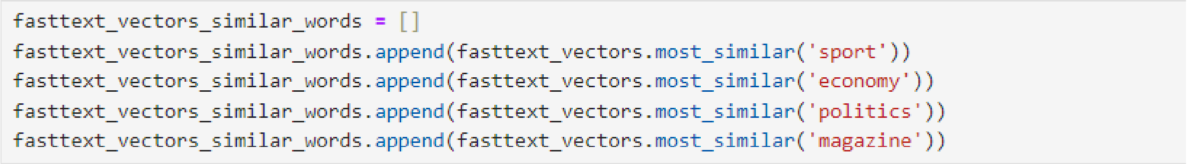
Finding The Similar Words with The Given (‘Spor’,’Siyaset’,’Magazin’,’Ekonomi’) Words:



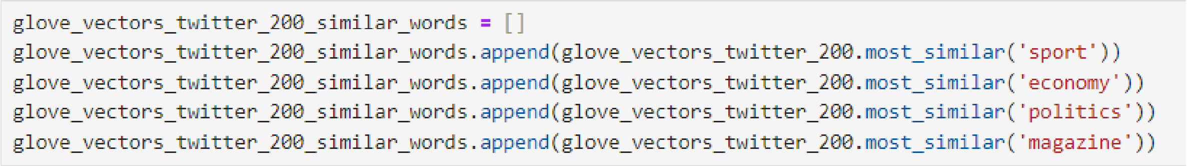
```
myModel_similar_words = []
myModel_similar_words.append(myModel.wv.most_similar('spor'))
myModel_similar_words.append(myModel.wv.most_similar('ekonomi'))
myModel_similar_words.append(myModel.wv.most_similar('siyaset'))
myModel_similar_words.append(myModel.wv.most_similar('magazin'))
```



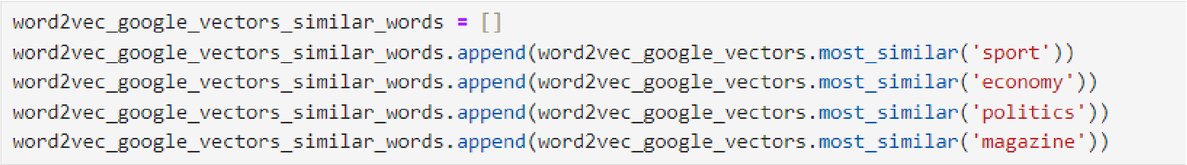
```
word_vectors_similar_words = []
word_vectors_similar_words.append(word_vectors.most_similar('spor'))
word_vectors_similar_words.append(word_vectors.most_similar('ekonomi'))
word_vectors_similar_words.append(word_vectors.most_similar('siyaset'))
word_vectors_similar_words.append(word_vectors.most_similar('magazin'))
```



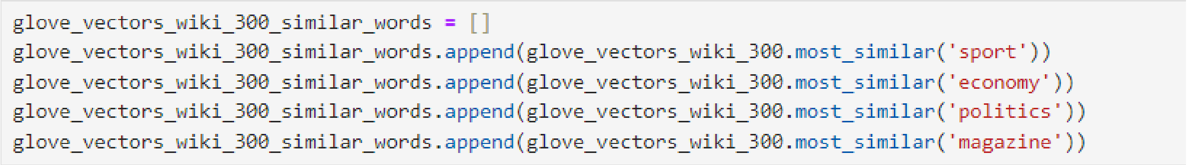
```
fasttext_vectors_similar_words = []
fasttext_vectors_similar_words.append(fasttext_vectors.most_similar('sport'))
fasttext_vectors_similar_words.append(fasttext_vectors.most_similar('economy'))
fasttext_vectors_similar_words.append(fasttext_vectors.most_similar('politics'))
fasttext_vectors_similar_words.append(fasttext_vectors.most_similar('magazine'))
```



```
glove_vectors_twitter_200_similar_words = []
glove_vectors_twitter_200_similar_words.append(glove_vectors_twitter_200.most_similar('sport'))
glove_vectors_twitter_200_similar_words.append(glove_vectors_twitter_200.most_similar('economy'))
glove_vectors_twitter_200_similar_words.append(glove_vectors_twitter_200.most_similar('politics'))
glove_vectors_twitter_200_similar_words.append(glove_vectors_twitter_200.most_similar('magazine'))
```



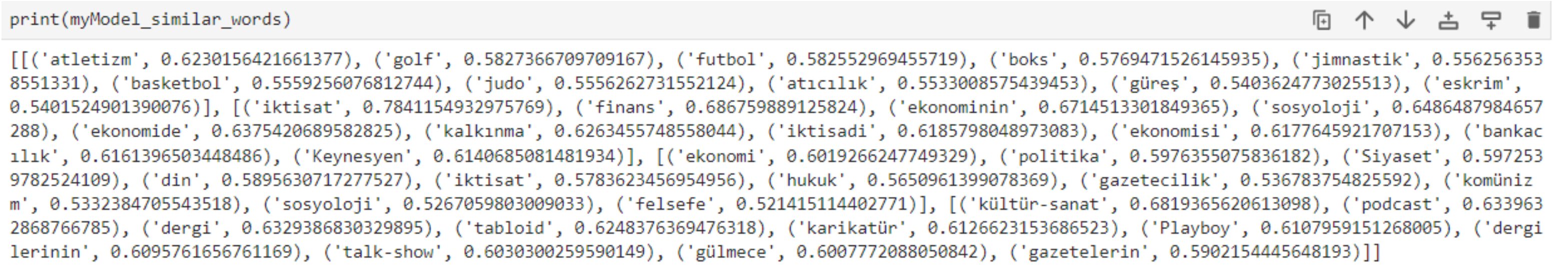
```
word2vec_google_vectors_similar_words = []
word2vec_google_vectors_similar_words.append(word2vec_google_vectors.most_similar('sport'))
word2vec_google_vectors_similar_words.append(word2vec_google_vectors.most_similar('economy'))
word2vec_google_vectors_similar_words.append(word2vec_google_vectors.most_similar('politics'))
word2vec_google_vectors_similar_words.append(word2vec_google_vectors.most_similar('magazine'))
```



```
glove_vectors_wiki_300_similar_words = []
glove_vectors_wiki_300_similar_words.append(glove_vectors_wiki_300.most_similar('sport'))
glove_vectors_wiki_300_similar_words.append(glove_vectors_wiki_300.most_similar('economy'))
glove_vectors_wiki_300_similar_words.append(glove_vectors_wiki_300.most_similar('politics'))
glove_vectors_wiki_300_similar_words.append(glove_vectors_wiki_300.most_similar('magazine'))
```

An Example of The Found Similar Words:

These words were created from the model we trained with the Wiki Dump.

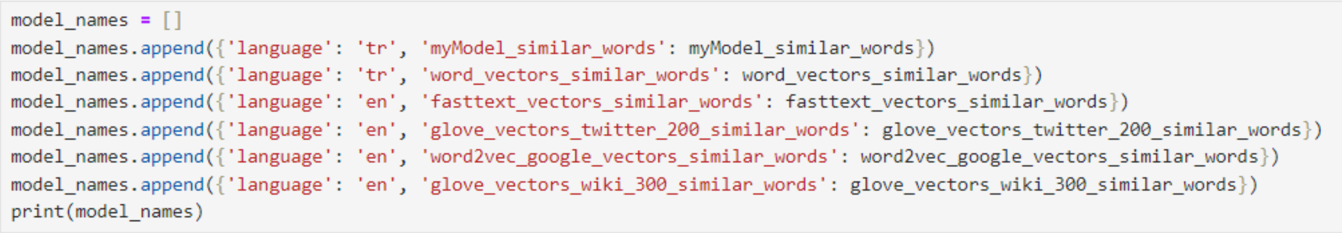


```
print(myModel_similar_words)
```

```
[[('atletizm', 0.6230156421661377), ('golf', 0.5827366709709167), ('futbol', 0.582552969455719), ('boks', 0.5769471526145935), ('jimnastik', 0.556256353
8551331), ('basketbol', 0.5559256076812744), ('judo', 0.5556262731552124), ('atıcılık', 0.5533008575439453), ('güreş', 0.5403624773025513), ('eskrim',
0.5401524901390076)], [('iktisat', 0.7841154932975769), ('finans', 0.686759889125824), ('ekonominin', 0.6714513301849365), ('sosyoloji', 0.6486487984657
288), ('ekonomide', 0.6375420689582825), ('kalkınma', 0.6263455748558044), ('iktisadi', 0.6185798048973083), ('ekonomisi', 0.6177645921707153), ('bankac
ılık', 0.6161396503448486), ('Keynesyen', 0.6140685081481934)], [('ekonomi', 0.6019266247749329), ('politika', 0.5976355075836182), ('Siyaset', 0.597253
9782524109), ('din', 0.5895630717277527), ('iktisat', 0.5783623456954956), ('hukuk', 0.5650961399078369), ('gazetecilik', 0.536783754825592), ('komüniz
m', 0.5332384705543518), ('sosyoloji', 0.5267059803009033), ('felsefe', 0.521415114402771)], [('kültür-sanat', 0.6819365620613098), ('podcast', 0.633963
2868766785), ('dergi', 0.6329386830329895), ('tabloid', 0.6248376369476318), ('karikatür', 0.6126623153686523), ('Playboy', 0.6107959151268005), ('dergi
lerinin', 0.6095761656761169), ('talk-show', 0.6030300259590149), ('gülmece', 0.6007772088050842), ('gazetelerin', 0.5902154445648193)]]
```

Appending the Found Similar Words:

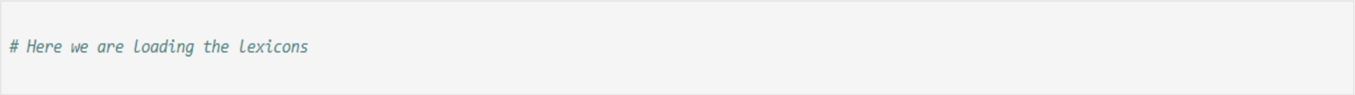
Here we appended the words we found from each model. So we can use them while calculating the mean values with lexicons.



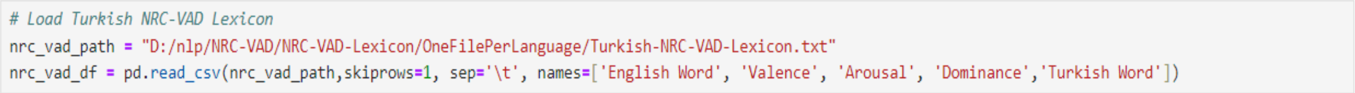
```
model_names = []
model_names.append({'language': 'tr', 'myModel_similar_words': myModel_similar_words})
model_names.append({'language': 'tr', 'word_vectors_similar_words': word_vectors_similar_words})
model_names.append({'language': 'en', 'fasttext_vectors_similar_words': fasttext_vectors_similar_words})
model_names.append({'language': 'en', 'glove_vectors_twitter_200_similar_words': glove_vectors_twitter_200_similar_words})
model_names.append({'language': 'en', 'word2vec_google_vectors_similar_words': word2vec_google_vectors_similar_words})
model_names.append({'language': 'en', 'glove_vectors_wiki_300_similar_words': glove_vectors_wiki_300_similar_words})
print(model_names)
```

Loading The Lexicons We Are Going to Use:

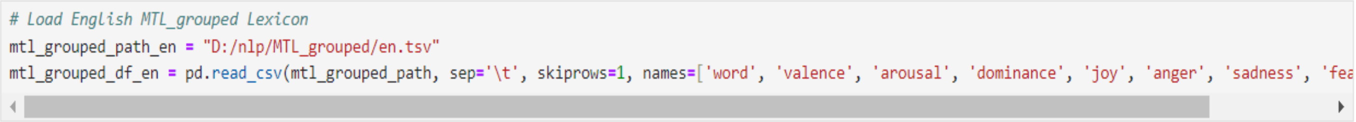
These are the lexicons. NRC-VAD only has “Valence” “Arousal” and “Dominance” values while the MTL-Grouped has additional emotion values such as “Joy” “Anger” “Sadness” etc.



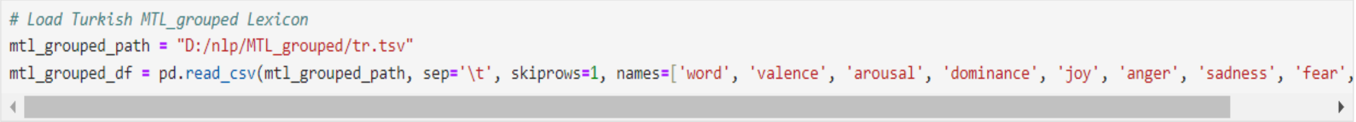
```
# Here we are loading the lexicons
```



```
# Load Turkish NRC-VAD Lexicon
nrc_vad_path = "D:/nlp/NRC-VAD/NRC-VAD-Lexicon/OneFilePerLanguage/Turkish-NRC-VAD-Lexicon.txt"
nrc_vad_df = pd.read_csv(nrc_vad_path,skiprows=1, sep='\t', names=['English Word', 'Valence', 'Arousal', 'Dominance','Turkish Word'])
```



```
# Load English MTL_grouped Lexicon
mtl_grouped_path_en = "D:/nlp/MTL_grouped/en.tsv"
mtl_grouped_df_en = pd.read_csv(mtl_grouped_path, sep='\t', skiprows=1, names=['word', 'valence', 'arousal', 'dominance', 'joy', 'anger', 'sadness', 'fea
```

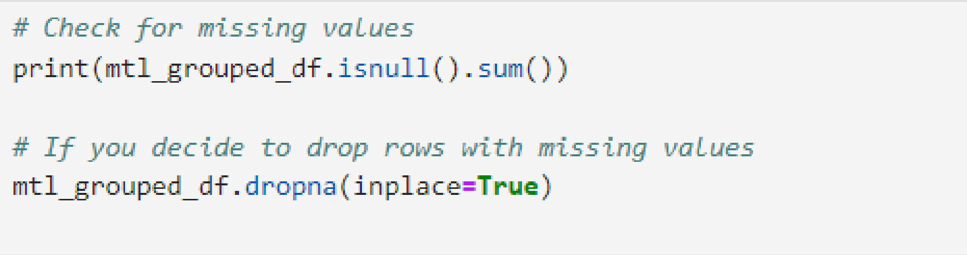


```
# Load Turkish MTL_grouped Lexicon
mtl_grouped_path = "D:/nlp/MTL_grouped/tr.tsv"
mtl_grouped_df = pd.read_csv(mtl_grouped_path, sep='\t', skiprows=1, names=['word', 'valence', 'arousal', 'dominance', 'joy', 'anger', 'sadness', 'fear',
```

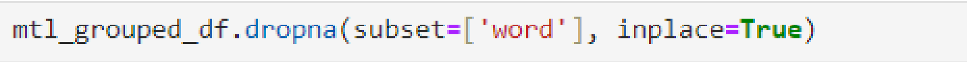
Removing NaN and Checking Datatypes of Lexicons:

Here we removed the nan inside the MTL-Grouped lexicon.

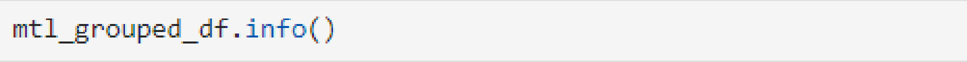
```
# Here we checked the null variables inside the lexicons
```



```
# Check for missing values
print(mtl_grouped_df.isnull().sum())

# If you decide to drop rows with missing values
mtl_grouped_df.dropna(inplace=True)
```

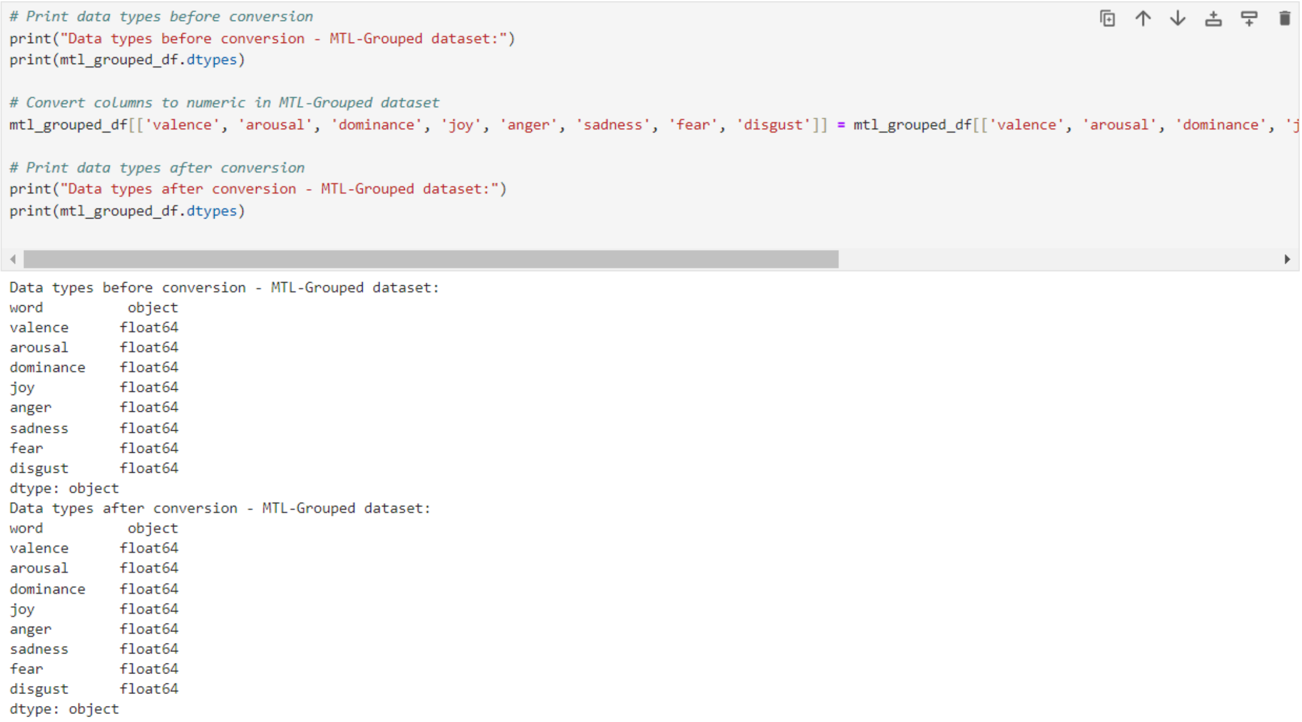


```
mtl_grouped_df.dropna(subset=['word'], inplace=True)
```



```
mtl_grouped_df.info()
```

And here we checked the datatypes of lexicons.

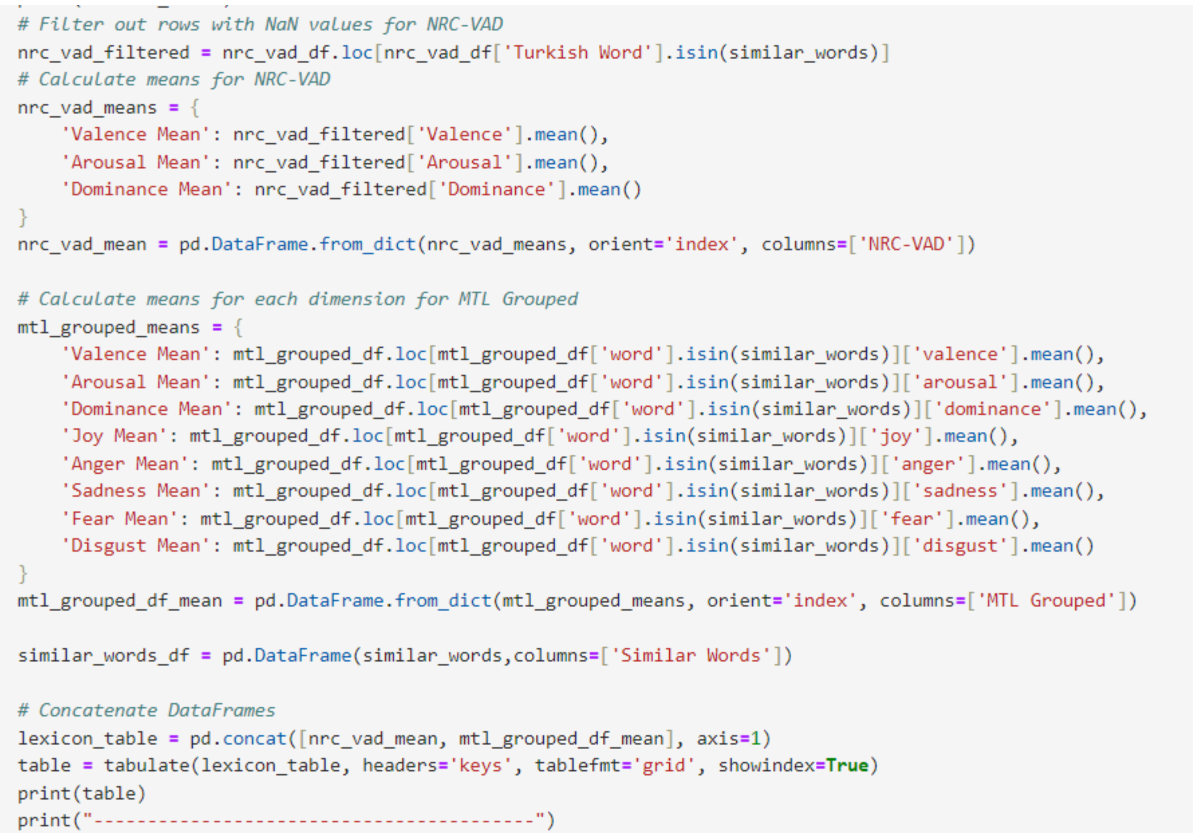


```
# Print data types before conversion
print("Data types before conversion - MTL-Grouped dataset:")
print(mtl_grouped_df.dtypes)

# Convert columns to numeric in MTL-Grouped dataset
mtl_grouped_df[['valence', 'arousal', 'dominance', 'joy', 'anger', 'sadness', 'fear', 'disgust']] = mtl_grouped_df[['valence', 'arousal', 'dominance', 'j

# Print data types after conversion
print("Data types after conversion - MTL-Grouped dataset:")
print(mtl_grouped_df.dtypes)
```

```
Data types before conversion - MTL-Grouped dataset:
word          object
valence      float64
arousal      float64
dominance    float64
joy          float64
anger        float64
sadness      float64
fear         float64
disgust      float64
dtype: object
Data types after conversion - MTL-Grouped dataset:
word          object
valence      float64
arousal      float64
dominance    float64
joy          float64
anger        float64
sadness      float64
fear         float64
disgust      float64
dtype: object
```

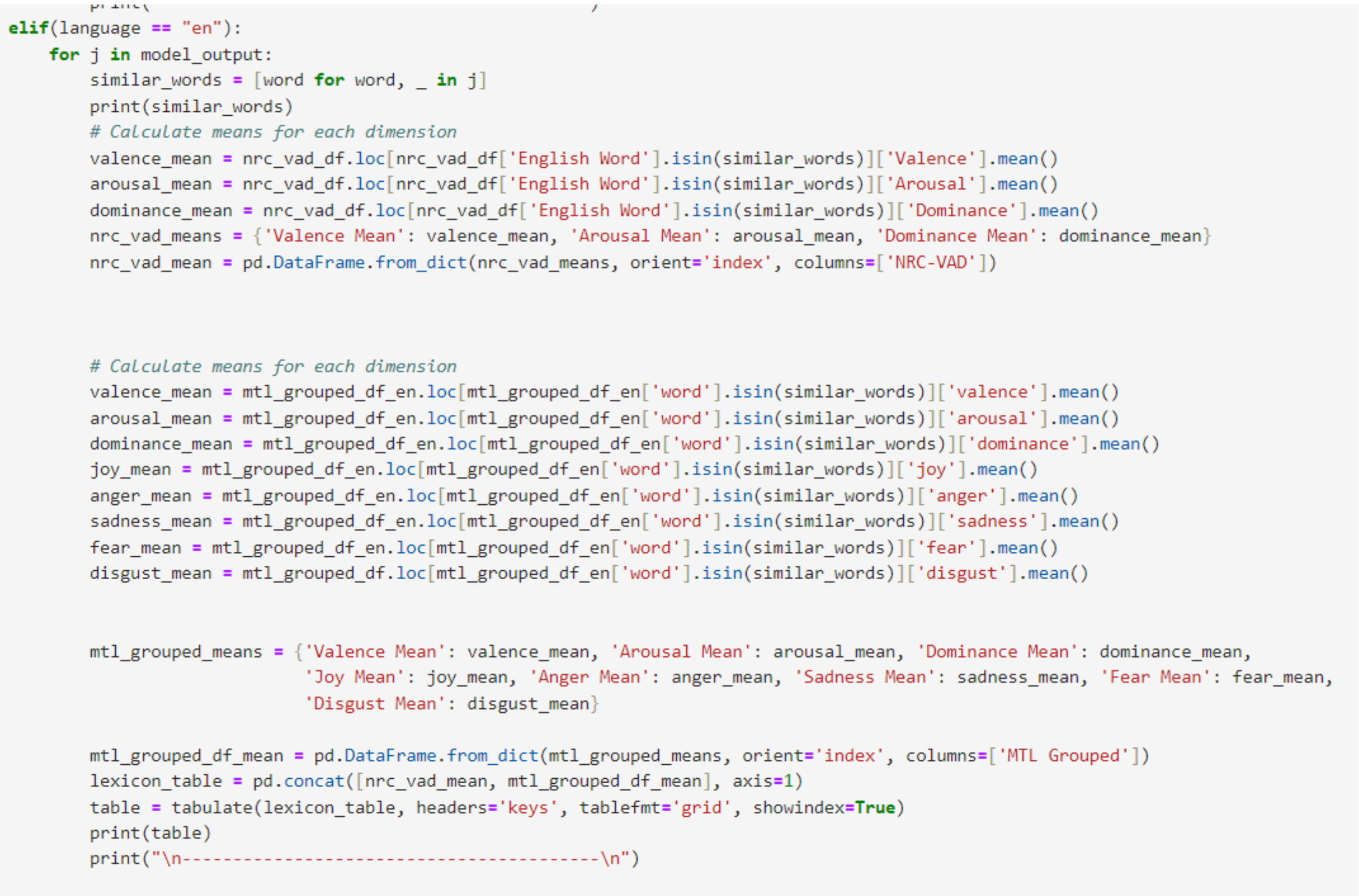
Here We Calculated the Mean Values of Similar Words and Created Tables:

Here we found mean values for Turkish Models.



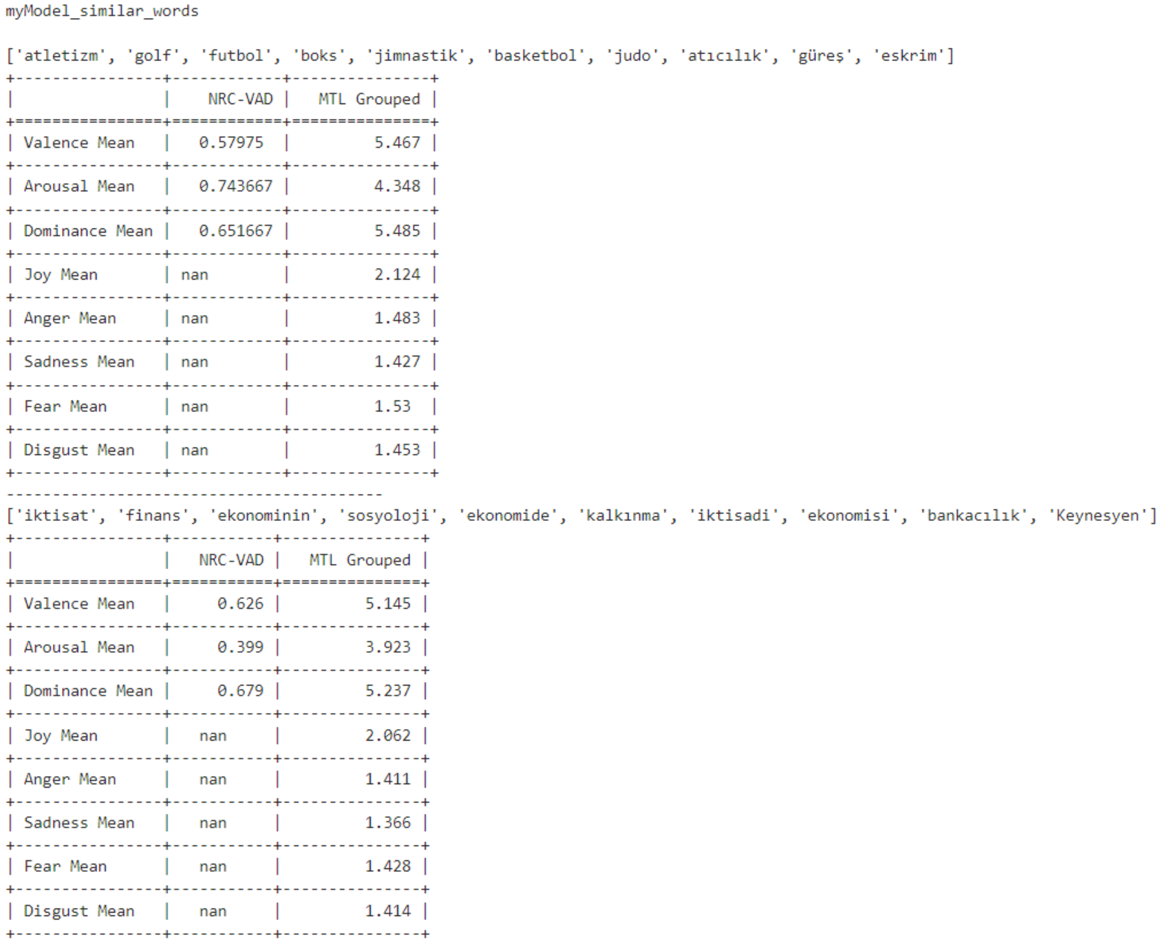
```
# Filter out rows with NaN values for NRC-VAD
nrc_vad_filtered = nrc_vad_df.loc[nrc_vad_df['Turkish Word'].isin(similar_words)]
# Calculate means for NRC-VAD
nrc_vad_means = {
    'Valence Mean': nrc_vad_filtered['Valence'].mean(),
    'Arousal Mean': nrc_vad_filtered['Arousal'].mean(),
    'Dominance Mean': nrc_vad_filtered['Dominance'].mean()
}
nrc_vad_mean = pd.DataFrame.from_dict(nrc_vad_means, orient='index', columns=['NRC-VAD'])

# Calculate means for each dimension for MTL Grouped
mtl_grouped_means = {
    'Valence Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['valence'].mean(),
    'Arousal Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['arousal'].mean(),
    'Dominance Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['dominance'].mean(),
    'Joy Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['joy'].mean(),
    'Anger Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['anger'].mean(),
    'Sadness Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['sadness'].mean(),
    'Fear Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['fear'].mean(),
    'Disgust Mean': mtl_grouped_df.loc[mtl_grouped_df['word'].isin(similar_words)]['disgust'].mean()
}
mtl_grouped_df_mean = pd.DataFrame.from_dict(mtl_grouped_means, orient='index', columns=['MTL Grouped'])

similar_words_df = pd.DataFrame(similar_words,columns=['Similar Words'])

# Concatenate DataFrames
lexicon_table = pd.concat([nrc_vad_mean, mtl_grouped_df_mean], axis=1)
table = tabulate(lexicon_table, headers='keys', tablefmt='grid', showindex=True)
print(table)
print("-----------------------------------------")
```

And here is the rest of the models that are in English.



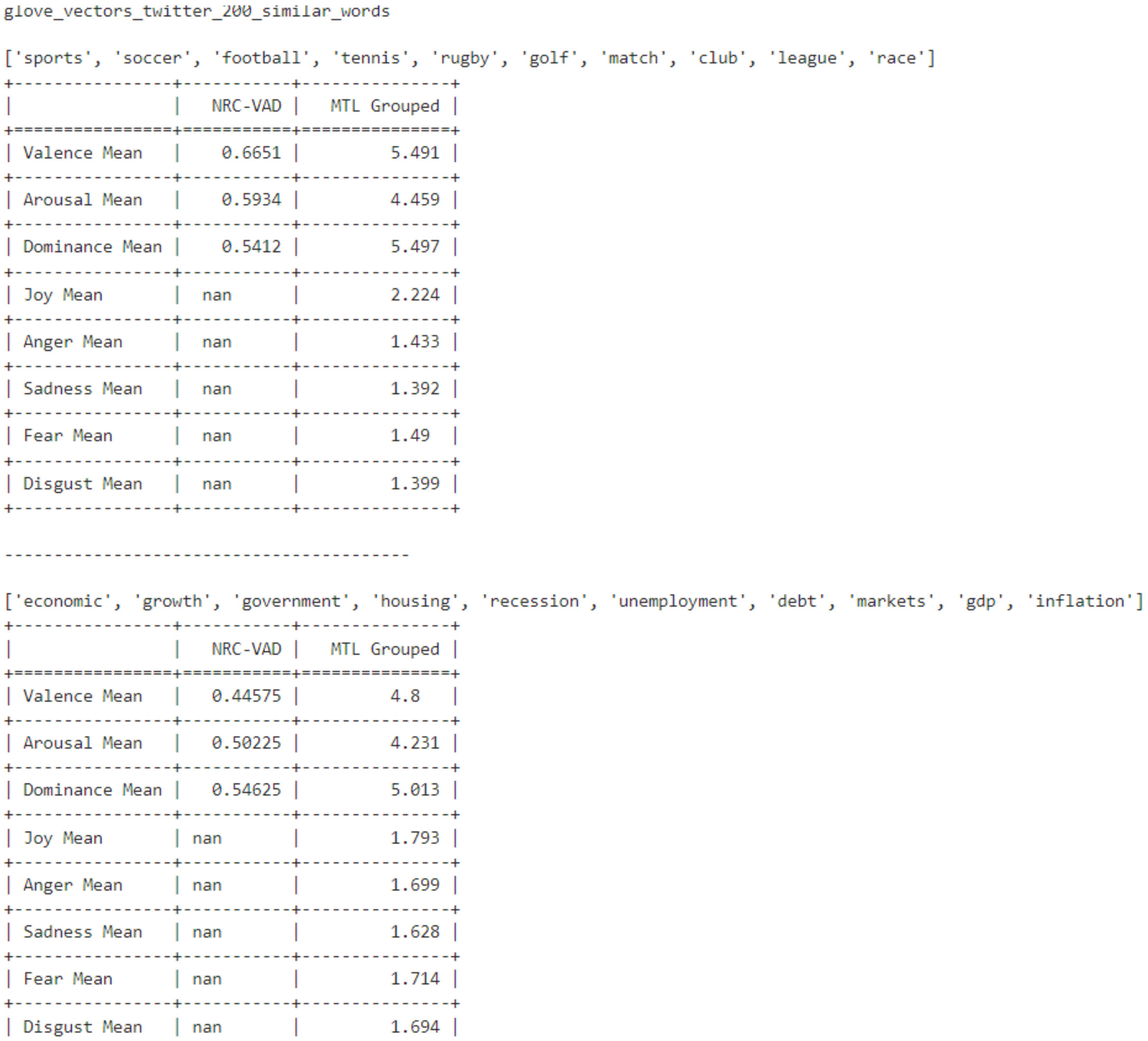
```
        print(                                                  )
elif(language == "en"):
    for j in model_output:
        similar_words = [word for word, _ in j]
        print(similar_words)
        # Calculate means for each dimension
        valence_mean = nrc_vad_df.loc[nrc_vad_df['English Word'].isin(similar_words)]['Valence'].mean()
        arousal_mean = nrc_vad_df.loc[nrc_vad_df['English Word'].isin(similar_words)]['Arousal'].mean()
        dominance_mean = nrc_vad_df.loc[nrc_vad_df['English Word'].isin(similar_words)]['Dominance'].mean()
        nrc_vad_means = {'Valence Mean': valence_mean, 'Arousal Mean': arousal_mean, 'Dominance Mean': dominance_mean}
        nrc_vad_mean = pd.DataFrame.from_dict(nrc_vad_means, orient='index', columns=['NRC-VAD'])



        # Calculate means for each dimension
        valence_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['valence'].mean()
        arousal_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['arousal'].mean()
        dominance_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['dominance'].mean()
        joy_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['joy'].mean()
        anger_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['anger'].mean()
        sadness_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['sadness'].mean()
        fear_mean = mtl_grouped_df_en.loc[mtl_grouped_df_en['word'].isin(similar_words)]['fear'].mean()
        disgust_mean = mtl_grouped_df.loc[mtl_grouped_df_en['word'].isin(similar_words)]['disgust'].mean()


        mtl_grouped_means = {'Valence Mean': valence_mean, 'Arousal Mean': arousal_mean, 'Dominance Mean': dominance_mean,
                             'Joy Mean': joy_mean, 'Anger Mean': anger_mean, 'Sadness Mean': sadness_mean, 'Fear Mean': fear_mean,
                             'Disgust Mean': disgust_mean}

        mtl_grouped_df_mean = pd.DataFrame.from_dict(mtl_grouped_means, orient='index', columns=['MTL Grouped'])
        lexicon_table = pd.concat([nrc_vad_mean, mtl_grouped_df_mean], axis=1)
        table = tabulate(lexicon_table, headers='keys', tablefmt='grid', showindex=True)
        print(table)
        print("\n----------------------------------------\n")
```

Here Are Some Examples of the Results of Each Model With Given Test Words And Their Similar Words:



```
myModel_similar_words

['atletizm', 'golf', 'futbol', 'boks', 'jimnastik', 'basketbol', 'judo', 'atıcılık', 'güreş', 'eskrim']
+----------------+-----------+---------------+
|                |   NRC-VAD |   MTL Grouped |
+================+===========+===============+
| Valence Mean   |  0.57975  |         5.467 |
+----------------+-----------+---------------+
| Arousal Mean   |  0.743667 |         4.348 |
+----------------+-----------+---------------+
| Dominance Mean |  0.651667 |         5.485 |
+----------------+-----------+---------------+
| Joy Mean       | nan       |         2.124 |
+----------------+-----------+---------------+
| Anger Mean     | nan       |         1.483 |
+----------------+-----------+---------------+
| Sadness Mean   | nan       |         1.427 |
+----------------+-----------+---------------+
| Fear Mean      | nan       |         1.53  |
+----------------+-----------+---------------+
| Disgust Mean   | nan       |         1.453 |
+----------------+-----------+---------------+

----------------------------------------

['iktisat', 'finans', 'ekonominin', 'sosyoloji', 'ekonomide', 'kalkınma', 'iktisadi', 'ekonomisi', 'bankacılık', 'Keynesyen']
+----------------+-----------+---------------+
|                |  NRC-VAD  |   MTL Grouped |
+================+===========+===============+
| Valence Mean   |     0.626 |         5.145 |
+----------------+-----------+---------------+
| Arousal Mean   |     0.399 |         3.923 |
+----------------+-----------+---------------+
| Dominance Mean |     0.679 |         5.237 |
+----------------+-----------+---------------+
| Joy Mean       |   nan     |         2.062 |
+----------------+-----------+---------------+
| Anger Mean     |   nan     |         1.411 |
+----------------+-----------+---------------+
| Sadness Mean   |   nan     |         1.366 |
+----------------+-----------+---------------+
| Fear Mean      |   nan     |         1.428 |
+----------------+-----------+---------------+
| Disgust Mean   |   nan     |         1.414 |
+----------------+-----------+---------------+
```

Here is another examples that comes from Glove Vectors Twitter 200.



```
glove_vectors_twitter_200_similar_words

['sports', 'soccer', 'football', 'tennis', 'rugby', 'golf', 'match', 'club', 'league', 'race']
+----------------+-----------+---------------+
|                |   NRC-VAD |   MTL Grouped |
+================+===========+===============+
| Valence Mean   |    0.6651 |         5.491 |
+----------------+-----------+---------------+
| Arousal Mean   |    0.5934 |         4.459 |
+----------------+-----------+---------------+
| Dominance Mean |    0.5412 |         5.497 |
+----------------+-----------+---------------+
| Joy Mean       |  nan      |         2.224 |
+----------------+-----------+---------------+
| Anger Mean     |  nan      |         1.433 |
+----------------+-----------+---------------+
| Sadness Mean   |  nan      |         1.392 |
+----------------+-----------+---------------+
| Fear Mean      |  nan      |         1.49  |
+----------------+-----------+---------------+
| Disgust Mean   |  nan      |         1.399 |
+----------------+-----------+---------------+

-----------------------------------------

['economic', 'growth', 'government', 'housing', 'recession', 'unemployment', 'debt', 'markets', 'gdp', 'inflation']
+----------------+-----------+---------------+
|                |   NRC-VAD |   MTL Grouped |
+================+===========+===============+
| Valence Mean   |   0.44575 |         4.8   |
+----------------+-----------+---------------+
| Arousal Mean   |   0.50225 |         4.231 |
+----------------+-----------+---------------+
| Dominance Mean |   0.54625 |         5.013 |
+----------------+-----------+---------------+
| Joy Mean       | nan       |         1.793 |
+----------------+-----------+---------------+
| Anger Mean     | nan       |         1.699 |
+----------------+-----------+---------------+
| Sadness Mean   | nan       |         1.628 |
+----------------+-----------+---------------+
| Fear Mean      | nan       |         1.714 |
+----------------+-----------+---------------+
| Disgust Mean   | nan       |         1.694 |
```

REFERENCES

1. https://github.com/akoksal/Turkish-Word2Vec/wiki/
2. https://dumps.wikimedia.org/trwiki/
3. https://drive.google.com/drive/folders/1IBMTAGtZ4DakSCyAoA4j7Ch0Ft1aFoww
4. https://saifmohammad.com/WebDocs/Lexicons/NRC-VAD-Lexicon.zip
5. https://zenodo.org/record/3756607/files/MTL_grouped.zip?download=1