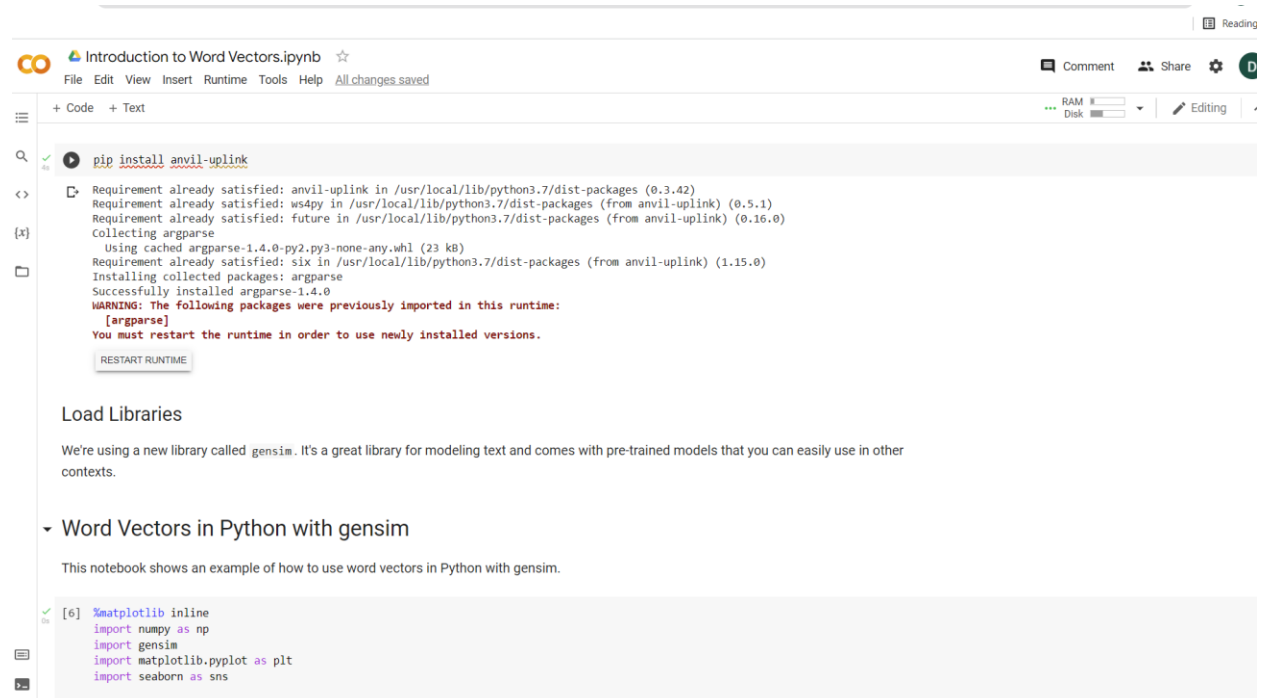


Google Colab link: <https://colab.research.google.com/drive/1jAlemA-T7pgjGmiby-oDul50D4gJxIAk#scrollTo=8B5bw-Om76VD>

1) Install the Anvil-uplink package:



The screenshot shows the Google Colab interface. At the top, the notebook is titled "Introduction to Word Vectors.ipynb". Below the title bar, there's a menu with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The main area displays a code cell where the command `pip install anvil-uplink` has been executed. The output shows that several requirements are already satisfied, and `argparse` is being collected. A warning message states: "WARNING: The following packages were previously imported in this runtime: [argparse]. You must restart the runtime in order to use newly installed versions." Below the code cell, there's a section titled "Load Libraries" with a paragraph about the `gensim` library. Further down, a section titled "Word Vectors in Python with gensim" contains a paragraph explaining the notebook's purpose. At the bottom, another code cell is partially visible, showing imports for `numpy`, `gensim`, `matplotlib.pyplot`, and `seaborn`.

```
pip install anvil-uplink
```

Requirement already satisfied: anvil-uplink in /usr/local/lib/python3.7/dist-packages (0.3.42)
Requirement already satisfied: ws4py in /usr/local/lib/python3.7/dist-packages (from anvil-uplink) (0.5.1)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from anvil-uplink) (0.16.0)
Collecting argparse
Using cached argparse-1.4.0-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from anvil-uplink) (1.15.0)
Installing collected packages: argparse
Successfully installed argparse-1.4.0
WARNING: The following packages were previously imported in this runtime:
[argparse]
You must restart the runtime in order to use newly installed versions.

RESTART RUNTIME

Load Libraries

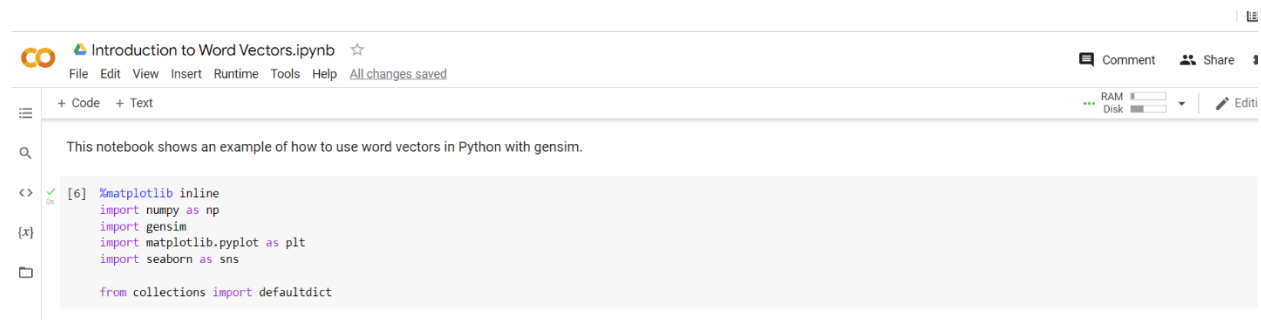
We're using a new library called `gensim`. It's a great library for modeling text and comes with pre-trained models that you can easily use in other contexts.

Word Vectors in Python with gensim

This notebook shows an example of how to use word vectors in Python with `gensim`.

```
[6] %matplotlib inline
import numpy as np
import gensim
import matplotlib.pyplot as plt
import seaborn as sns
```

2) Import a few key packages, including numpy and gensim

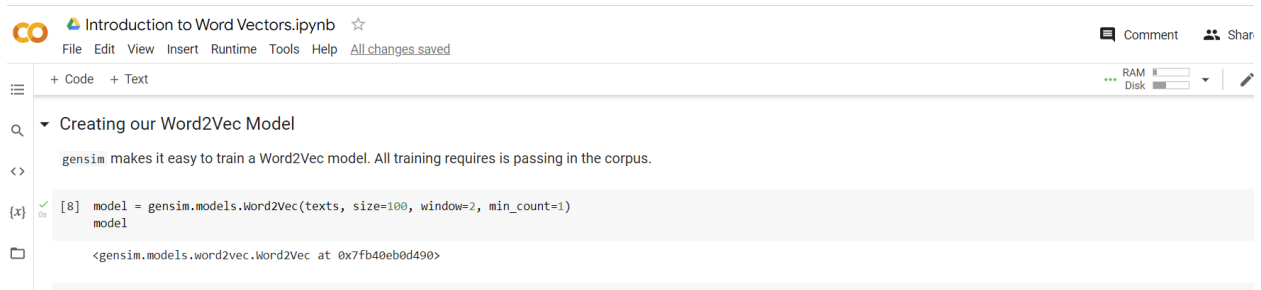


The screenshot shows the Google Colab interface with the same notebook. The code cell from the previous screenshot is still visible. Below it, a new code cell is shown, which continues the imports from the previous cell, adding `from collections import defaultdict`.

```
%matplotlib inline
import numpy as np
import gensim
import matplotlib.pyplot as plt
import seaborn as sns

from collections import defaultdict
```

3) Define the model from gensim library. The word2vec size is defined as 100.



```
Introduction to Word Vectors.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

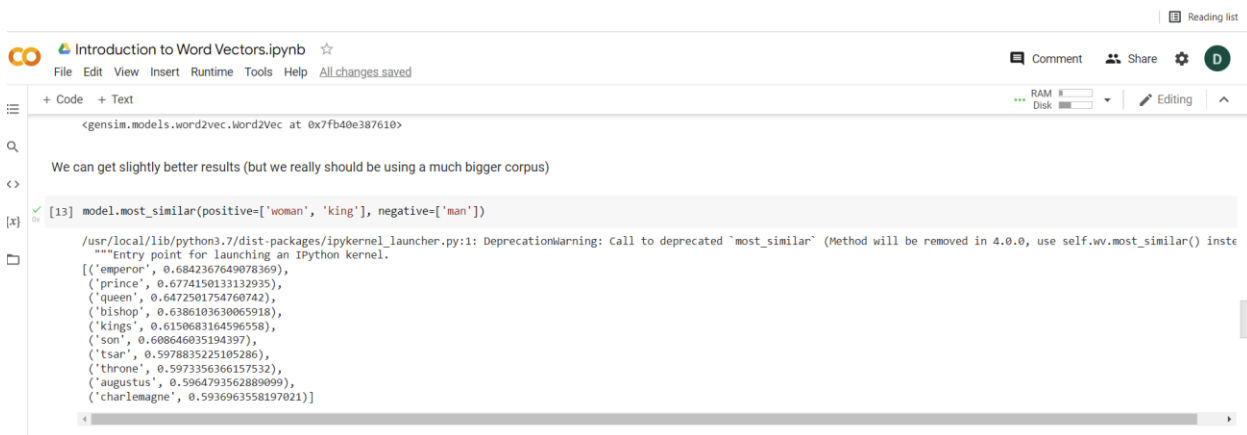
Creating our Word2Vec Model

gensim makes it easy to train a Word2Vec model. All training requires is passing in the corpus.

[8] model = gensim.models.Word2Vec(texts, size=100, window=2, min_count=1)
model

<gensim.models.word2vec.Word2Vec at 0x7fb40eb0d490>
```

4) Try the classic example in NLP: king – man + women = queen



```
Introduction to Word Vectors.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

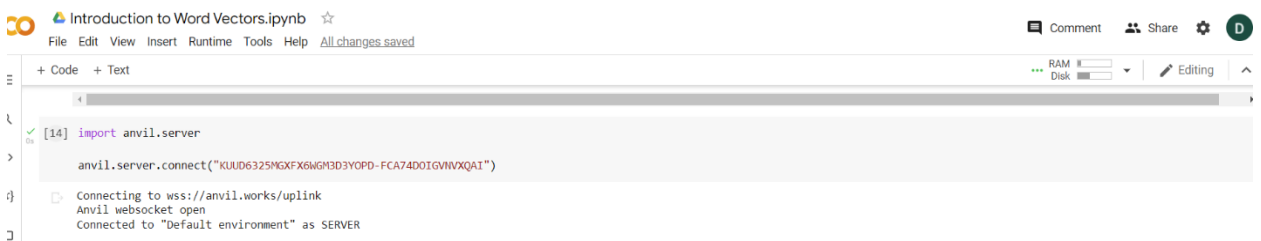
<gensim.models.word2vec.Word2Vec at 0x7fb40e387610>

We can get slightly better results (but we really should be using a much bigger corpus)

[13] model.most_similar(positive=['woman', 'king'], negative=['man'])

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `most_similar` (Method will be removed in 4.0.0, use self.wv.most_similar() inste
"""Entry point for launching an IPython kernel.
[('emperor', 0.6842367649078369),
 ('prince', 0.6774150133132935),
 ('queen', 0.6472501754760742),
 ('bishop', 0.6386103630065918),
 ('kings', 0.6150683164596558),
 ('son', 0.608646035194397),
 ('tsar', 0.5978835225105286),
 ('throne', 0.5973356366157532),
 ('augustus', 0.5964793562889099),
 ('charlemagne', 0.5936963558197021)]
```

5) import the anvil.server library, which is a key to connect the Google Colab notebook and the web-based front-end app.



```
Introduction to Word Vectors.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[14] import anvil.server

anvil.server.connect("KUUD6325MGXF6X6MGH3D3YQPD-FCA74D0IGVINXQAI")

Connecting to wss://anvil.works/uplink
Anvil websocket open
Connected to "Default environment" as SERVER
```

6) Explore the operations on two words, such as the average of two words and similarity between two words.

The screenshot shows a Jupyter Notebook titled "Introduction to Word Vectors.ipynb". The interface includes a top bar with a logo, title, and navigation icons. Below the title bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link for "All changes saved". The main area is divided into a left sidebar with icons for file explorer, search, and input/output, and a central code editor. The code editor contains three code cells. The first cell calculates the average of two word vectors and finds the most similar word in a pre-defined list. The second cell calculates the difference between two word vectors and finds the most similar word. The third cell calculates the similarity between two words. The output of the third cell is displayed below the code.

```
[ ] #average meaning of two words
vector3 = (vector1 + vector2)/2
model.wv.most_similar(positive=[vector3])

[ ] #subtraction of two words
vector4 = vector1 - vector2
model.wv.most_similar(positive=[vector4])

[ ] #calculate similarity between two words
model.wv.similarity(word1, word2)

0.70301175
```

7) This is the algorithm to summarize an input sentence or collection of words into one representative word:

The screenshot shows a Jupyter Notebook titled "Introduction to Word Vectors.ipynb". The interface is similar to the previous one, with a top bar, menu bar, and a central code editor. The code editor contains a single code cell with a Python script for summarizing a sentence. The script imports necessary libraries, takes a sentence as input, tokenizes it, and then iterates through the words to find the most similar word to the average vector of all words in the sentence. The output of the script is printed at the end.

```
#new Algorithm
from gensim.summarization.textcleaner import tokenize_by_word
from scipy.spatial import distance
import numpy as np
sentence = input("Enter sentence here: ")
words = tokenize_by_word(sentence)
vector1 = 0
words2 = tokenize_by_word(sentence)
dst1 = 100000
for i in words:
    vector1 += model.wv[i]
vector1 = vector1/(len(sentence))
for i in words2:
    dst = distance.euclidean(model.wv[i], vector1)
    if dst < dst1:
        dst1 = dst
        j = i
print(j)
model.most_similar(positive=[vector1], topn=1)
```

Code:

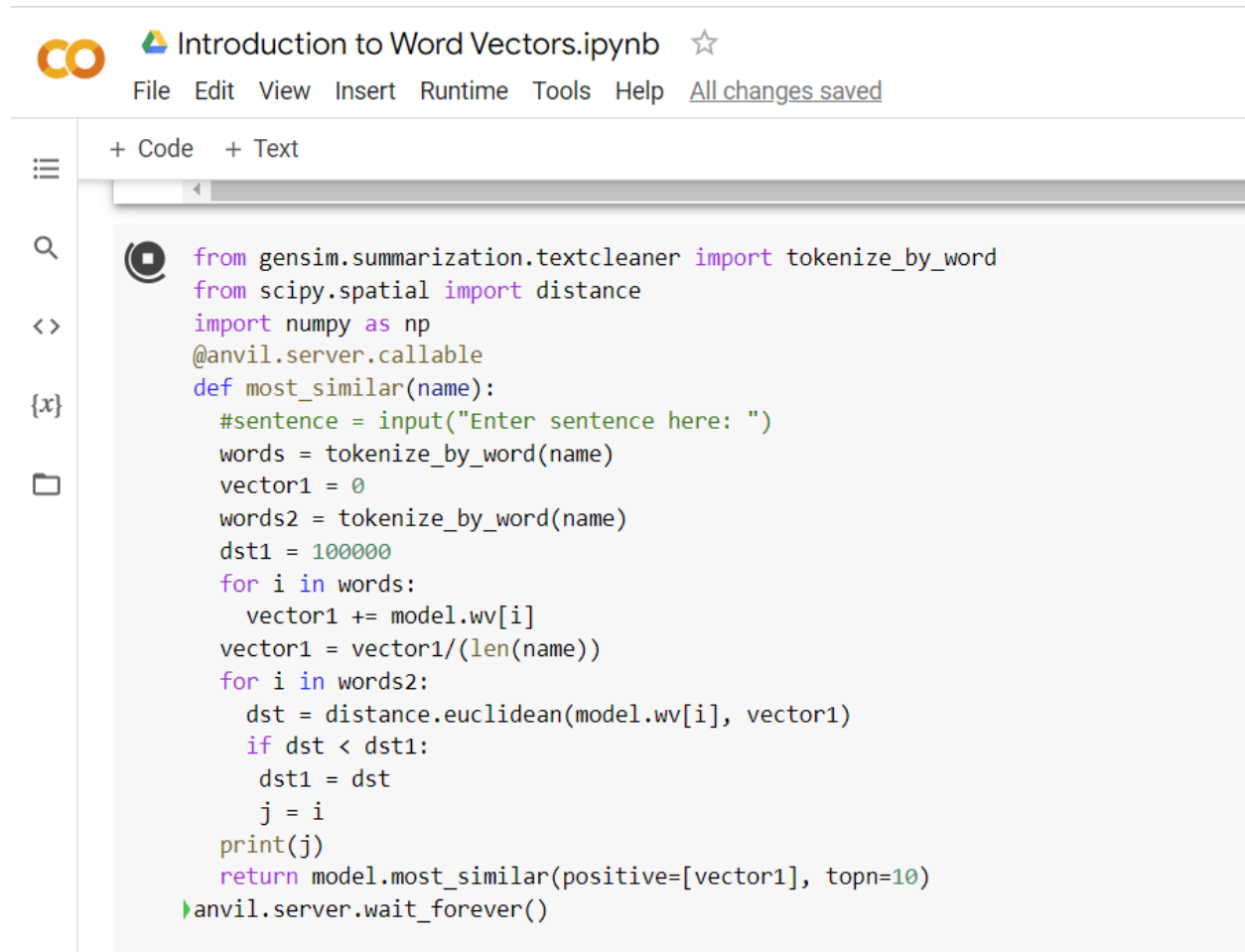
```
from gensim.summarization.textcleaner import tokenize_by_word
from scipy.spatial import distance
import numpy as np
sentence = input("Enter sentence here: ")
words = tokenize_by_word(sentence)
vector1 = 0
words2 = tokenize_by_word(sentence)
dst1 = 100000
for i in words:
    vector1 += model.wv[i]
vector1 = vector1/(len(sentence))
```

```

for i in words2:
    dst = distance.euclidean(model.wv[i], vector1)
    if dst < dst1:
        dst1 = dst
        j = i
print(j)
model.most_similar(positive=[vector1], topn=1)

```

8) This is the callback function that receives sentence inputs from the front-end app and returns output to the front-end app:



```

from gensim.summarization.textcleaner import tokenize_by_word
from scipy.spatial import distance
import numpy as np
@anvil.server.callable
def most_similar(name):
    #sentence = input("Enter sentence here: ")
    words = tokenize_by_word(name)
    vector1 = 0
    words2 = tokenize_by_word(name)
    dst1 = 100000
    for i in words:
        vector1 += model.wv[i]
    vector1 = vector1/(len(name))
    for i in words2:
        dst = distance.euclidean(model.wv[i], vector1)
        if dst < dst1:
            dst1 = dst
            j = i
    print(j)
    return model.most_similar(positive=[vector1], topn=10)
anvil.server.wait_forever()

```

Code:

```

from gensim.summarization.textcleaner import tokenize_by_word
from scipy.spatial import distance
@anvil.server.callable
def most_similar(name):
    words = tokenize_by_word(name)
    vector1 = 0

```

```
words2 = tokenize_by_word(name)
dst1 = 100000
for i in words:
    vector1 += model.wv[i]
vector1 = vector1/(len(name))
for i in words2:
    dst = distance.euclidean(model.wv[i], vector1)
    if dst < dst1:
        dst1 = dst
        j = i
print(j)
if len(name) < 10:
    return model.most_similar(j, topn=10)
else:
    return model.most_similar(j, topn=15)
anvil.server.wait_forever()
```