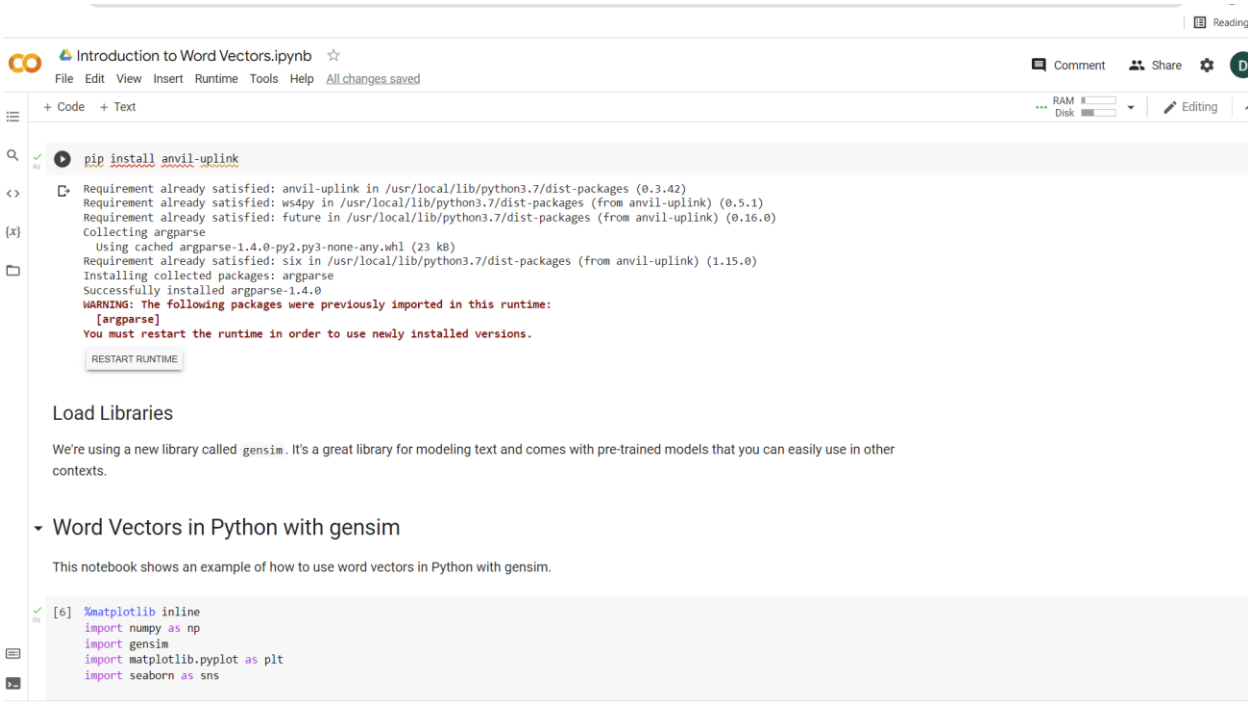
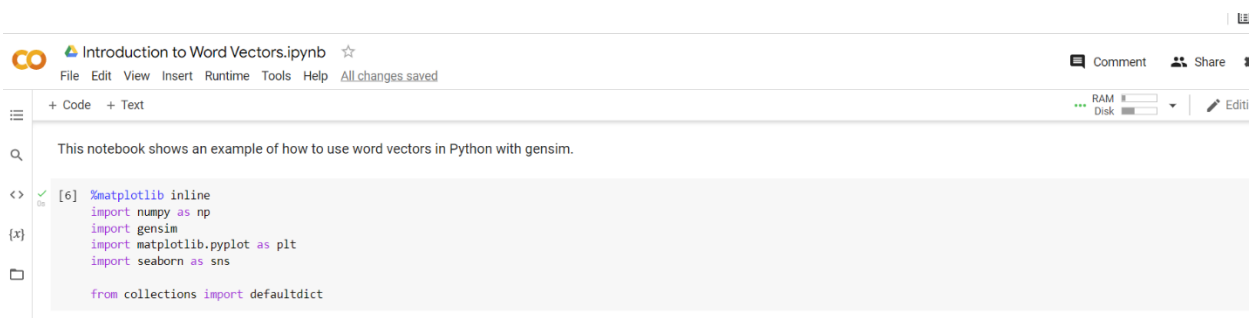
Google Colab link: https://colab.research.google.com/drive/1jAIemA-T7pgjGmiby-oDuI50D4gJxlAk#scrollTo=8B5bw-Om76VD

1) Install the Anvil-uplink package:



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



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

Introduction to Word Vectors.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

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

Reading list

Introduction to Word Vectors.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
<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', 0.597883522510528 6),
 ('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

Comment Share

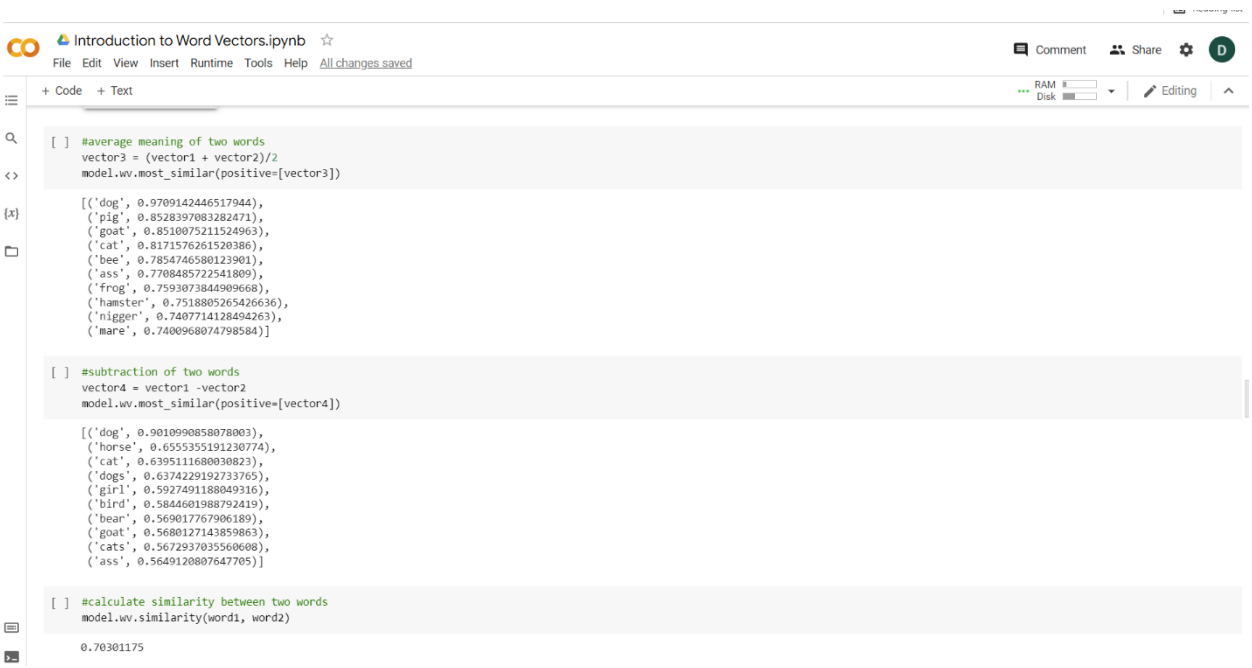
+ Code + Text

RAM Disk Editing

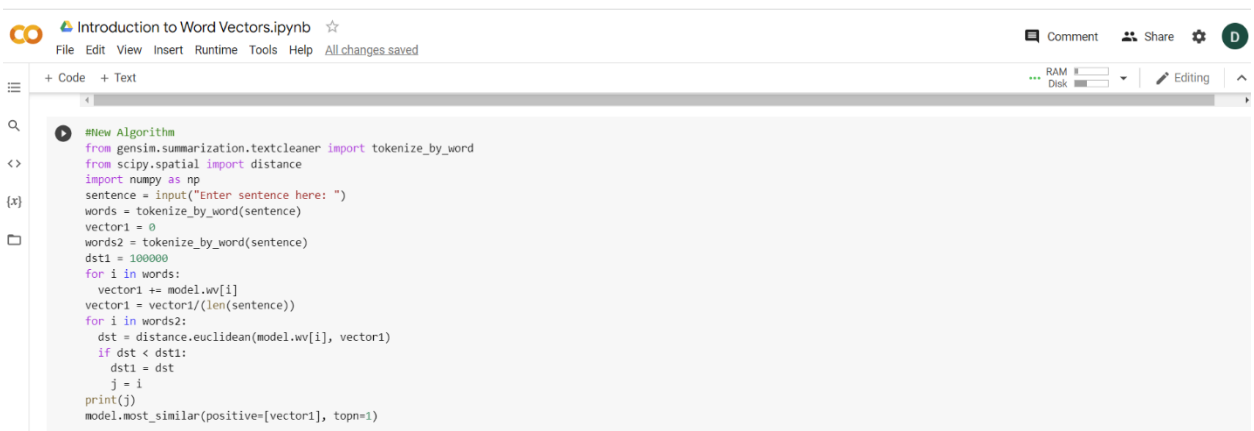
```
[14] import anvil.server

     anvil.server.connect("KUUD6325MGXFX6WGM3D3YOPD-FCA74DOIGVNVXQAI")
```

```
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 works and similarity between two words.



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

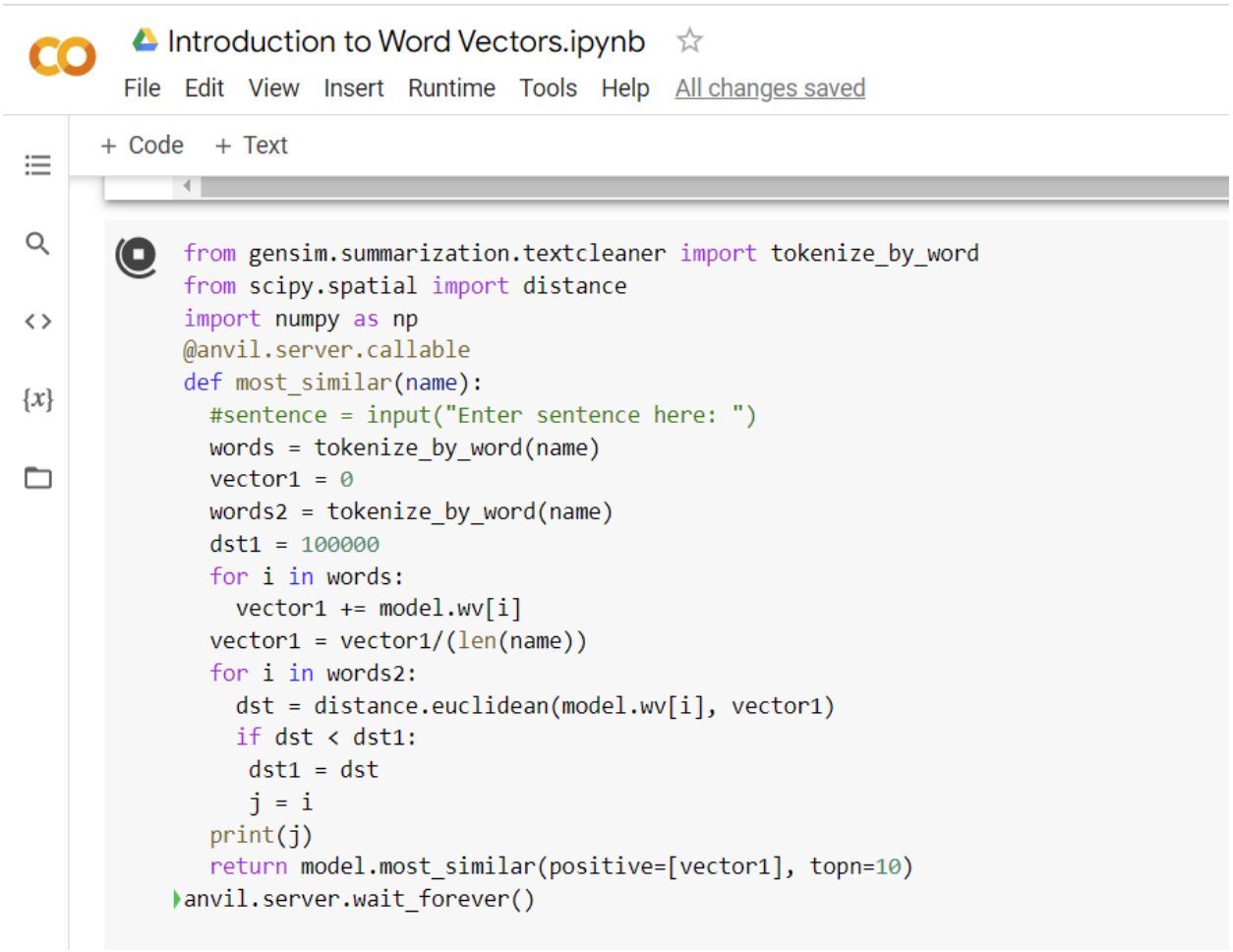


Code:

```
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:



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()
```