Lab No. 13

Natural Language Processing (NLP)

This laboratory session introduces students to Word2Vec, a popular technique in Natural Language Processing (NLP) used to represent words as meaningful dense vectors. Using textual data from the *Game of Thrones* series, students will learn how word embeddings capture semantic relationships between words based on their context. Through preprocessing, model training, and similarity analysis, this lab helps students understand how machines learn word meanings and relationships from large text corpora in an unsupervised manner.

LAB Objectives:

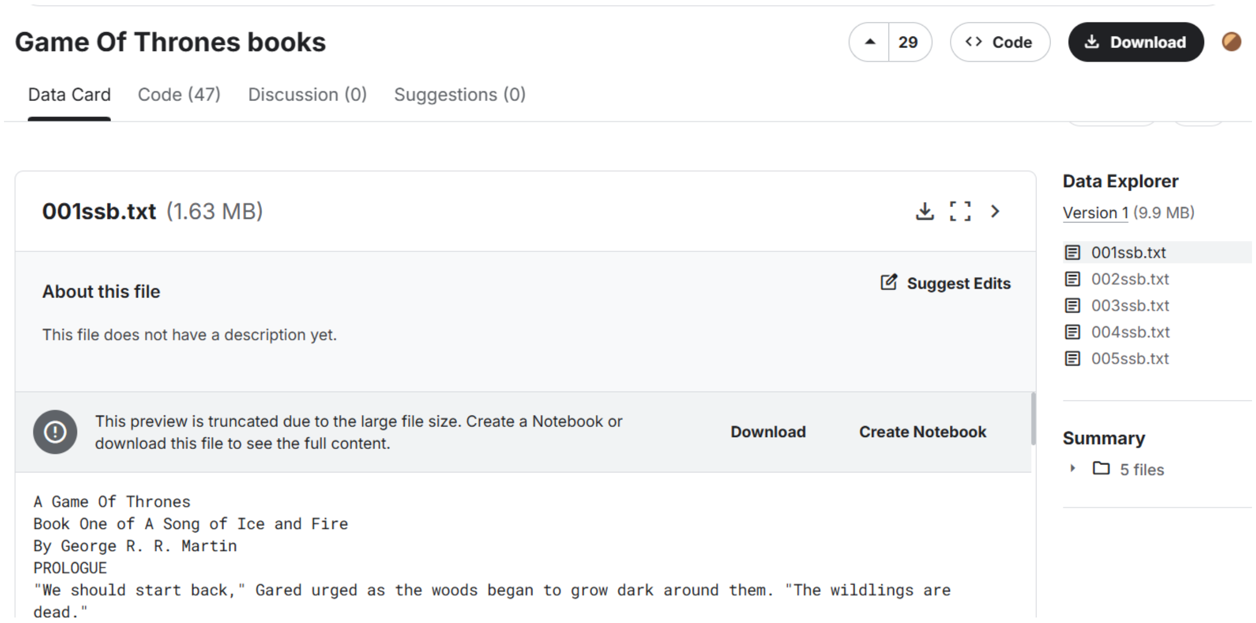
- Understand **distributional semantics**
- Learn how **Word2Vec** converts words into dense vectors
- Apply **Word2Vec (Skip-Gram / CBOW)** on real textual data (Game of Thrones)
- Explore **word similarity, analogy, and visualization**

game-of-thrones-word2vec

word2vec applied on game of thrones data

Dataset Link: https://www.kaggle.com/khulasasndh/game-of-thrones-books

Download the data set from Kaggle



001ssb.txt (1.63 MB)

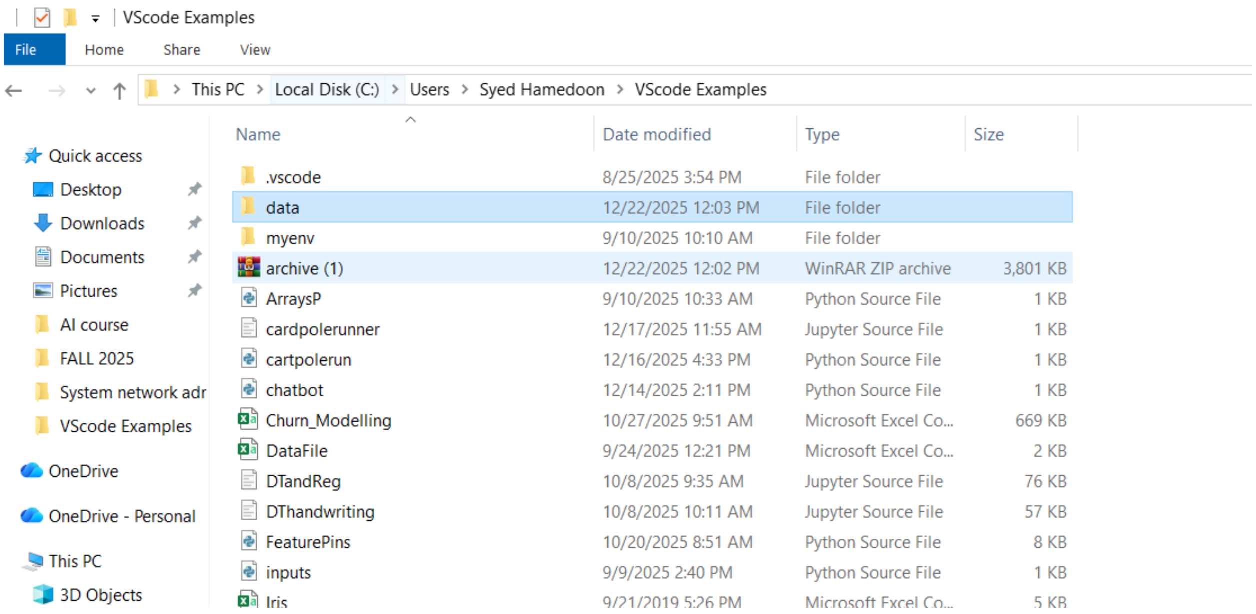
About this file

This file does not have a description yet.

This preview is truncated due to the large file size. Create a Notebook or download this file to see the full content.

```
A Game Of Thrones
Book One of A Song of Ice and Fire
By George R. R. Martin
PROLOGUE
"We should start back," Gared urged as the woods began to grow dark around them. "The wildlings are
dead."
```

Add the dataset in folder data where VS code directory present



Name	Date modified	Type	Size
.vscode	8/25/2025 3:54 PM	File folder	
data	12/22/2025 12:03 PM	File folder	
myenv	9/10/2025 10:10 AM	File folder	
archive (1)	12/22/2025 12:02 PM	WinRAR ZIP archive	3,801 KB
ArraysP	9/10/2025 10:33 AM	Python Source File	1 KB
cardpolerunner	12/17/2025 11:55 AM	Jupyter Source File	1 KB
cartpolerun	12/16/2025 4:33 PM	Python Source File	1 KB
chatbot	12/14/2025 2:11 PM	Python Source File	1 KB
Churn_Modelling	10/27/2025 9:51 AM	Microsoft Excel Co...	669 KB
DataFile	9/24/2025 12:21 PM	Microsoft Excel Co...	2 KB
DTandReg	10/8/2025 9:35 AM	Jupyter Source File	76 KB
DThandwriting	10/8/2025 10:11 AM	Jupyter Source File	57 KB
FeaturePins	10/20/2025 8:51 AM	Python Source File	8 KB
inputs	9/9/2025 2:40 PM	Python Source File	1 KB
Iris	9/21/2019 5:26 PM	Microsoft Excel Co...	5 KB

Code in jupter VS code:

```
import numpy as np
import pandas as pd
```

```
!pip install gensim
```

```
import gensim
import os
```

```
!pip install nltk
```

```
data = "C:/Users/Syed Hamedoon/VScode Examples/data"
```

```
import nltk

nltk.download('punkt')
nltk.download('punkt_tab')
```

```
import os
from nltk import sent_tokenize
from gensim.utils import simple_preprocess

DATA_PATH = r"C:\Users\Syed Hamedoon\VScode Examples\data"

story = []

for filename in os.listdir(DATA_PATH):
    if filename.endswith(".txt"):
        file_path = os.path.join(DATA_PATH, filename)

        try:
            with open(file_path, "r", encoding="utf-8") as f:
                corpus = f.read()
        except UnicodeDecodeError:
            with open(file_path, "r", encoding="cp1252") as f:
                corpus = f.read()

        for sent in sent_tokenize(corpus):
            story.append(simple_preprocess(sent))
```

```
print(len(story))
print(story[:2])
```

```
model = gensim.models.Word2Vec(
    window=10,
    min_count=2
)
```

```
model.build_vocab(story)
```

```
model.train(story, total_examples=model.corpus_count, epochs=model.epochs)
```

```
model.wv.most_similar('daenerys')
```

```
model.wv.doesnt_match(['jon','rikon','robb','arya','sansa','bran'])
```

```
model.wv.doesnt_match(['cersei', 'jaime', 'bronn', 'tyrion'])
```

```
model.wv['king']
```

```
model.wv.similarity('arya','sansa')
model.wv.similarity('tywin','sansa')
```

```
model.wv.get_normed_vectors()
```

```
y = model.wv.index_to_key
```

```
len(y)
```

```
y
```

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=3)
```

```
X = pca.fit_transform(model.wv.get_normed_vectors())
```

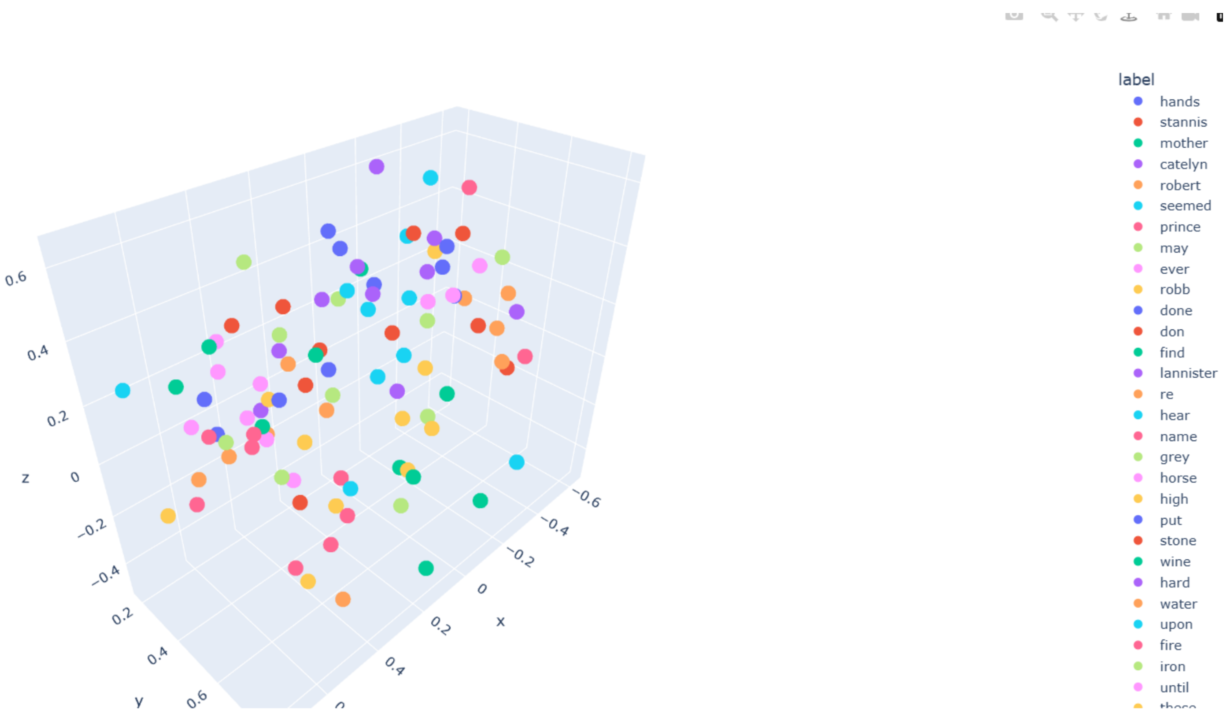
```
X.shape
```

```
!pip install --upgrade nbformat
```

```
import pandas as pd
import plotly.express as px
import plotly.io as pio

pio.renderers.default = "browser"  # ← IMPORTANT

df = pd.DataFrame(X[200:300], columns=["x", "y", "z"])
df["label"] = y[200:300]
```

```
fig = px.scatter_3d(
    df,
    x="x",
    y="y",
    z="z",
    color="label"
)

fig.show()
```

Output:



LAB Questions

1. What is the core idea behind Word2Vec?
2. Difference between CBOW and Skip-Gram?
3. Why is one-hot encoding inefficient?

4. Why do character names appear close in vector space?
5. How does window size affect semantic learning?
6. Why might rare characters have poor embeddings?

---

7. Which model performed better: CBOW or Skip-Gram? Why?
8. What happens if vector size is too small or too large?
9. Can Word2Vec understand word meaning without labels? Explain.

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	