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:

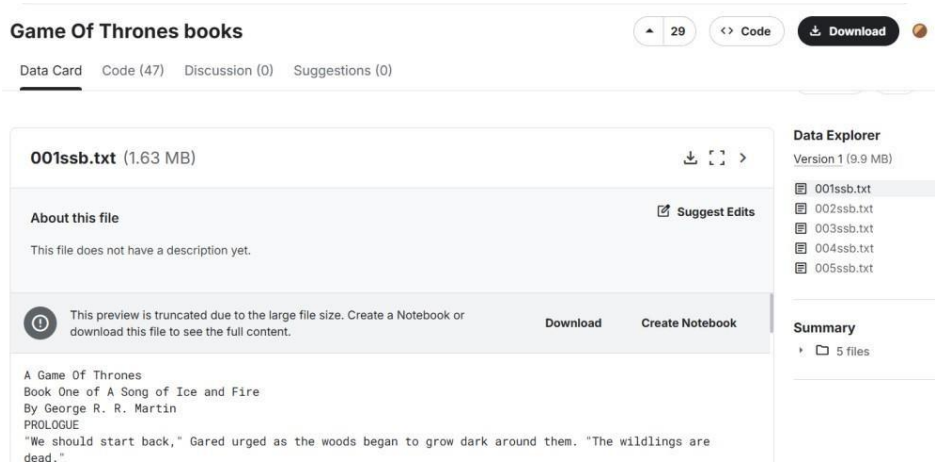
- Understand **distributional semantics**
- Learn how **Word2Vec** converts words into dense vectors
- Apply **Word2Vec (Skip-Gram / CBOW)** on real textual data (Game of Thrones)
- Explore word similarity, analogy, and visualization

game-of-thrones-word2vec

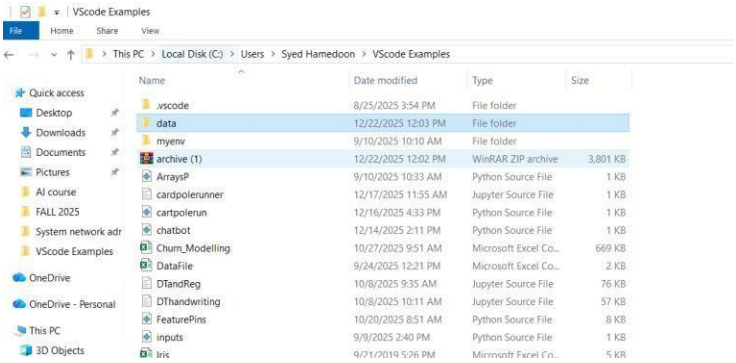
word2vec applied on game of thrones data

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

Download the data set from Kaggle



Add the dataset in folder data where VS code directory present



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())
```

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

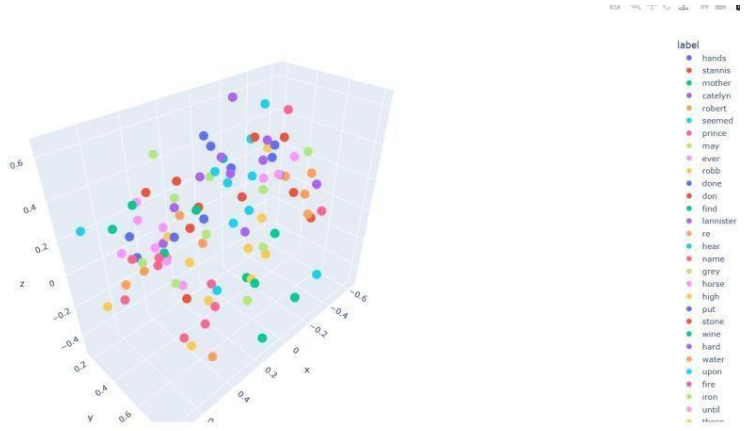
```
import pandas as pd
import plotly.express as px
import plotly.io as pio

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?

Answer:

The core idea behind **Word2Vec** is that **words appearing in similar contexts tend to have similar meanings.**
Word2Vec learns **dense numerical vectors (embeddings)** for words by analyzing surrounding words in large text data.
These vectors capture **semantic and syntactic relationships** between words.

Example:
king and queen appear in similar contexts → their vectors become close in space

2. Difference between CBOW and Skip-Gram?

Answer:

Feature	**CBOW**	**Skip-Gram**
Prediction	Predicts target word from context	Predicts context from target word
Speed	Faster	Slower
Best for	Large datasets	Small datasets
Rare words	Poor performance	Better performance

3. Why is one-hot encoding inefficient?

Answer:

One-hot encoding is inefficient because:

- Vectors are **very large** (equal to vocabulary size)
- Mostly **zeros,** wasting memory
- Cannot capture **semantic similarity**
- No relationship between words

Example:
king and queen have completely different vectors although meanings are related.

---

4. Why do character names appear close in vector space?

Answer:

Character names appear close because:

- They occur in **similar contexts**
- Appear together in scenes, families, or storylines
- Share roles (e.g., Stark family members)

Example:
arya, sansa, bran → frequently mentioned together → similar embeddings

5. How does window size affect semantic learning?

Answer:

- Small window size (2–5):
  - Learns syntactic relationships
  - Focuses on nearby words
- Large window size (8–15):
  - Learns semantic relationships
  - Captures broader context

6. Why might rare characters have poor embeddings?

Answer:

Rare characters have poor embeddings because:

- They appear **few times** in the corpus
- Model cannot learn enough context
- Limited co-occurrence with other words

---

7. Which model performed better: CBOW or Skip-Gram? Why?

Answer:

Skip-Gram performed better for this dataset because:

- Game of Thrones has many **rare character names**
- Skip-Gram handles rare words more effectively
- Produces more meaningful embeddings for names

8. What happens if vector size is too small or too large?

Answer:

Too small vector size:

- Cannot capture enough semantic information
- Poor similarity results

Too large vector size:

- Overfitting
- Slower training
- Higher memory usage

9. Can Word2Vec understand word meaning without labels? Explain.

Answer:

Yes, **Word2Vec learns word meaning without labels** because:

- It uses **unsupervised learning**
- Learns from **word co-occurrence patterns**
- Discovers semantic relationships automatically

Lab Assessment:

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

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency 10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
G	**Deliver y 10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awarenesso f Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPsand protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and isnon-compliant to the guidelines