

Artificial Intelligence Mini Project

Next Word Predictor

Using Python

Samriddho Das (119)

Madura Sankar S. (114)

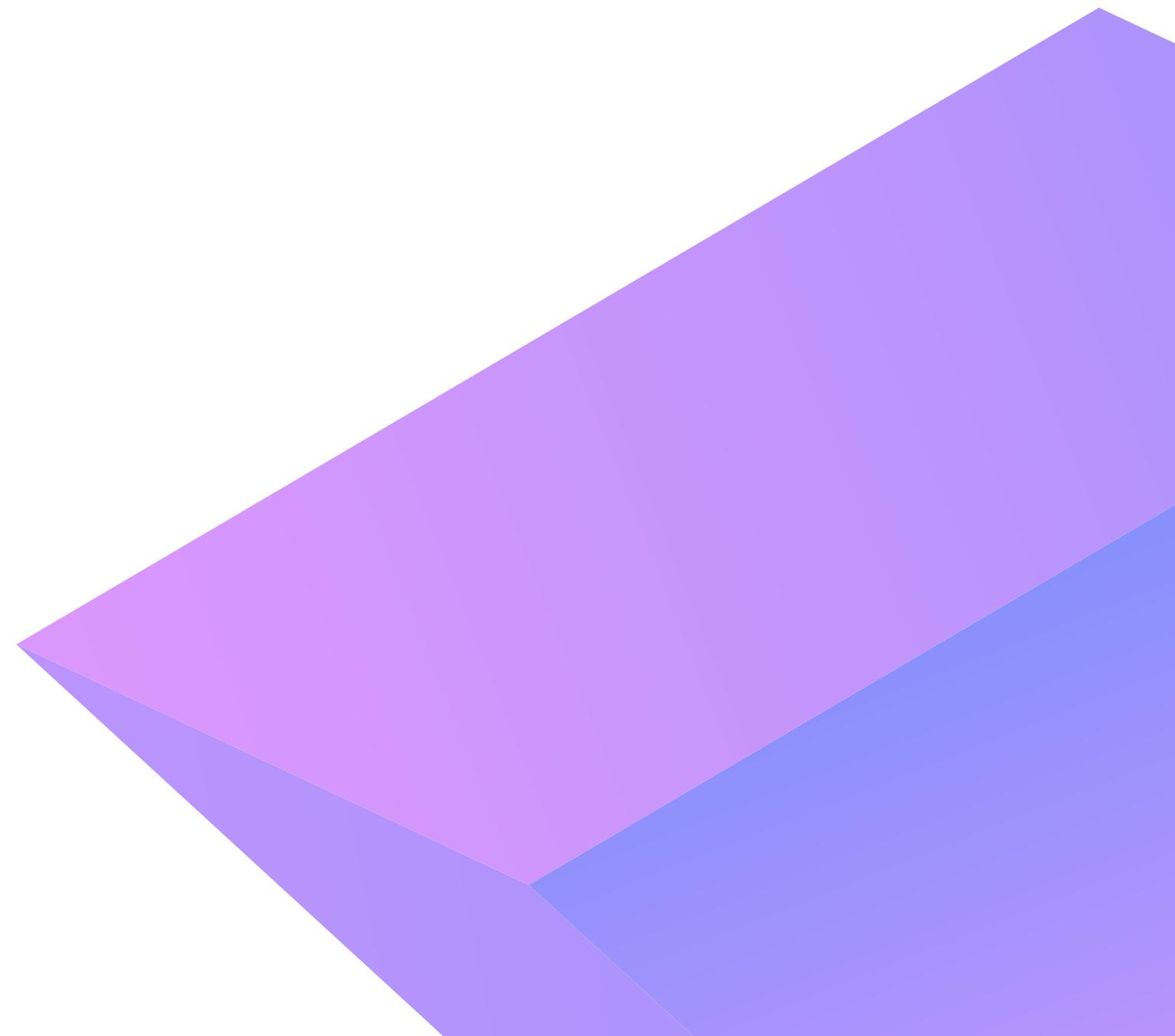
Ashwin S. (109)



You must therefore have seen the prediction of the next word on the keyboard of our smartphones which is most of the time precise. How does our keyboard predict the next word? In this article, I will walk you through the task of the next word prediction with Python.



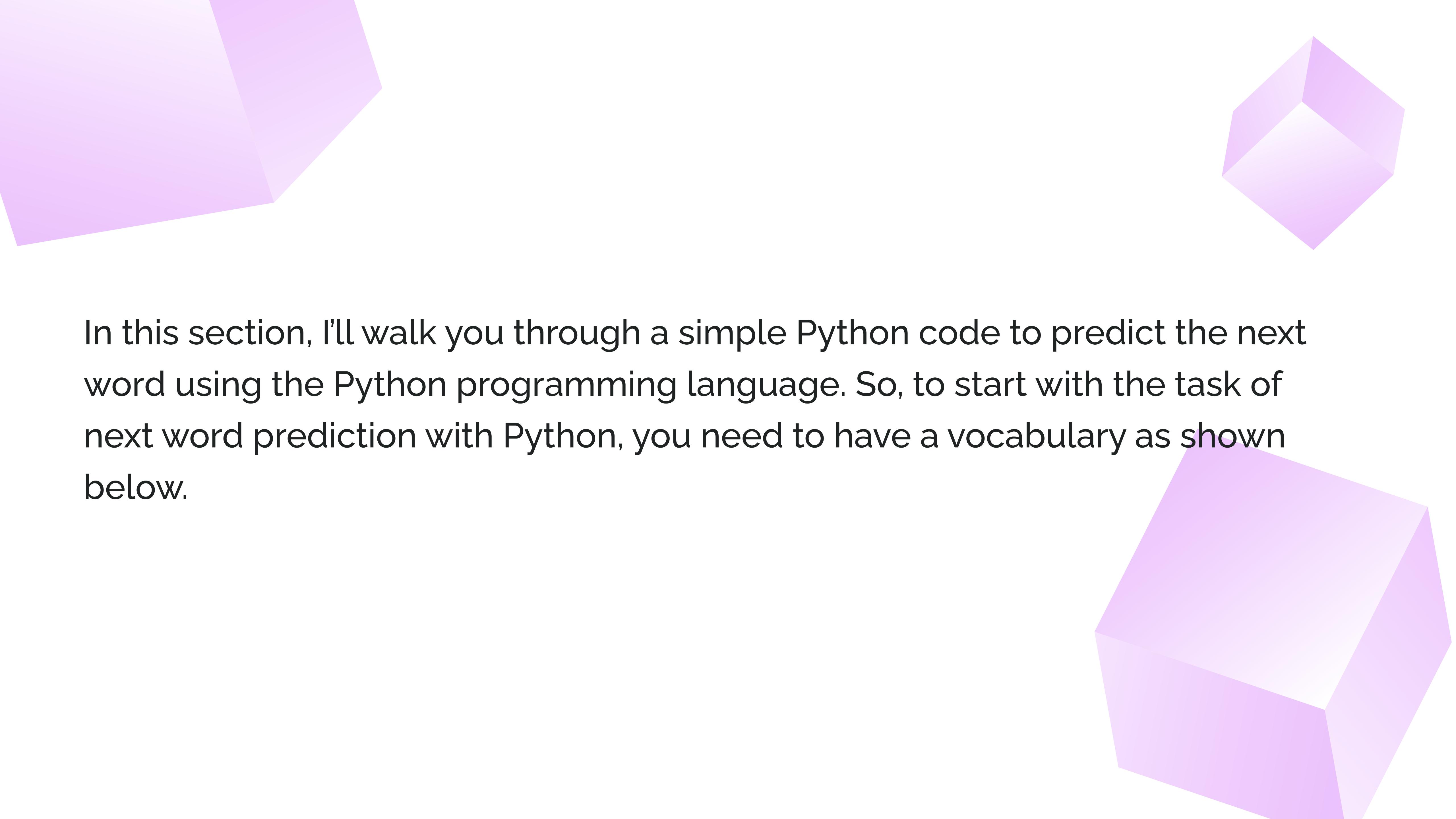
Predicting the next word is a machine learning task, but we can also achieve good results using only Python. Below are the steps you need to take when writing code to predict the next word with Python:



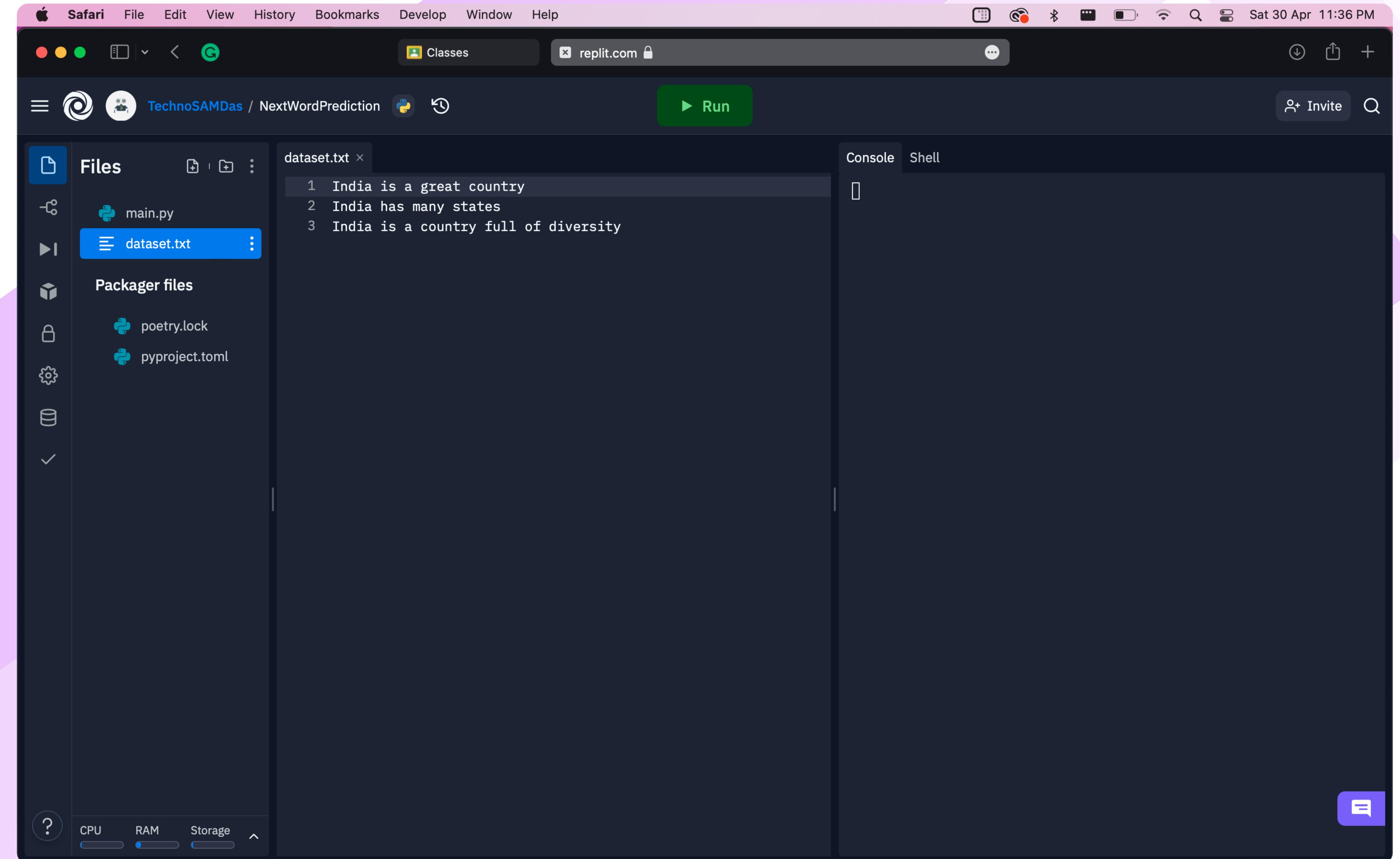


- First, divide the sentence into words
- Then select the last word of the sentence
- Then find the probability of the last word by looking at the vocabulary (dataset)
- Then, finally, select the next possible word

Hands-on Demo

The background features three large, semi-transparent pink geometric shapes: a hexagon on the left, a cube-like shape on the right, and a larger, irregular polygon at the bottom right.

In this section, I'll walk you through a simple Python code to predict the next word using the Python programming language. So, to start with the task of next word prediction with Python, you need to have a vocabulary as shown below.



Step 1:

Type the initial portion of any line and press enter. The code will predict a next word for the line by referencing the given dataset.txt

The screenshot shows the Replit IDE interface. The top bar includes the Safari menu, a tab for 'Classes', and the URL 'replit.com'. The main area has tabs for 'Files' and 'main.py'. The 'main.py' tab is active, displaying the following code:

```
1 import numpy as np
2
3 lexicon = {}
4
5 def update_lexicon(current : str, next_word : str) -> None:
6     # Add the input word to the lexicon if it is there yet.
7     if current not in lexicon:
8         lexicon.update({current: {next_word: 1}})
9         return
10
11     # Receive the probabilities of the input word.
12     options = lexicon[current]
13
14     # Check if the output word is in the probability list.
15     if next_word not in options:
16         options.update({next_word : 1})
17     else:
18         options.update({next_word : options[next_word] + 1})
19
20     # Update the lexicon
21     lexicon[current] = options
22
23 # Populate lexicon
24 with open('dataset.txt', 'r') as dataset:
25     for line in dataset:
26         words = line.strip().split(' ')
27         for i in range(len(words) - 1):
28             update_lexicon(words[i], words[i+1])
29
30 # Adjust probability
31 for word, transition in lexicon.items():
32     transition = dict((key, value / sum(transition.values())))
33     for key, value in transition.items():
            lexicon[word] = transition
```

To the right of the code editor is the 'Console' tab, which displays the output of the program's execution. The console shows the following text:

```
> india is a
india is a great
>
```

Step 2:

Now copy the output line. Run the code once again and paste it. The code will add a new word to the incomplete sentence. Repeat this process till the whole sentence is complete

The screenshot shows the Replit IDE interface. On the left, the 'Files' sidebar lists 'main.py' (selected), 'dataset.txt', and 'Packager files' (poetry.lock, pyproject.toml). The main workspace displays the 'main.py' code:

```
1 import numpy as np
2
3 lexicon = {}
4
5 def update_lexicon(current : str, next_word : str) -> None:
6     # Add the input word to the lexicon if it is there yet.
7     if current not in lexicon:
8         lexicon.update({current: {next_word: 1}})
9         return
10
11     # Receive the probabilities of the input word.
12     options = lexicon[current]
13
14     # Check if the output word is in the probability list.
15     if next_word not in options:
16         options.update({next_word : 1})
17     else:
18         options.update({next_word : options[next_word] + 1})
19
20     # Update the lexicon
21     lexicon[current] = options
22
23 # Populate lexicon
24 with open('dataset.txt', 'r') as dataset:
25     for line in dataset:
26         words = line.strip().split(' ')
27         for i in range(len(words) - 1):
28             update_lexicon(words[i], words[i+1])
29
30 # Adjust probability
31 for word, transition in lexicon.items():
32     transition = dict((key, value / sum(transition.values())))
33     for key, value in transition.items():
            lexicon[word] = transition
```

The 'Console' tab shows the output of running the code:

```
> india is a great
india is a great country
>
```

Step 3:

The screenshot shows a Replit interface running on a Mac OS X desktop. The top bar displays the system menu, battery status, signal strength, and the date and time (Sat 30 Apr 11:34 PM). The browser tab is titled "replit.com".

The left sidebar contains project files: "main.py" (selected), "dataset.txt", "poetry.lock", and "pyproject.toml".

The main area shows the "main.py" code:

```
1 import numpy as np
2
3 lexicon = {}
4
5 def update_lexicon(current : str, next_word : str) -> None:
6     # Add the input word to the lexicon if it is there yet.
7     if current not in lexicon:
8         lexicon.update({current: {next_word: 1}})
9         return
10
11     # Receive the probabilities of the input word.
12     options = lexicon[current]
13
14     # Check if the output word is in the probability list.
15     if next_word not in options:
16         options.update({next_word : 1})
17     else:
18         options.update({next_word : options[next_word] + 1})
19
20     # Update the lexicon
21     lexicon[current] = options
22
23 # Populate lexicon
24 with open('dataset.txt', 'r') as dataset:
25     for line in dataset:
26         words = line.strip().split(' ')
27         for i in range(len(words) - 1):
28             update_lexicon(words[i], words[i+1])
29
30 # Adjust probability
31 for word, transition in lexicon.items():
32     transition = dict((key, value / sum(transition.values())))
33     for key, value in transition.items():
34         lexicon[word] = transition
```

The right panel shows the "Console" tab with the output of the program:

```
> india is a great country
india is a great country full
>
```

Step 4:

The screenshot shows a Replit interface on a Mac OS X desktop. The top bar displays the system menu, battery status, signal strength, and the date and time (Sat 30 Apr 11:35 PM). The browser tab is titled "replit.com".

The left sidebar contains project files: "main.py" (selected), "dataset.txt", "poetry.lock", and "pyproject.toml".

The main area shows the "main.py" code:

```
1 import numpy as np
2
3 lexicon = {}
4
5 def update_lexicon(current : str, next_word : str) -> None:
6     # Add the input word to the lexicon if it is there yet.
7     if current not in lexicon:
8         lexicon.update({current: {next_word: 1}})
9         return
10
11     # Receive the probabilities of the input word.
12     options = lexicon[current]
13
14     # Check if the output word is in the probability list.
15     if next_word not in options:
16         options.update({next_word : 1})
17     else:
18         options.update({next_word : options[next_word] + 1})
19
20     # Update the lexicon
21     lexicon[current] = options
22
23 # Populate lexicon
24 with open('dataset.txt', 'r') as dataset:
25     for line in dataset:
26         words = line.strip().split(' ')
27         for i in range(len(words) - 1):
28             update_lexicon(words[i], words[i+1])
29
30 # Adjust probability
31 for word, transition in lexicon.items():
32     transition = dict((key, value / sum(transition.values())))
33     for key, value in transition.items():
34         lexicon[word] = transition
```

The right panel shows the "Console" tab with the output of running the script:

```
> india is a great country full
india is a great country full of
:>
```

Step 5:

The screenshot shows a Replit interface running on a Mac OS X desktop. The title bar indicates it's a Safari window with the URL `replit.com`. The main area displays a Python script named `main.py` which reads from a file named `dataset.txt` and updates a lexicon. The script uses `numpy` and defines a function `update_lexicon` to handle word transitions. The console output shows the script processing the dataset and printing the updated lexicon.

```
import numpy as np
lexicon = {}

def update_lexicon(current : str, next_word : str) -> None:
    # Add the input word to the lexicon if it is there yet.
    if current not in lexicon:
        lexicon.update({current: {next_word: 1}})
        return

    # Receive the probabilities of the input word.
    options = lexicon[current]

    # Check if the output word is in the probability list.
    if next_word not in options:
        options.update({next_word : 1})
    else:
        options.update({next_word : options[next_word] + 1})

    # Update the lexicon
    lexicon[current] = options

# Populate lexicon
with open('dataset.txt', 'r') as dataset:
    for line in dataset:
        words = line.strip().split(' ')
        for i in range(len(words) - 1):
            update_lexicon(words[i], words[i+1])

# Adjust probability
for word, transition in lexicon.items():
    transition = dict((key, value / sum(transition.values())))
    for key, value in transition.items():
        lexicon[word] = transition
```

Console Output:

```
> india is a great country full of
india is a great country full of diversity
> |
```

Step 5:

The screenshot shows a Replit interface running on a Mac OS X desktop. The title bar indicates it's a Safari window with the URL `replit.com`. The main area displays a Python script named `main.py` which reads from a file named `dataset.txt` and updates a lexicon. The script uses `numpy` and defines a function `update_lexicon` to handle word transitions. The console output shows the script processing the dataset and printing the updated lexicon.

```
import numpy as np
lexicon = {}

def update_lexicon(current : str, next_word : str) -> None:
    # Add the input word to the lexicon if it is there yet.
    if current not in lexicon:
        lexicon.update({current: {next_word: 1}})
        return

    # Receive the probabilities of the input word.
    options = lexicon[current]

    # Check if the output word is in the probability list.
    if next_word not in options:
        options.update({next_word : 1})
    else:
        options.update({next_word : options[next_word] + 1})

    # Update the lexicon
    lexicon[current] = options

# Populate lexicon
with open('dataset.txt', 'r') as dataset:
    for line in dataset:
        words = line.strip().split(' ')
        for i in range(len(words) - 1):
            update_lexicon(words[i], words[i+1])

# Adjust probability
for word, transition in lexicon.items():
    transition = dict((key, value / sum(transition.values())))
    for key, value in transition.items():
        lexicon[word] = transition
```

Console Output:

```
> india is a great country full of
india is a great country full of diversity
> |
```



YAAAAAY!!!

The whole sentence is complete.
This is how a simple next word predictor works.

Thank you!!!

Our Teammates

- **Samriddho Das** (RA1911003010119)
- **Madura Sankar S.** (RA1911003010114)
- **Ashwin S.** (RA1911003010109)

