

Bag of Words

Name		Date	
Class / Sec		Roll no.	

1 - AIM

To write a Python program that implements the **Bag of Words** model — building a dictionary of unique words from a corpus, and creating a **document vector** of word counts for each document.

2 - SOFTWARE AND LIBRARIES USED

TOOL	WHAT IT IS FOR
Python 3	The programming language the program is written in
Jupyter Notebook	The environment used to write and run the code (IDLE or Colab also acceptable)
No extra library	Plain Python only — lists, loops and the count() method. Nothing to install

3 - THEORY

Bag of Words is a model that turns text into numbers, so a machine learning algorithm can work with it. It records **which words appear** and **how often** — and nothing else.

#	STEP	WHAT IT PRODUCES
1	Text Processing	Clean tokens — the normalisation from Program 11
2	Create a Dictionary	A list of every unique word in the corpus
3	Create document vectors	For each document, a count of every dictionary word
4	Repeat for all documents	The complete table of numbers

Why it is called a *bag*: Calling it a bag means the **sequence of words does not matter**. “Avni helped Aman” and “Aman helped Avni” produce **identical vectors** — same words, same counts, opposite meanings. Order is discarded by design.

4 - PROGRAM

```
# Three documents, already normalised in Program 11
documents = [
    "ai is fun",
    "ai is hard",
    "ai ai is fun fun"
]

# Step 2 - Build the dictionary of unique words
dictionary = []
for doc in documents:
    for word in doc.split():
        if word not in dictionary:
            dictionary.append(word)

print("Dictionary :", dictionary)
print("Unique words:", len(dictionary))

# Step 3 and 4 - Build a document vector for every document
```

```
print("\nDocument vectors")
print("      ", dictionary)

for i, doc in enumerate(documents, start=1):
    tokens = doc.split()
    vector = [tokens.count(word) for word in dictionary]
    print("D" + str(i), "      ", vector, "    sum =", sum(vector))
```

Built without a library on purpose: Libraries such as **scikit-learn** can do this in two lines. Writing the loops yourself shows you understand what the model is actually doing — which is what the viva will ask about.

5 • EXPECTED OUTPUT

Dictionary : ['ai', 'is', 'fun', 'hard']
Unique words: 4

Document vectors
 ['ai', 'is', 'fun', 'hard']
 D1 [1, 1, 1, 0] sum = 3
 D2 [1, 1, 0, 1] sum = 3
 D3 [2, 1, 2, 0] sum = 5

Written out as a table, this is what the corpus has become:

	ai	is	fun	hard
D1	1	1	1	0
D2	1	1	0	1
D3	2	1	2	0

[PASTE SCREENSHOT HERE]

Paste a screenshot of YOUR program running, with the code and output visible together

6 • CHECKING THE TABLE

Every number can be verified by hand, and there is a check that catches almost any counting error.

CHECK	WORKING
Dictionary size	4 unique words — ai, is, fun, hard. Repeats are listed only once
Vector length	Always 4 — one entry per dictionary word, whatever the document length
Row sums	D1 = 3, D2 = 3, D3 = 5 — matching the token count of each document
The zeros	hard appears only in D2, so D1 and D3 score 0 for it. A zero is a real answer
D3 has 5 tokens	But still 4 entries — the extra length shows as larger numbers, not more columns

The row-sum check: Every vector should add up to the number of tokens in that document. It takes five seconds and it catches nearly every counting mistake — use it in the exam too.

7 • OBSERVATION AND RESULT

Two or three sentences in your own words. Say what the corpus became, and what the model kept and discarded.

Example observation: The three documents produced a dictionary of 4 unique words, so every document vector has 4 entries regardless of its length. D3 contains 5 tokens but still has only 4 numbers — the extra length appears as larger counts rather than extra columns. Each row adds up to the token count of its document, which confirms the counting is correct.

Worth knowing: If you use **scikit-learn** instead, its dictionary is sorted **alphabetically** — ai, fun, hard, is — so the columns appear in a different order. The counts are the same; only the arrangement differs. Do not panic if your table looks different from this one.

8 · VIVA QUESTIONS FOR THIS PROGRAM

LIKELY QUESTION	SHAPE OF A STRONG ANSWER
Why is it called a bag of words?	Because the order does not matter — only which words appear and how often
D3 has 5 tokens. Why does its vector have only 4 numbers?	Vector length comes from the dictionary , not the document. Four unique words, four entries
What does a 0 in the table mean?	That word exists somewhere in the corpus but not in this document. It is a real measurement
Why does ai appear once in the dictionary but twice in D3?	The dictionary lists unique words. Repetition is recorded in the counts , not the word list
What has the model thrown away?	Word order and grammar. Two sentences with opposite meanings can give identical vectors

9 · HOW THIS ENTRY IS MARKED

FULL MARKS

Dictionary and vectors **both printed** · the table drawn out with **every zero shown** · row sums checked against token counts · a screenshot of **your own** output · you can explain why vector length comes from the dictionary

LOSES MARKS

Zeros left blank instead of written as 0 · vectors of different lengths for different documents · no screenshot, or someone else's · an observation that just repeats the numbers · cannot say why the model is called a **bag**

Before you submit: Write **your own** three documents. Make sure at least one word appears **twice in one document**, and one word appears in **only one** document — otherwise your table has no counts above 1 and no zeros, and there is nothing to explain in the viva.