# Text Vectorization

By Kanav Bansal
(Thatai guy.com)

# Data Preparation for Text :

Data Preparation

Historical Raw Data → [ Clean → Transform ] → Processed Data → Learning [ Algorithm ] → Model

**Text Preprocessing** → Clean

( To clean the noise from text data)

**Text Vectorization** → Transform

( To convert Text data to numerical representation )

## Terminologies :

ⓐ Document → It is a single piece of text. It can be a sentence, para, article, review, email body etc..

ⓑ Corpus → Corpus is a collection of documents

ⓒ Vocabulary → Set of all unique words that appear in a corpus.

ⓓ Vectorization → It is a technique to convert text data into numerical vectors.

ⓔ Document Vectors → Numerical Representation of a document

ⓕ Document Term Matrix → A DTM is a matrix where rows represent document vectors & columns represent terms from the vocabulary.

## Text Vectorization Techniques (AKA Feature Extraction)

ⓐ Bag of Words

ⓑ Term Frequency Inverse Document Frequency

ⓒ Word2Vec

ⓓ GloVe

ⓔ FastText

ⓕ ELMo

ⓖ GPT

ⓗ BERT

ⓘ LLM's

## Text Cleaning (AKA Text Preprocessing)

ⓐ Removing Special Characters

ⓑ Converting to lower case

ⓒ Removing Stop words

ⓓ Converting to root form

# Text Vectorization → Bag of Word

Converts each document into a vector, where each element of vector is the count of a term in the document.

### Corpus

Doc-1: we are learning machine learning

Doc-2: processing natural language data

Doc-3: machine learning algorithms

**Step 1: Build the vocabulary (fit)**

i.e. Learn the unique words in the corpus

Vocabulary = { 'algorithms', 'are', 'data', 'language', 'learning', 'machine', 'natural', 'processing', 'we' }

Represented in sorted order

size of vocabulary = 9

**Step 2: Apply transformation (aka vectorization)**

For each document, count the number of times each vocabulary term appears in that document.

Doc Vector

| | 'algorithms' | 'are' | 'data' | 'language' | 'learning' | 'machine' | 'natural' | 'processing' | 'we' |
|---|---|---|---|---|---|---|---|---|---|
| DV-1: | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 1 |
| DV-2: | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| DV-3: | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

$n*d$

Shape of DTM = $n * d$

# of documents ← → Size of vocabulary

## Problems / Disadvantages

1. **Dimensionality & Sparsity of DTM**

Ques: What is a sparse Matrix?

Ans: A matrix where most entries are zeroes.

# of entries in matrix (dtm) = $n*d = 3*9 = 21$

Average no. of words per document = 4 words / doc

# of non-zero entries in dtm = $4 * n = 4*3 = 12$

i.e Almost 50% of dtm is filled with zeroes

Ques: Why is it a problem?

Ans: Visualization, Memory Consumption, Computational Complexity, Curse of dimensionality and Interpretability.

Ques: Solution?

Ans: Remove Stop Words & Convert to root form

↑ Problem: Even after this dimensionality remains very large.

2. **Sequence Info is lost**

Ques: Solution?

Ans: n-grams approach

↑ Problem: Dim & Sparsity problem with this approach.

3. **Out of vocabulary words can't be handled.**

4. **Semantics (or meaning) of words is lost.** Word Similarity is not captured.

# Text Vectorization → TF IDF

Converts each document into a vector, where each element of vector is the TFIDF score of a term in the document.

### Corpus

**Doc-1:** we are learning machine learning
**Doc-2:** processing natural language data
**Doc-3:** machine learning algorithms

### Step 1: Build the vocabulary (fit)

Vocabulary = { 'algorithms', 'are', 'data', 'language', 'learning', 'machine', 'natural', 'processing', 'we' }

### Step 2: Apply transformation (aka vectorization)

For each document, calculate the TFIDF score for each vocabulary term which appears in that document.

$$\text{TF IDF Score} = \text{TF} * \text{IDF}$$

$$TF(w_i, doc_J) = \frac{\text{No. of times } w_i \text{ occurs in } doc_J}{\text{Total no. of words in } doc_J}$$

$$IDF(w_i, corpus) = \log\left(\frac{\text{No. of docs in corpus}}{\text{No. of docs containing } w_i}\right) + 1$$

**Note:**

* If a word is more frequent in a document, its TF will be more.
* If a word is rare in a document, its IDF will be more.

Let's see the TFIDF score for 'learning' in Doc-1:

$$TF(\text{'learning'}, \text{Doc-1}) = \frac{2}{5} \qquad IDF(\text{'learning'}, corpus) = \log\left(\frac{3}{2}\right) + 1$$

$$\begin{array}{c|ccccccccc} & \text{'algorithms'} & \text{'are'} & \text{'data'} & \text{'language'} & \text{'learning'} & \text{'machine'} & \text{'natural'} & \text{'processing'} & \text{'we'} \\ \hline DV1 & [\,0 & - & 0 & 0 & \frac{2}{5} * \{\log(\frac{3}{2}) + 1\} & - & 0 & 0 & -\,] \\ DV2 & [\,0 & 0 & - & - & 0 & 0 & - & - & 0\,] \\ DV3 & [\,- & 0 & 0 & 0 & - & - & 0 & 0 & 0\,] \end{array}$$

Let's now compute the TF IDF score for a word which was never present in the corpus.

**Doc-4:** cleaning natural language

$$TF(\text{'cleaning'}, \text{Doc-4}) = \frac{1}{3} \qquad IDF(\text{'cleaning'}, corpus) = \log\left(\frac{3}{0}\right) + 1$$

Zero Divison Error

### IDF Formula with Smoothing

$$IDF(w_i, corpus) = \log\left(\frac{N+1}{DF(w_i)+1}\right) + 1$$

### Significance:

Incorporating smoothing into the IDF calculation ensures that new or unknown terms, which were not present in the original corpus, are handled appropriately.

Avoids zero division error for terms/words which are not present in any document.

# What's Next?

## A Big Question :

Let's assume there are following vocabulary words in the corpus:

{'good', 'great', 'bad', 'wonderful', 'poor'}

Numerical Representation for each word :

$$
\begin{array}{c} \\ \text{good} \\ \text{great} \\ \text{bad} \\ \text{wonderful} \\ \text{poor} \end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{'bad'} & \text{'good'} & \text{'great'} & \text{'poor'} & \text{'wonderful'} \end{array} \\
\begin{bmatrix} [\,0 & 1 & 0 & 0 & 0\,] \\ [\,0 & 0 & 1 & 0 & 0\,] \\ [\,1 & 0 & 0 & 0 & 0\,] \\ [\,0 & 0 & 0 & 0 & 1\,] \\ [\,0 & 0 & 0 & 1 & 0\,] \end{bmatrix}
\end{array}
$$

Que: Is it possible to visualize a scatter plot for above 5 dimensional vectors?

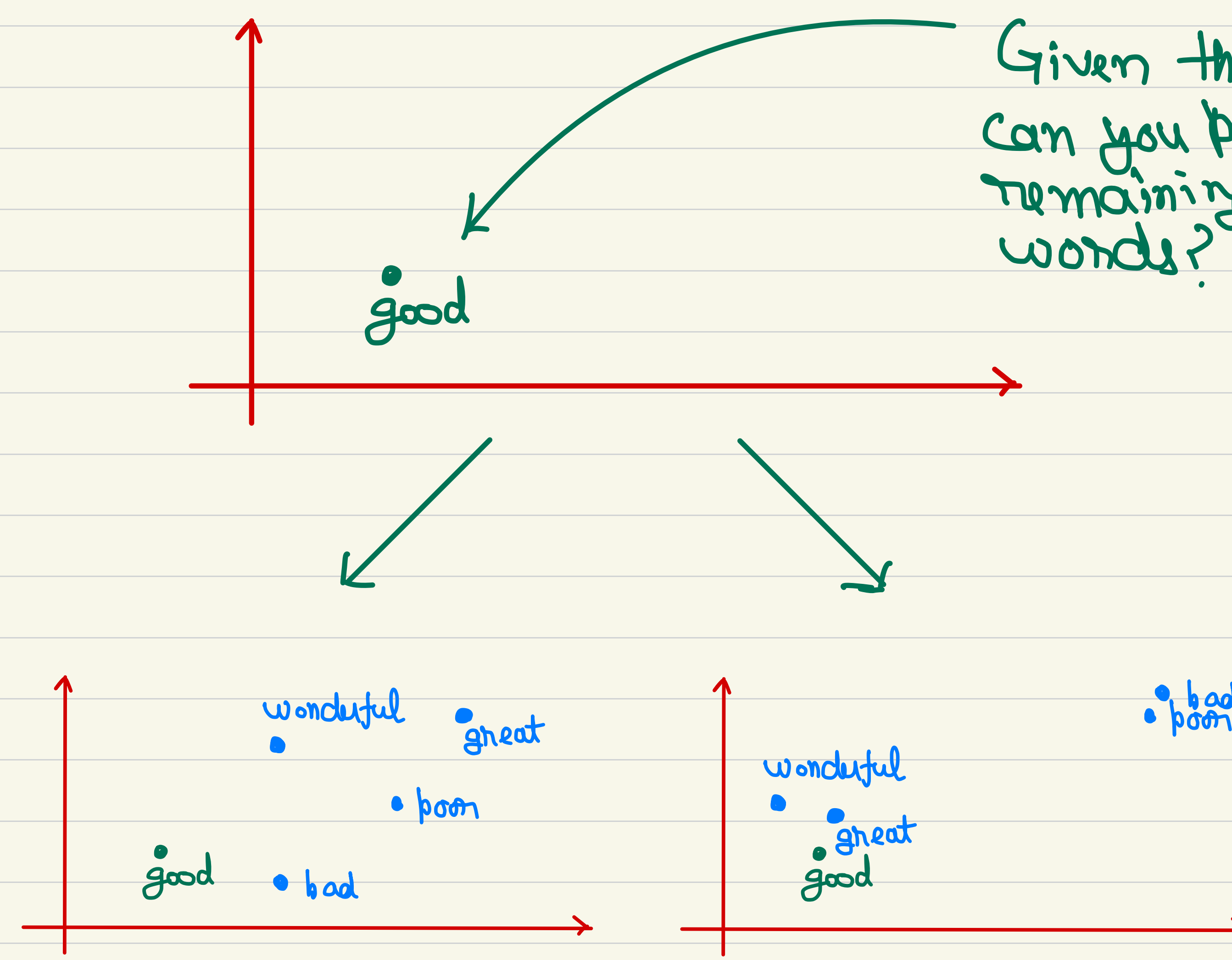
Ans: Not directly. We can reduce the dimensionality of the data & bring it to 2D with the help of Algorithms like PCA.

d-dim vectors → $\boxed{\text{PCA}}$ → 2 dim vectors

Que: Assuming that all the vocabulary words can be represented using 2 dimensional vector representation, which plot do you think make more sense?

## Scatter plot :

Given this can you plot remaining words?




Plot - 1 (BOW & TFIDF)

Plot - 2 (W2V, GloVe, FastText, BERT, GPT, LLM's...)

## Vectorization Techniques

| Problems | BOW & TFIDF | W2V & GloVe | GPT, BERT |
|---|---|---|---|
| 1. Dimensionality & Sparsity | High | Low (AKA Embeddings) | Low (AKA Embeddings) |
| 2. Semantic | ✗ | ✓ | ✓ |
| 3. Sequence info | ✗ | ✗ | ✓ |
| 4. OOV | ✗ | ✗ | ✓ |