**NLP 2**

****

### **Introduction to the Series:**

Welcome to the exciting series on Natural Language Processing (NLP)! In this journey, we will unravel the secrets behind the techniques and models that power language-based applications. Here’s what you can look forward to:

1. **Text Pre-processing**
2. **Text Representation (This Article)**
3. **Word Embedding**
4. **Language Modeling**
5. **Topic Modeling**
6. **Recurrent Neural Network (RNNs) for NLP**
7. **LSTM for NLP**
8. **Transformer Models**
9. **BERT**

#### **Prerequisites:**

1. **Knowledge of Python**
2. **Text Pre-processing (Part 1)**

### **In This Article:**

Today, we’ll dive into the text representation techniques used in NLP restricted to the following topics:

* **Why do we need to represent text as numbers?**
* **How do we represent text as numbers?**
  + **Sparse representation of text**
  + **Dense representation of text**
* **One Hot Encoding**
* **Bag of Words**
* **TF IDF**
* **Cosine Similarity**
* **Case Study: Medium Article Recommendation System**

**What is Text Representation?**

**The process of converting a text into numerical vectors that the machine can understand is called as text representation in NLP.**

**Why do we need to represent text as numbers?**

**Computers know only number (binary), whatever the activity that need to be performed using it should be done through numbers. The computer understands the text using ascii codes.**

**Not like human, the machine learning model learns to do an activity based on the numbers.**

**Hence it is required to represent the text as numbers.**

**How do we represent text as numbers?**

**Broadly the texts are represented as a number in two categories.**

**Sparse Representation Techniques:**

**It refers to a way of representing data where most of the elements in the data structure are zero.**

**Dense Representation Techniques:**

**There are variety of ways that we can represent the text as a number or vector based on the requirements. The representation should be capable of represent the semantic and syntactic meaning of the word or text.**

**In the previous article, we have seen a simple word frequency dictionary based text representation technique. Here we will some advanced methods.**

**One hot encoding:**

One-hot encoding is a straightforward text representation technique where each word or token is represented by a binary vector with a single "1" indicating the presence of the word and all other positions set to "0"

**Example:**

Consider the following sentences

Sentence 1: “**it was the best of times**”

Sentence 1: “**it was the worst of times**”

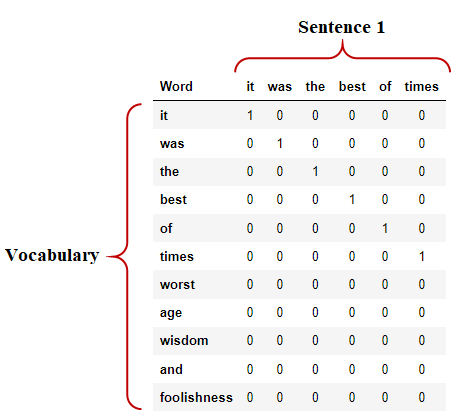
Sentence 3: “**it was the age of wisdom and the age of foolishness**”

1. **Build the vocabulary:**

* Create a list of all unique words across the entire corpus (Here 3 sentences)
* Vocabulary: [it, was, the, best, of, times, worst, age, wisdom, and, foolishness]

1. **Create one hot encoded representation:**

Each word in the vocabulary is represented by a binary vector with a size equal to the number of unique words.



**For the sentence 1,**

"it" → [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

"was" → [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

"the" → [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

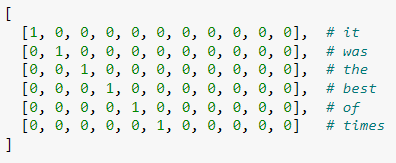
"best" → [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

"of" → [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

"times" → [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

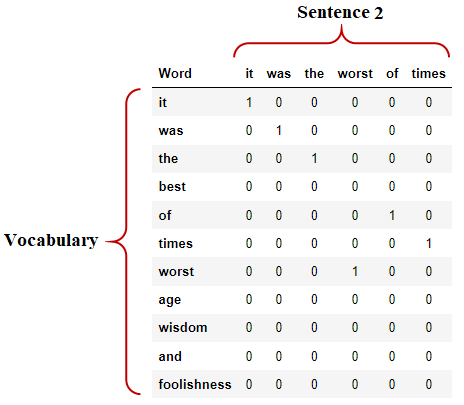
**Hence each word is represented with the size of 1 x N vector, where N is the size of the vocabulary.**

**On combining these vectors, the sentence is represented as:**



**Hence the sentence is transformed into numerical vector of size M x N, where M is the number of words in the sentence.**

**Similarly for the sentence 2, one hot encoded representation will be as,**

****

**Drawbacks:**

* **High dimensional sparse matrix representation**
* **All the words in the sentence were given equal importance**
* **The vector representation of each word is orthogonal and hence the relationship between different words cannot be measured**

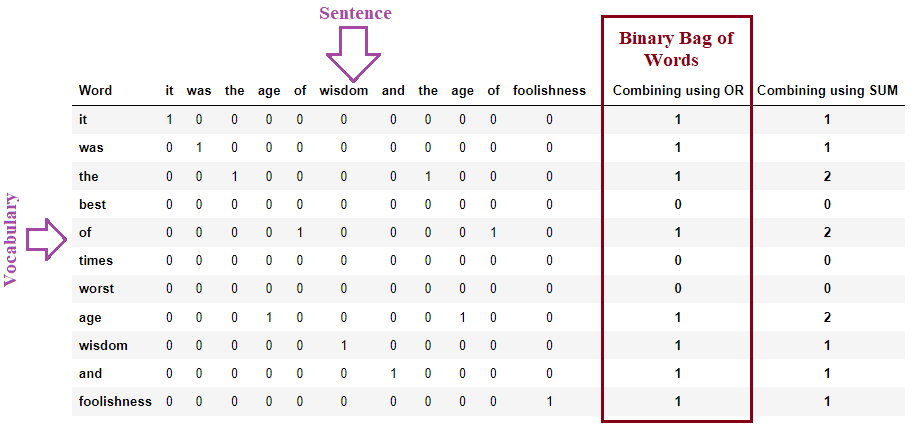
**Bag of Words:**

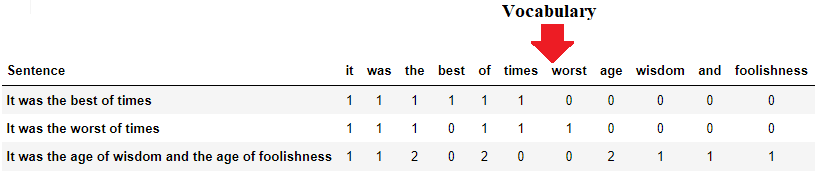
**Some of the drawbacks are addressed in the bag of words technique.**

**Follow the same steps as that of one hot encoding technique, instead of representing the word with 1 or 0, represent it with count of the word in the sentence i.e., frequency of occurrence.**

**Once we have the OHE of all words in a document, the only additional step is to combine them, to get a get a single vector representation for the document. There are two common ways to do:**

* **Using a binary OR operator between the OHE vectors. The final vector that we get in this case simply tells the absence or presence of certain words in the document. It is the binary bag of words.**
* **Using a vector sum operator. The final vector that we get in this case tells the frequency of each word in the document.**

****

****

**The dimension of the sentence is represented with the size of 1 x N vector, where N is the size of the vocabulary. Hence the size is irrespective of number of words in the sentence.**

**Drawbacks:**

* **Though the dimensionality is reduced compared to sparse matrix representation, the words in the sentence were given equal importance.**
* **It cannot distinguish between the rare important words and the common words.**

**For example:**

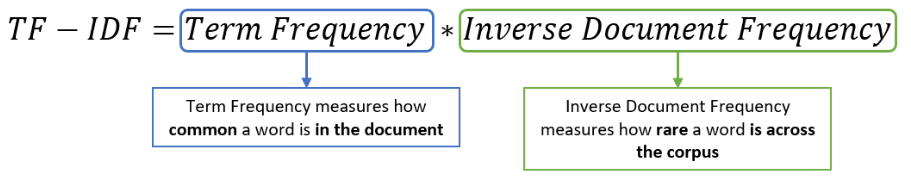
If you're looking at a set of articles about deep learning, then the phrase "neural network" might be present in a lot of articles and hence does not convey a lot of information (considering the corpus), whereas the bag of words doesn’t capture but returns the results

**TF IDF:**

TF-IDF is the method that gives higher weightage to rare important words and lesser weightage for frequent words.

TF-IDF is mainly composed of two components –

* TF: Term Frequency
* IDF: Inverse Document Frequency

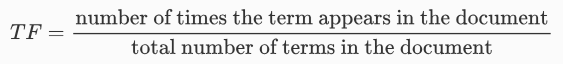


**In our example, document refers to the sentence and the corpus refers to the collection of all the sentences.**

**Both Term Frequency and Inverse Document Frequency is calculated for every word in the vocabulary.**

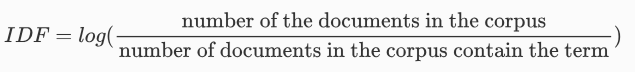
**Term Frequency:**

* **More common words (or tokens) would have a higher term frequency**
* **This is calculated for every word in a document**



**Inverse Document Frequency:**

* **A rarer word (or token) would have a larger IDF.**
* **It is calculated for every word in the vocabulary**



**Example:**

Consider the following sentences

Sentence 1: “**it was the best of times**”

Sentence 1: “**it was the worst of times**”

Sentence 3: “**it was the age of wisdom and the age of foolishness**”

**First let’s calculate the IDF as it corresponds to the words in vocabulary.**

**In our example we have 3 sentences, hence the No. of documents in the corpus is 3.**

|  |  |  |
| --- | --- | --- |
| **Words** | **No. of documents in the corpus the word is occurring (x)** | **IDF** |
| it | 3 | 0 |
| was | 3 | 0 |
| the | 3 | 0 |
| best | 1 | 0.4771 |
| of | 3 | 0 |
| times | 2 | 0.1761 |
| worst | 1 | 0.4771 |
| age | 1 | 0.4771 |
| wisdom | 1 | 0.4771 |
| and | 1 | 0.4771 |
| foolishness | 1 | 0.4771 |

**The IDF score for the words that are present in all the sentences are 0. Unique words in a sentence is given higher weightage.**

**Now let’s calculate term frequency for each word in the sentence.**

**Sentence 1:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Word** | **it** | **was** | **the** | **best** | **of** | **times** |
| **No of times the word is occurring** | **1** | **1** | **1** | **1** | **1** | **1** |
| **TF** | **1/6** | **1/6** | **1/6** | **1/6** | **1/6** | **1/6** |
|  |  |  |  |  |  |  |

**Case Study:**

**Understanding the Dataset**

**Dataset source:**[Link](https://drive.google.com/file/d/1MyOEKk_z78P8JL0mTYSerRiPLVflkVK6/view?usp=sharing)

**Total 208 articles with title, article text, author name, reading time etc.**

