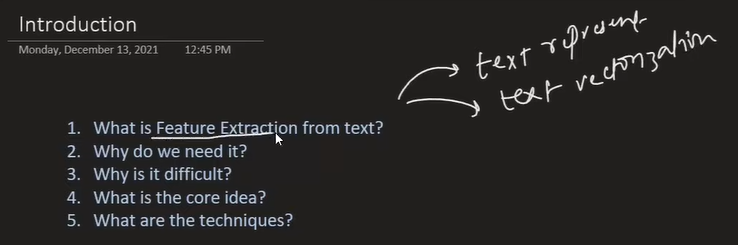
# **Text Representation | NLP Lecture 4 | Bag of Words | Tf-Idf | N-grams, Bi-grams and Uni-grams**

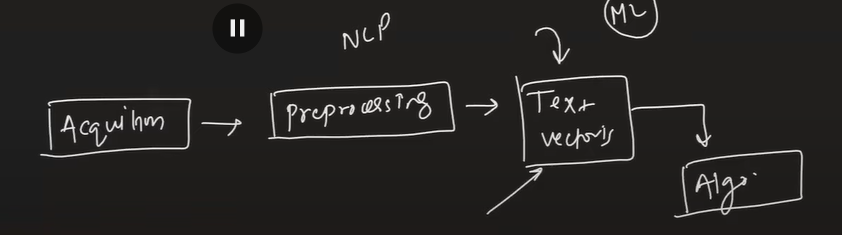


## **I. Introduction to Text Representation**

The goal of text representation (also known as **Text Vectorisation** or **Feature Extraction from Text**) is to c**onvert text data into numbers**. This conversion is essential because Machine Learning **(ML) algorithms cannot understand language** directly; they only process numbers and mathematics.

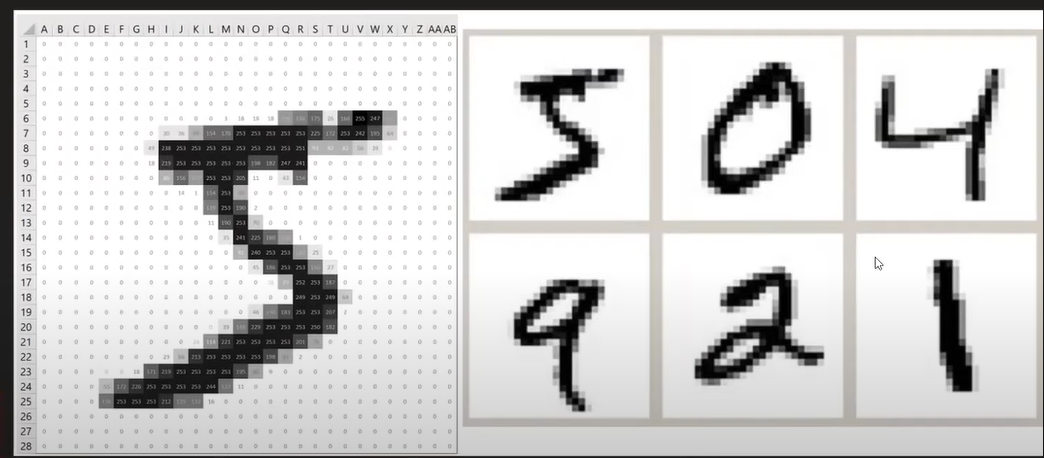
### **A. Importance and Context**

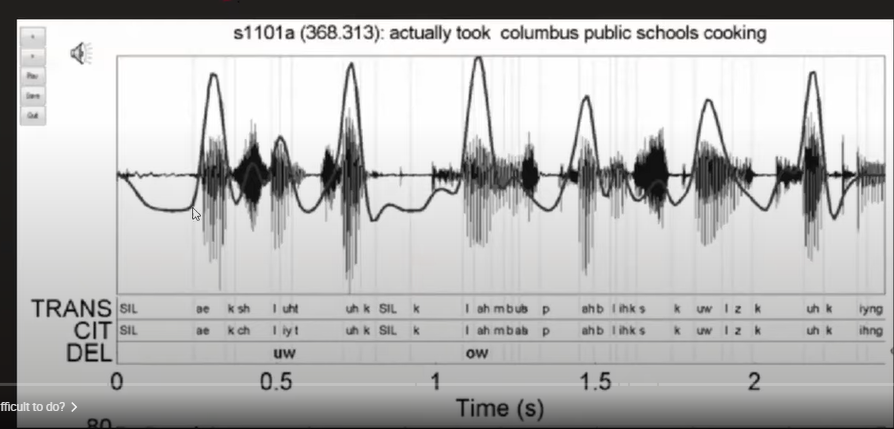
1. **Feature Quality:** The performance of an ML model heavily depends on the quality of the features provided. Providing good features to even a simple algorithm yields better performance than providing poor features to a sophisticated algorithm **("Garbage in, garbage out")**.
2. **NLP Pipeline:** Text representation is a crucial step in the NLP pipeline, occurring after data acquisition and text pre-processing. This stage is often considered the **most important** step in machine learning-based NLP.
3. **Core Objective:** The numbers derived from the text must convey the **semantic meaning** (the hidden meaning) of the text to the model.

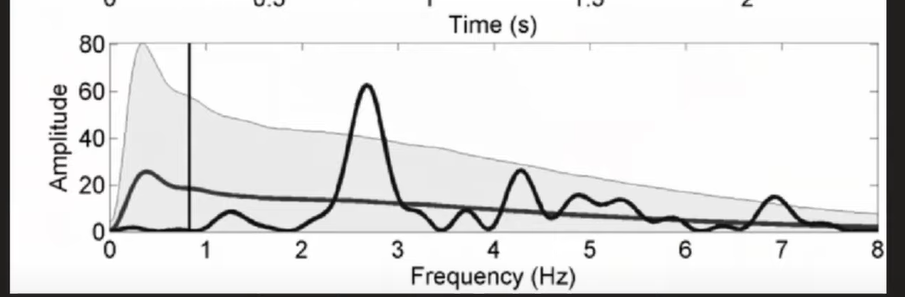


### **B. Challenges in Text Vectorisation**

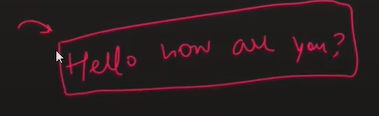
Converting text to numbers is highly difficult. Unlike image data (where images are already pixels, which are numbers), or speech data (where audio amplitude can be represented as a series of numbers), text data requires complex techniques to map meaningful concepts to vectors.







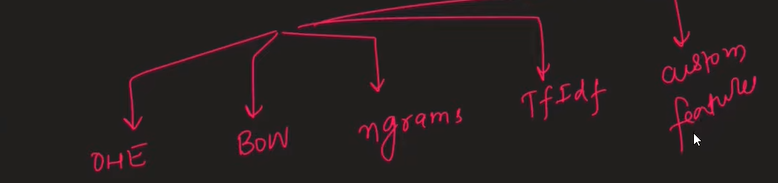
What is in text data



### **C. Techniques Covered**

This covers several techniques for converting text to vectors (numbers):

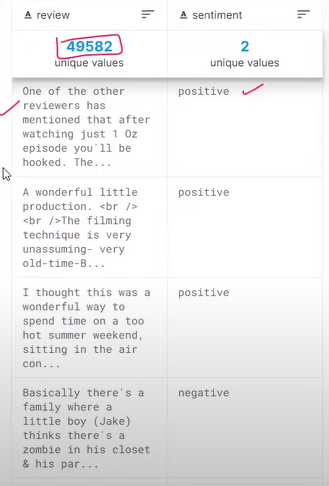
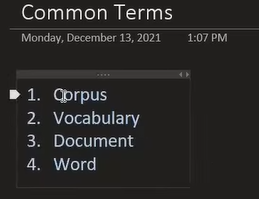
* One-Hot Encoding
* Bag of Words (BoW)
* N-grams (Uni-grams, Bi-grams, Tri-grams)
* TF-IDF (Term Frequency–Inverse Document Frequency)
* Custom Features
* Future topics mentioned include **Word2Vec** or **Embeddings**, which are Deep Learning-based approaches.



## **II. Key Terminology**

Four terms are fundamental to understanding text representation techniques:

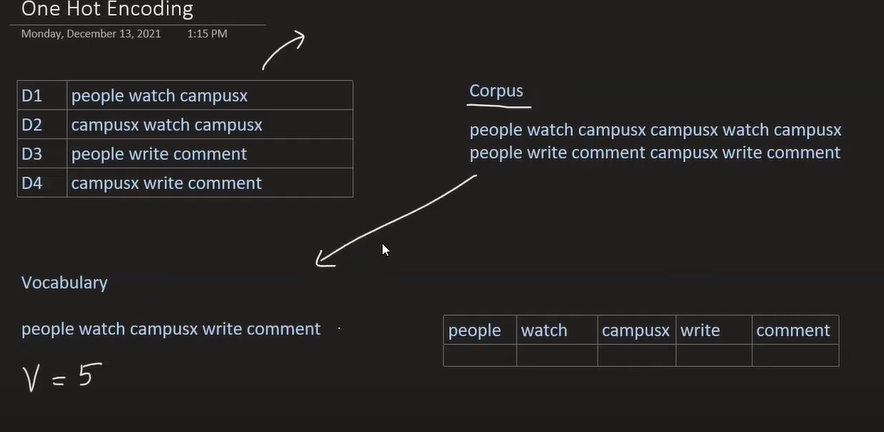
| **Term** | **Definition** |
| --- | --- |
| **Corpus (C)** | The combination of *all* words from all documents in the dataset, including repeated words. |
| **Vocabulary (V)** | The set of *unique* words present in the corpus. The size of the vocabulary determines the dimensionality of the resulting vectors. |
| **Document (D)** | An individual text unit, such as a single review or sentence. |
| **Word (W) or Term (T)** | An individual word within a document. |

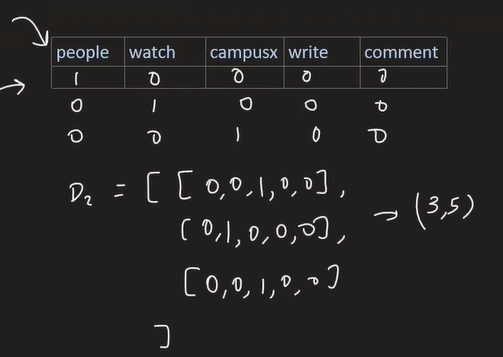


## **III. One-Hot Encoding (OHE)**

### **A. Mechanism**

1. **Vector Creation:** OHE converts every word in a document into a vector whose size is equal to the size of the Vocabulary ($V$).
2. **Representation:** If $V=5$, a word is represented by a $5$-dimensional vector. Only the position corresponding to that specific word is set to $1$, while all other positions are $0$.
3. **Document Representation:** A document is represented by concatenating the vectors of its individual words.

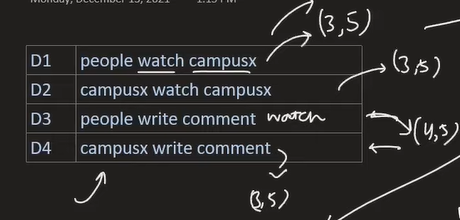




### **B. Disadvantages (Why it is not used)**

OHE is primarily taught as a conceptual stepping stone because it has major drawbacks:

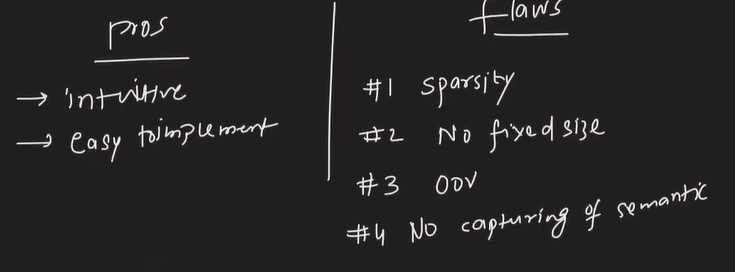
1. **Sparsity:** In a real-world dataset with a large vocabulary (e.g., 50,000 unique words), each word vector will be 50,000 dimensions long, with only one '1' and thousands of '0's. This **sparse array** is computationally difficult to handle and can lead to overfitting.
2. **Non-Fixed Size:** ML algorithms require a fixed input size. Since the vector resulting from OHE depends on the number of words in the document, documents of different lengths (e.g., $3\times5$ vs. $4\times5$ shape) cannot be fed into the model.

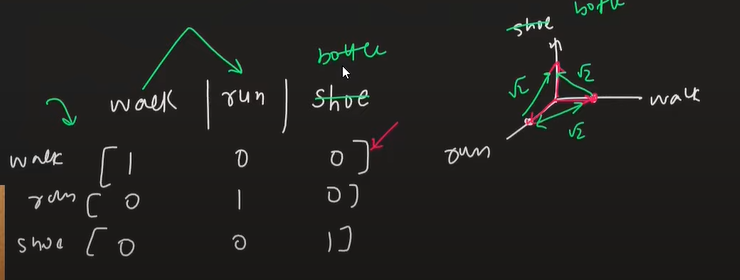


1. **Out of Vocabulary (OOV) Problem:** If a word appears during the prediction phase that was not present in the training vocabulary, the system cannot convert it into a number vector, halting the process.



1. **No Semantic Meaning:** In the vector space created by OHE, all word vectors are **equidistant** from each other. This fails to capture the subtle semantic differences (e.g., *walk* and *run* are semantically closer than *walk* and *bottle*, but OHE treats them equally distant).

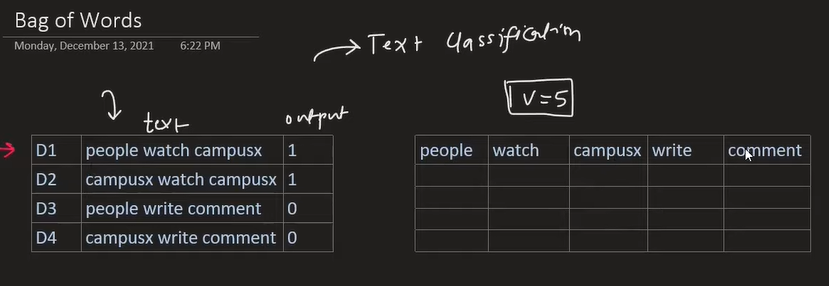




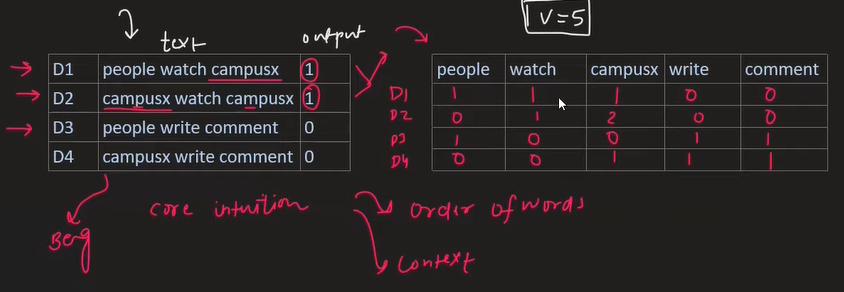
## **IV. Bag of Words (BoW)**

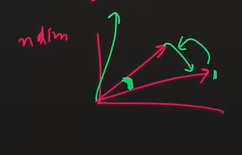
BoW is one of the most widely used techniques, especially for Text Classification problems.

### **A. Mechanism and Core Idea**



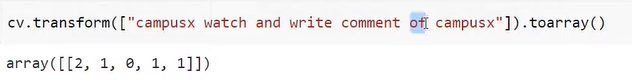
1. **Vector Creation:** After determining the vocabulary, each document is converted into a $V$-dimensional vector.
2. **Frequency Count:** Each dimension of the document vector records how many times the corresponding vocabulary word occurred in that specific document.
3. **Intuition:** The core intuition is that similar documents (e.g., belonging to the same category) will use the same type of words with similar frequency.
4. **Order Ignored:** BoW treats the entire text as a "bag" of words. The **order of words does not matter**, and contextual meaning is not directly covered.
5. **Vector Space:** Sentences are represented as vectors in an $N$-dimensional space. The similarity between two sentences/documents is often calculated based on the angle (cosine similarity) between their respective vectors.







1. **Implementation:** Scikit-learn implements this technique using the CountVectorizer class.



### **B. Hyperparameters (Customisation)**



The CountVectorizer allows for several powerful customisations:

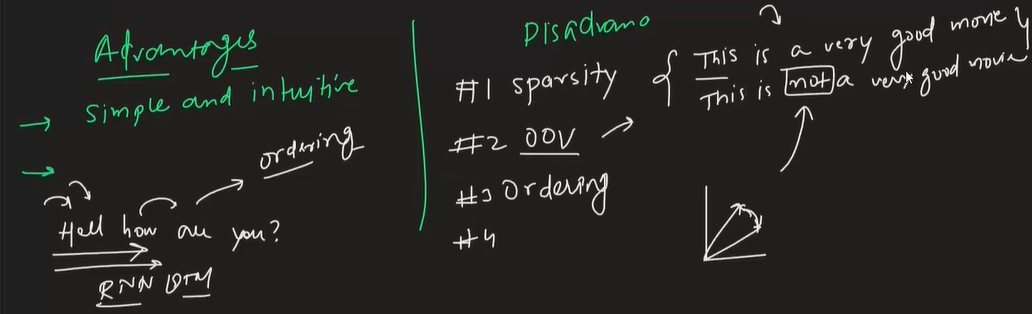
* **binary=True:** Instead of counting raw frequency, this sets the value to 1 if the word exists and 0 if it does not. This is often more effective than raw counts for tasks like sentiment analysis.
* **max\_features:** This limits the vocabulary to the top $N$ words based on frequency. This is useful for pruning rare words and managing dimensionality.
* Other parameters allow defining stop words or custom tokenisation patterns.

### **C. Advantages over OHE**

1. **Fixed Size:** The output vector size is always $V$ (the size of the vocabulary), regardless of the length of the document. This solves OHE's non-fixed size problem, allowing ML models to train.
2. **OOV Handling:** If a new word (OOV) appears during prediction, it is simply ignored because it is not in the training vocabulary, allowing the model to continue functioning (though information is lost).

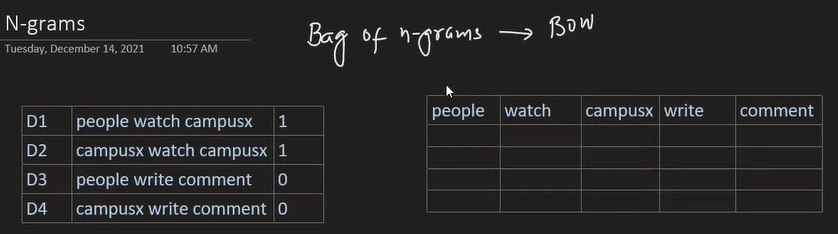
### **D. Disadvantages**

1. **Sparsity:** BoW still suffers from high sparsity, especially with large vocabularies.
2. **Order Neglected:** Ignoring word order prevents the capture of sequence-dependent meaning.
3. **Context and Negation Failure:** BoW cannot capture negation effectively. For example, "This is a very **good** movie" and "This is **not** a very **good** movie" will produce vectors that are very close together (as most words are shared), despite having completely opposite semantic meanings.



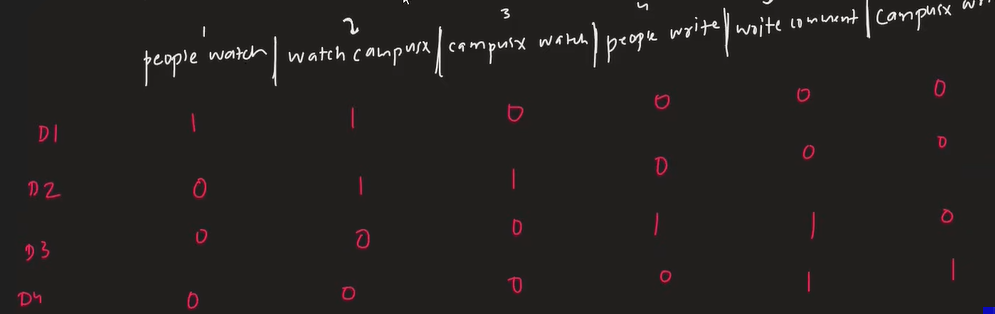
## **V. N-grams (Bag of N-grams)**

N-grams are an extension of the Bag of Words concept designed specifically to address the problem of neglecting word order and context.

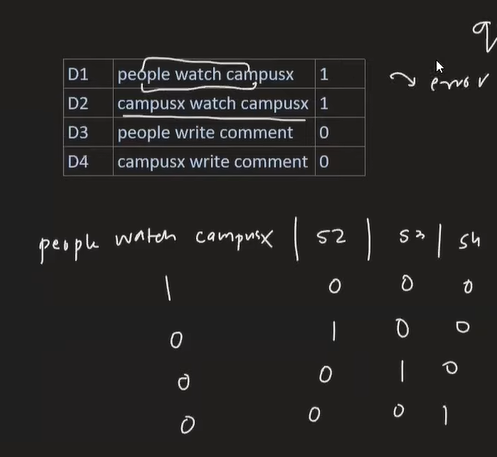


### **A. Mechanism**

1. **Vocabulary Definition:** Instead of building the vocabulary from single words (known as **Uni-grams**), N-grams use continuous sequences of $N$ words.
   * **Bi-grams (N=2):** The vocabulary consists of pairs of consecutive words (e.g., "People watch", "watch CampusX").



* + **Tri-grams (N=3):** The vocabulary consists of triplets of consecutive words (e.g., "People watch CampusX").

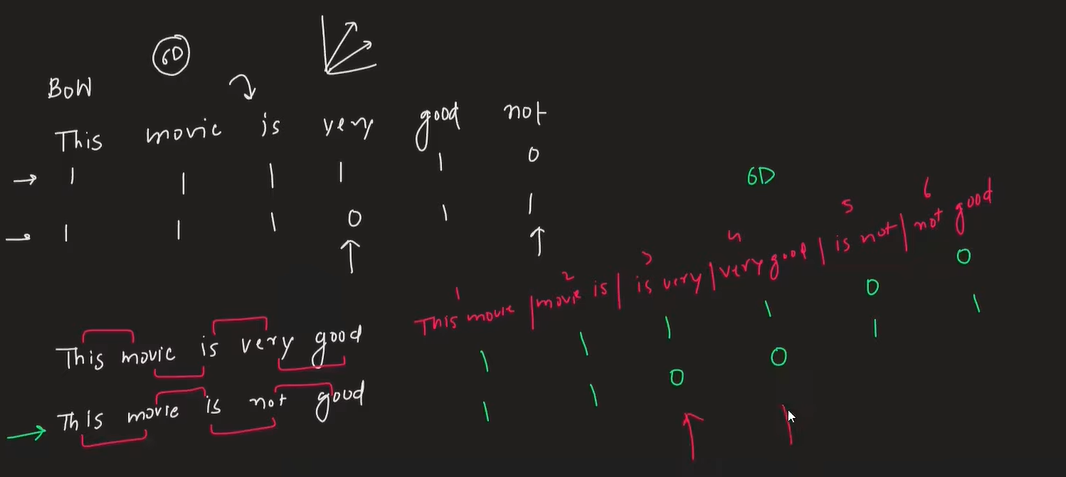


1. **Count:** Once the N-gram vocabulary is established, document vectors are generated by counting the frequency of these N-gram phrases.
2. **Implementation:** In Scikit-learn, this is controlled using the ngram\_range parameter in CountVectorizer (e.g., (1, 1) for uni-grams, (2, 2) for bi-grams, or (1, 3) to include uni, bi, and tri-grams). The standard BoW approach is a special case of N-grams where the range is restricted to uni-grams.



### **B. Benefits and Disadvantages**

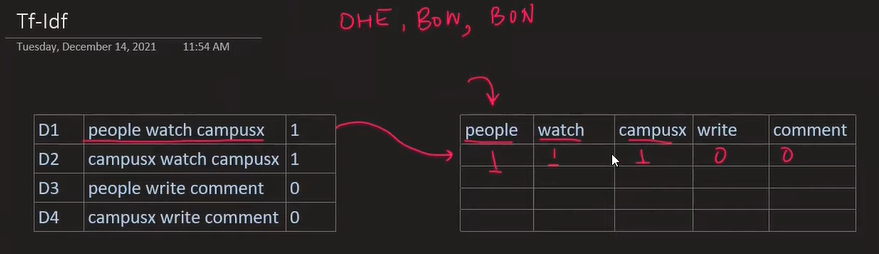
1. **Benefit:** By capturing contiguous word sequences (phrases), N-grams are much better at capturing the **semantic meaning** of the sentence, especially handling negation (like the "is not good" example).



1. **Disadvantage (Increased Complexity):** As $N$ increases (moving from uni-grams to bi-grams, and bi-grams to tri-grams), the dimensionality of the vocabulary grows significantly. This increases the time complexity of the algorithm, making the model slower to train and predict.
2. **Disadvantage (OOV):** N-grams still have no solution for handling new phrases that appear as out-of-vocabulary terms during prediction.

## **VI. TF-IDF (Term Frequency–Inverse Document Frequency)**

TF-IDF is a weighting scheme that assigns a numerical importance score to each word in a document, unlike BoW/N-grams, which treat all counts equally

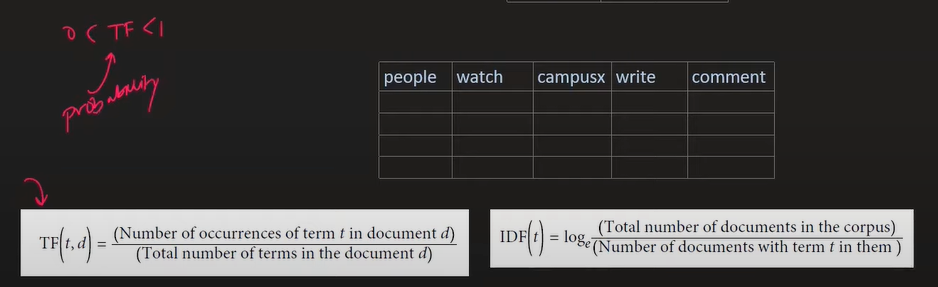
.

### **A. Core Logic**

The importance of a word is determined by two factors:

1. How frequently the word appears in the current document (Term Frequency).
2. How rarely the word appears across the entire corpus (Inverse Document Frequency).

TF-IDF is calculated by multiplying these two components: $\text{Weight} = \text{TF} \times \text{IDF}$. Words that are document-specific (high TF in one document, low IDF overall) receive a higher weight.



### **B. Term Frequency (TF)**

TF measures the probability or frequency of a term $T$ occurring within a document $D$. $$ \text{TF}(T, D) = \frac{\text{Number of occurrences of } T \text{ in } D}{\text{Total number of terms in } D} $$ TF values always range between 0 and 1.

### **C. Inverse Document Frequency (IDF)**

IDF measures the rarity of a term $T$ across the entire corpus. $$ \text{IDF}(T) = \log \left(\frac{\text{Total number of documents in the corpus (N)}}{\text{Number of documents containing term } T}\right) $$

* If a word appears in almost all documents (e.g., common stop words like "the"), its IDF score is low (approaching zero).
* If a word is rare, its IDF score is high.

**Why use Logarithm?** Logarithms are applied to the IDF calculation to smooth the value and normalise the results. Without the logarithm, extremely rare words would receive disproportionately huge IDF scores (e.g., 1000/1 = 1000 for a rare word in a 1000-document corpus). This high IDF value would dominate the TF value when multiplied, skewing the overall word importance.

### **D. Implementation and Use**

1. **Implementation:** Scikit-learn uses the TfidfVectorizer class. Note that standard implementations often add $+1$ to the denominator or numerator in the IDF calculation (e.g., adding $+1$ to the denominator prevents terms appearing in all documents from yielding an IDF of exactly zero).



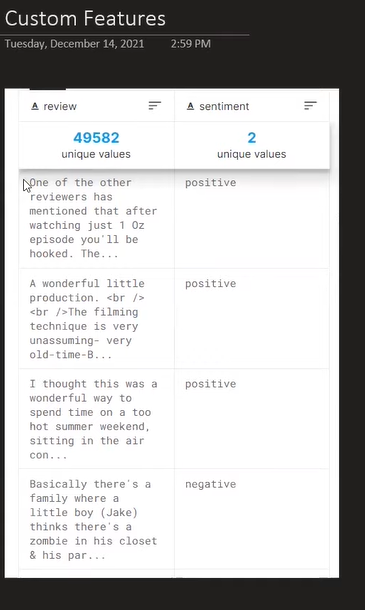
1. **Use Case:** TF-IDF is highly effective and widely used in **Information Retrieval Systems**, such as search engines, to determine which documents are most relevant to a specific search query.

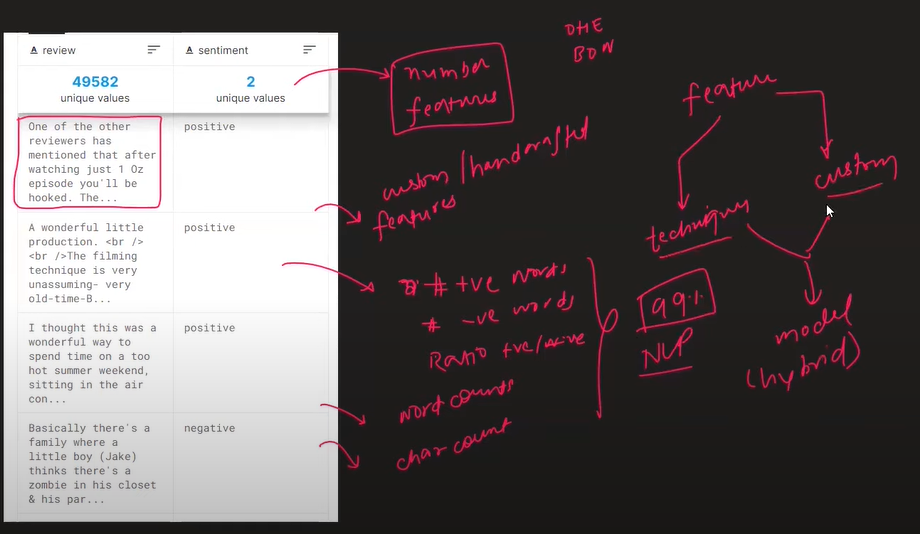
### **E. Disadvantages**

1. **Sparsity:** Like BoW and OHE, TF-IDF results in sparse matrices if the vocabulary is large.
2. **OOV Problem:** It cannot handle words that are outside the vocabulary during prediction.
3. **No Semantic Relationship:** TF-IDF still treats semantically similar words (e.g., "beautiful" and "gorgeous") as entirely separate terms, failing to capture their relatedness in meaning.

## **VII. Custom (Handcrafted) Features**

Custom features (or hand-crafted features) are designed manually based on the specific problem and domain knowledge.





1. **Examples (for Sentiment Analysis):**
   * Number of positive words in the text.
   * Number of negative words in the text.
   * Ratio of positive to negative words.
   * Word count or character count (e.g., if negative reviews tend to be shorter).
2. **Hybrid Approach:** Most real-world projects use **hybrid features**, combining standard vectorisation techniques (like BoW or TF-IDF) with these hand-crafted features to provide a robust input to the ML model.

## **VIII. Future Topics (Deep Learning Approaches)**

The limitations of OHE, BoW, N-grams, and TF-IDF—specifically their inability to fully capture deep semantic relationships between words (like between "beautiful" and "gorgeous")—lead to the need for Deep Learning-based techniques.

* Topics like **Word2Vec** and **Embeddings** use deep learning to map words and sentences into vector spaces where geometric proximity accurately reflects semantic meaning.

## **IX. Assigned Tasks**

The assignment encourages applying all learned techniques to a provided dataset of movie reviews:

1. Apply pre-processing techniques (lowercasing, stop word removal, etc.).
2. Determine the total number of words in the corpus and the total number of unique words (vocabulary size).
3. Apply One-Hot Encoding.
4. Apply Bag of Words, find the vocabulary, and determine the frequency of all words.
5. Apply Bag of Bi-grams and Tri-grams and analyse the difference in vocabulary size.
6. Apply TF-IDF to calculate and print the IDF scores and vocabulary.

