Text representation in NLP

- The raw text corpus is preprocessed and transformed into a number of text representations that are input to the machine learning model.
- Data preprocessing tasks such as tokenization, stopword removal, punctuation removal, stemming, and many more.
- Clean the data of any noise present.
- This cleaned data is represented in various forms according to the purpose of the application and the input requirements of the machine learning model.
- Process of converting text into numbers vectorization because when text is converted in numbers it is in vector form

Common Terminology used while representing text in NLP

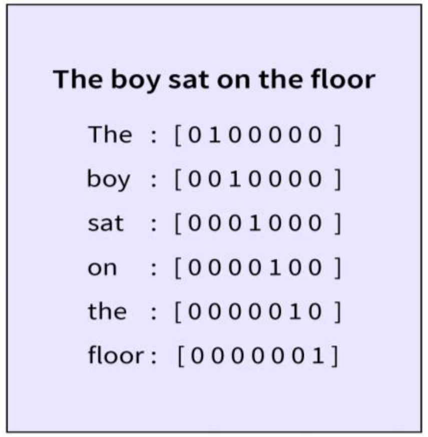
- Corpus(C):
  - All the text data or records of the dataset together are known as a corpus.
- Vocabulary(V):
  - This consists of all the unique words present in the corpus.
- Document(D):
  One single text record of the dataset is a Document.
- Word(W):
  - The words present in the vocabulary.

Types of text Representation in NLP

- Discrete text representation
  - The individual words in the corpus are mutually exclusive and independent of one another.
  - Eg: One Hot Encoding, Bag of words, Count Vectorizer, TF-IDF, Ngrams
- Distributed text representation
  - Distributed text representation thrives on the co-dependence of words in the corpus.
  - The information about a word is spread along the vector which represents it.
  - Eg: Word Embeddings.

One Hot Encoding

- Every word in a text corpus is assigned a vector that consists of 0 and 1.
- This vector is termed the one hot vector in NLP.
- Every word is assigned a unique hot vector.
- The one hot vector contains a single "1", the rest all being zeroes.
- The "1" here is present in the position corresponding to the position of the word in the sentence or in the vocabulary.
- Here the words “The and "the" have different one-hot vectors. We can avoid this by applying lower casing to all the words during text preprocessing.
- It is implemented through pxq matrix where p are the number of words in a sentence and q is the number of words in the vocabulary.



The boy sat on the floor

```
The  : [ 0 1 0 0 0 0 0 ]
boy  : [ 0 0 1 0 0 0 0 ]
sat  : [ 0 0 0 1 0 0 0 ]
on   : [ 0 0 0 0 1 0 0 ]
the  : [ 0 0 0 0 0 1 0 ]
floor: [ 0 0 0 0 0 0 1 ]
```

Advantage and Limitation of One Hot Encoding

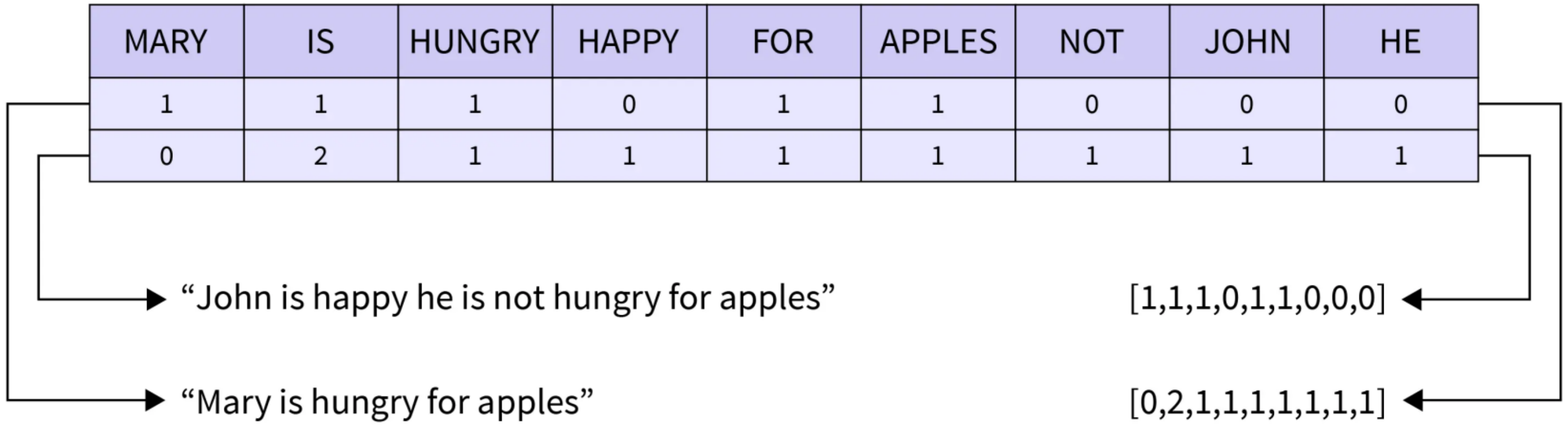
- Easy to understand and implement
- Highly sparse matrix with each sentence, the majority of the values are zero.
- If the documents are of different sizes, we get different-sized vectors. Hence applying machine learning algorithms are little bit difficult.
- This representation does not capture the semantic meaning of the words.
- Eg. Sent1: I need help. Sent2: I need assistance.
- It suffer from out of Vocab problem because if we found some words are in the documents which are not present in the Vocabulary, then we find the same vector for each undefined word.
- Consume too much computer memory and other resources.
- Ordering problem[eg. Hello! How are you? is not same as Hello! How are you?]
- The size of a one-hot vector is directly proportional to the size of the vocabulary of the corpus.
- As the size of the corpus increases, the size of the vector also increases.
- Hence it is not feasible to use one-hot encoding for very large corpora.

Bag of Words

- Converts a whole piece of text into fixed-length vectors.
- The process is done by counting the number of times a word has appeared in the document.
- The frequency count of words helps us to compare and contrast documents.
- This techniques has a variety of applications such as topic modeling, document classification, spam email detection, and many more.
- the bag of words is implemented using CountVectorizer.
- It is represented using PxQ matrix where P is the number of documents and Q is the number of words in the vocabulary list.

Count Vectorizer:

- Count Vectorizer implements the bag words concept by computing the frequency of each word in the corpus and creating a matrix with the sentences(or their position) as rows, and words as columns and filling it with the words' corresponding frequencies.
- The frequency can also denote the weightage of a word in a sentence. These weights can be leveraged for different types of analysis and used for training ML models.



MARY	IS	HUNGRY	HAPPY	FOR	APPLES	NOT	JOHN	HE
1	1	1	0	1	1	0	0	0
0	2	1	1	1	1	1	1	1

"John is happy he is not hungry for apples" [1,1,1,0,1,1,0,0,0]

"Mary is hungry for apples" [0,2,1,1,1,1,1,1,1]

Advantage and Limitation of CountVectorizer

- Adv:
  - Easy to implement
  - Semantic relationship capture better than one hot encoding
  - We get the frequency of each word
  - The size of the vector is the size of the dictionary.
  - Used to omit rarely occurred word.
  - Size of vector corresponding to every documents is same.
- Limitation:
  - We do not extract accurate semantic meaning from the words.
  - This method also loses the positional information of a word.
  - High-frequency words do not necessarily have the highest weightage.
  - On an advanced level, the bag of words is implemented using TF-IDF representation.

TF-IDF

- This representation of text in NLP overcomes some of the drawbacks of CountVectorizer.
- There is a need to suppress high-frequency words and ignore very low-frequency words so that we can normalize the weights.
- The word which appears multiple times in a document and rarely in the corpus should have a good weightage.
- This weightage is calculated using two formulas:-
  - Term Frequency and
  - Inverse Document Frequency.

- Term Frequency: The ratio of the number of occurrences of a word in a document to the number of terms in the document is called term frequency.
- *tf*(*t*,*d*)= count of *t* in *d* / number of words in *d*
- "t" is the term and "d" is the number of terms in the .document The tf of the word "love" in the sentence "I love nlp" is 1/3.

- **Inverse document frequency**: Idf of a term is the log of the ratio of the number of documents in the corpus to the number of documents with the term in them.
- idf(t)=log(N/(df+1))
- N is the number of documents and df is the number of documents with the term t. We add 1 to the denominator of the fraction to avoid division by 0.We multiply tf and idf to get the tf-idf weightage.
- The tf-idf of the term t in document d is:
- $tf-idf(t,d)=tf(t,d)*\log(N/(df+1))$
-

Advantage and Limitation of TF_IDF

- Advantages
  - Easy to understand and interpret.
  - Penalizes both high and low-frequency words. So in a way, IDF achieves in reducing noise in our matrix.
- Disadvantages
  - We do not capture the positional information of a word.
  - Semantic meanings of the words are not captured.
  - TF-IDF is highly corpus dependent.

N-grams

- This representation is similar to the countVectorizer bag words model, the only difference is that instead of calculating the frequency of one word we calculate the frequency of groups of words(in groups of two or more like unigram, bigram, Trigram etc,) occurring together in the corpus.
- In this way, we get a better understanding of the text. Based on the number of words grouped together the model has termed a bigram (2 words), trigram(3 words), and so on.
- They have a wide range of applications, like language models, semantic features, spelling correction, machine translation, text mining, etc.

Example of N-gram

Suppose sentence is “I reside in Bengaluru”.

SL.No.	Type of n-gram	Generated n-grams
1	Unigram	[“I”,”reside”,”in”,”Bengaluru”]
2	Bigram	[“I reside”,”reside in”,”in Bengaluru”]
3	Trigram	[“I reside in”, “reside in Bengaluru”]

Advantages and Limitation of N-grams

- Advantages of Ngrams
  - Ngrams capture the semantic meaning of the sentence and help in finding the relationship between words.
  - Easy to implement
- Disadvantages
  - Out of vocabulary(OOV): We cannot capture the relationships between the words or their semantic meanings if they are not in the vocabulary

Distributed Text Representation:

- When the representation of a word is not independent or mutually exclusive of another word and their configurations often represent various metrics & concepts in data.
- So the information about a word is distributed along the vector it is represented as.
- This is different from discrete representation where each word is considered unique & independent of each others.
  - Co-Occurrence matrix
  - Word2Vec
  - GloVe

Co-Occurrence Matrix:

- It considers the co-occurrence of entities nearby each other.
- It helps us understand the association between different words in a corpus.
- The representation of each word is its corresponding row (or column) in the co-occurrence matrix
- Eg. This is Ramesh. Ramesh is an engineer

	This	is	Ramesh	An
This	O	1	0	0
is	1	0	2	1
Ramesh	0	2	0	0
An	0	1	0	0
Engineer	0	0	0	1

Advantage and limitation of Co-Occurrence Matrix:

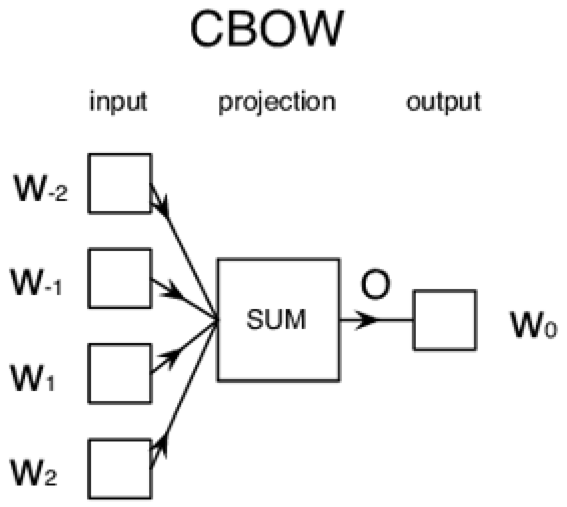
- Advantages:
- Simple representation for finding word associations
- It considers the order of words in the sentence unlike discrete techniques
- The representation coming out of this method is a global representation. i:e it uses the entire corpus for generating the representation
- Disadvantages:
- Similar to CountVectorizer & TF-IDF matrices, this too is a sparse matrix. This means its not storage efficient.
- Larger the vocabulary size, larger the matrix size (not scalable to large vocabulary)
- Not all word associations can be understood using this technique.

Word2Vec

- Word2Vec is a famous algorithm for representing word embeddings.
- It's a prediction-based method for representing words rather than count based technique like co-occurrence matrix.
- Word embeddings are vector representation of a word. Each word is represented by a fixed vector size while capturing its semantic & syntactic relation with other words.
- Word2Vec model is based on the Neural network.
- Input Layer-→ Hidden Layers --→Output layer
- word2vec constructs the vector representation via two methods/techniques:
  - CBOW (Continuous Bag of Words): find target word from context words
  - Skip-Gram : Context word using target word

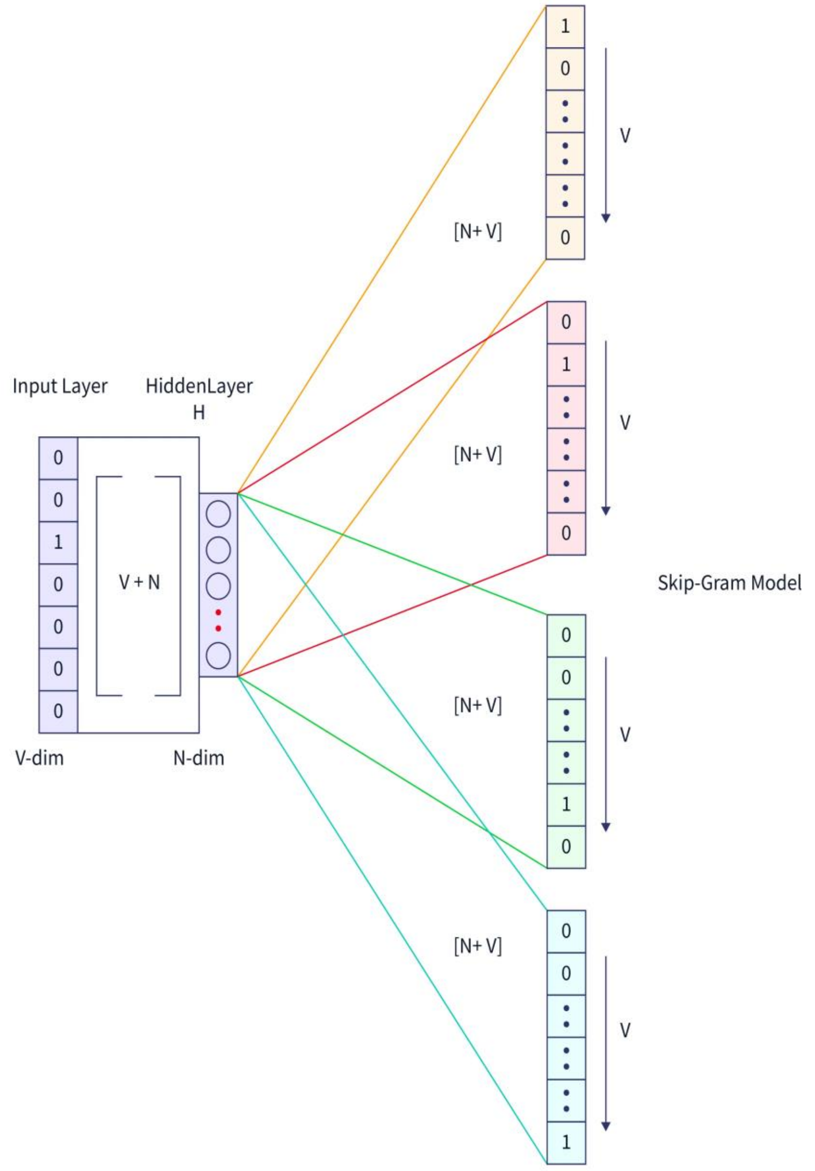
Word2Vec…

- CBOW —
  - Tries to predict the middle word in the context of the surrounding word.
  - So in simple terms, it tries to fill in the blanks as to what word will be more suitable to fit given the context/surrounding words.
  - More efficient with smaller datasets.
  - Window size varies
  - The window size determines the span of words on either side of a target_word .
  - Fast training time compared to Skip-Gram



Word2Vec..

- Skip-Gram —
  - Tries to predict the surrounding context words from a target word (opposite of CBOW).
  - Tends to perform better in the larger dataset but larger training time.
  - Window size varies
  - The window size determines the span of words on either side of a target_word



- In Word2Vec model vector of words are represented by
- Term –Document Matrix: VxD
  - row represents word and column represents documents
- Term-Term Matrix(word –word Matrix or term- context matrix)-VxV
  - Row represents target word and columns represents context word
  - Each cell records the number of times the row(target word)and the column(context word) co-occure in the same context in same training corpus.
  - The context could be the document in which case cell represents the number of times two words appear in the same documents.
- Word2vec is capable of capturing multiple degrees of similarity between words using simple vector arithmetic.
- Patterns like “man is to woman as king is to queen” can be obtained through arithmetic operations like “king” — “man” + “woman” = “queen” where “queen” will be the closest vector representation of the word itself.
- It is also capable of syntactic relationships like present & past tense & semantic relationships like country-capital relationships

Advantage and Limitation of wordtoVec

- Advantages:
  - Capable of capturing relationships between different words including their syntactic & semantic relationships
  - The size of the embedding vector is small & flexible, unlike all the previous algorithms discussed where the size of embedding is proportional to vocabulary size
  - Since its unsupervised, human effort in tagging the data is less
- Disadvantages:
- Word2Vec cannot handle out-of-vocabulary words well. It assigns a random vector representation for OOV words which can be suboptimal
- Parameters for training on new languages cannot be shared. If you want to train word2vec in a new language, you have to start from scratch
- Requires a comparatively larger corpus for the network to converge (especially if using skip-gram)

GloVe

- Global Vectors for word representation is another famous embedding technique used quite often in NLP.
- It tries to overcome the limitation of word2vec by trying to learn both local & global statistics of a word to represent it.
- i:e it tries to encompass the best of count-based technique (co-occurrence matrix) & prediction-based technique (Word2Vec) and hence is also referred to as a hybrid technique for continuous word representation
- It generates vector representations, or embeddings, of words.
- By using the statistical co-occurrence data of words in a given corpus, GloVe is intended to capture the semantic relationships between words.
- GloVe builds a co-occurrence matrix using word pairs and then optimizes the word vectors to minimize the difference between the pointwise mutual information of the corresponding words and the dot product of vectors.

Advantage and Limitation of Glove

- Advantages
- It tends to perform better than word2vec in analogy tasks
- It considers word pair to word pair relationship while constructing the vectors & hence tend to add more meaning to the vectors when compared to vectors constructed from word-word relationships
- GloVe is easier to parallelise compared to Word2Vec hence shorter training time
- Disadvantages
- Because it uses a co-occurrence matrix & global information, memory cost is more in GloVe compared to word2vec
- Similar to word2vec, it does not solve the problem of polysemous words since words & vectors have a one-to-one relationship

Similarity Measure : Cosine Similarity

- Measure similarity between two target vectors(Words) A & B
- Dimension of both vector are same
- Most common similarity metric is the cosine of the angle between the vectors.
- Cosine similarity measure based on the dot product or inner product of the vectors.
- Dot_product(A, B) = A.B= $\sum_{i=1}^{n} A_i B_i$
    - It will tend to high when two vector have large value in the same dimension
    - Vectors that have zero in different dimensions (orthogonal vectors), will have a dot product of zero, representing their strong dissimilarity.
    - Dot product has a problem in measuring similarity because it favors long vectors.
    - Modify the dot product to normalize for the vector length and this normalize dot product turns out to be same as the cosine of the angle between two vectors.
    - Cosine values range from 0-1

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \, ||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Cosine values
  - 1: Vector pointing in the same direction
  - -1: Vector pointing in opposite direction
  - 0- orthogonal
- Since row frequency are non-negative, hence cosine of these values range from 0-1

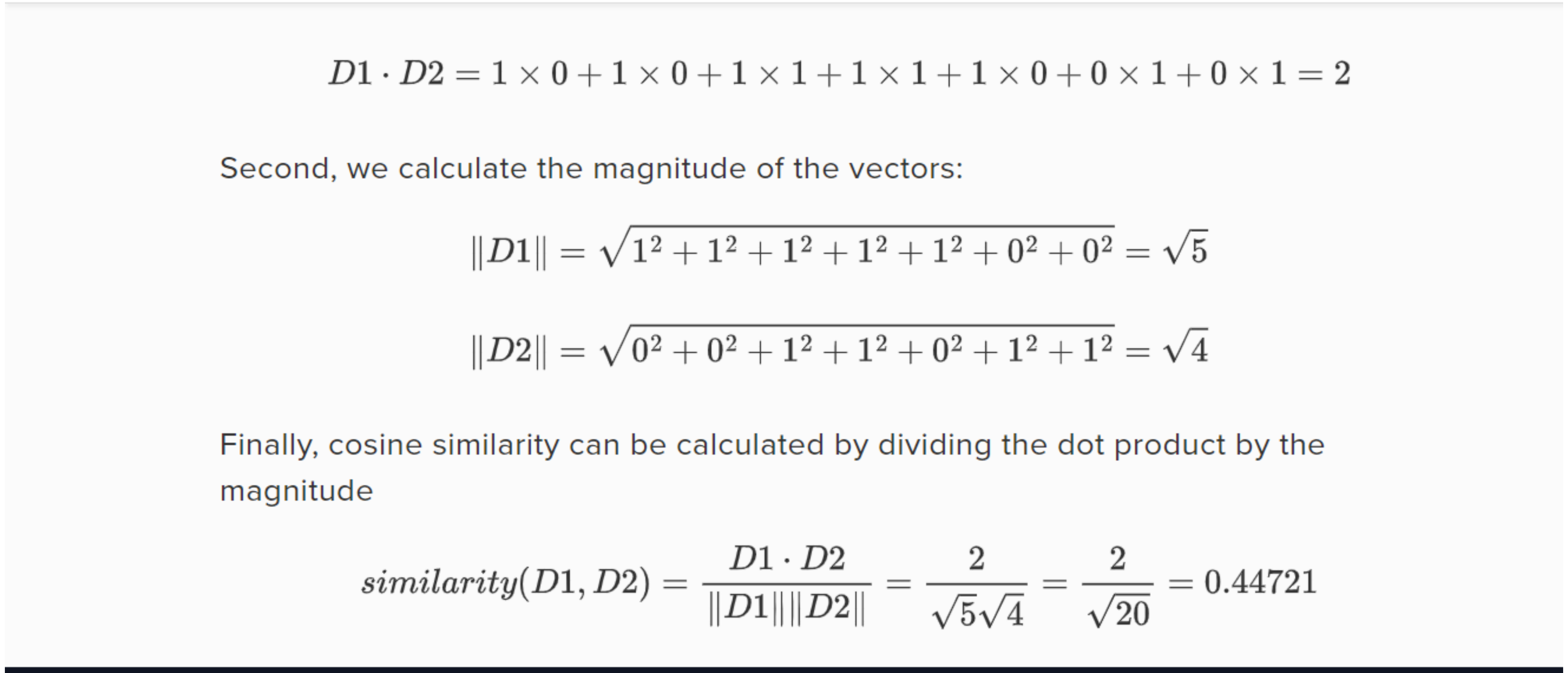
Numerical Example

- Calculate the cosine similarity of the two documents given below.

- Document 1 = 'the best data science course'
- Document 2 = 'data science is popular'
- First creates the word table from documents:

	the	best	data	science	course	is	popular
D1	1	1	1	1	1	0	0
D2	0	0	1	1	0	1	1

- D1 = [1,1,1,1,1,0,0]
- D2 = [0,0,1,1,0,1,1]
- First, we calculate the dot product of the vectors:



$$D1 \cdot D2 = 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 1 = 2$$

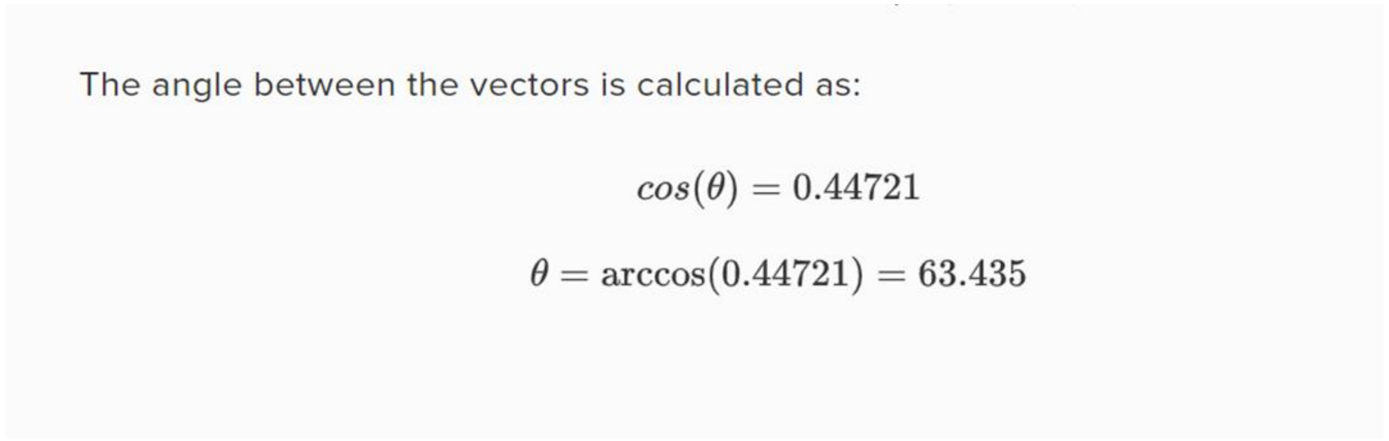
Second, we calculate the magnitude of the vectors:

$$\|D1\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2} = \sqrt{5}$$

$$\|D2\| = \sqrt{0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2} = \sqrt{4}$$

Finally, cosine similarity can be calculated by dividing the dot product by the magnitude

$$similarity(D1, D2) = \frac{D1 \cdot D2}{\|D1\|\|D2\|} = \frac{2}{\sqrt{5}\sqrt{4}} = \frac{2}{\sqrt{20}} = 0.44721$$



The angle between the vectors is calculated as:

$$cos(\theta) = 0.44721$$

$$\theta = \arccos(0.44721) = 63.435$$

Advantage and Limitation

- Cosine similarity is easy to compute, especially with sparse matrices, and it can capture the overall similarity of the documents regardless of their length.
- It does not account for the frequency of the words, so it may not reflect the importance of rare or common terms.
- It also does not consider the order or the context of the words, so it may miss some nuances or semantic differences.

Similarity Measure : Jaccard Similarity

- It is a measure of similarity between two asymmetric binary vectors or we can say a way to find the similarity between two sets.
- It is a common proximity measurement used to compute the similarity of two items, such as two text documents.
- Jaccard similarity between two document d1 and d2 is how many common word exist over total word in two document or two word embedding.
- J(A, B)= length (intersection of document A and B)/ length (union of two document A and B)
- **J(A, B) =** $|A \cap B| / |A \cup B|$
- The index ranges from 0 to 1.
- Range closer to 1 means more similarity in two sets of data.

The Jaccard Distance

- It measures the dissimilarity between two sets. It is calculated as:

- Jaccard Distance = 1 — Jaccard Similarity

- Suppose we have two sets:
- Set A = {1,3,5,7,9}
- Set B = {1,2,3,4,5,6,7,8}
- Number of observation in both: (A∩B) = {1,3,5,7} = 4

- Number of observation in either: (A∪B) = {1,2,3,4,5,6,7,8,9} = 9

- Hence, J(A, B) = |A∩B| / |A∪B| = |4| / |9| = 0.44

- As this number is close to one we can say that the two sets are quite similar.

- Jaccard Distance = 1–0.44 = 0.56

Advantage and Limitation of Jaccard Similarity

- Jaccard similarity is simple and intuitive, and it can account for the frequency of the words, since it only considers the presence or absence of the terms.
- It can also handle documents of different lengths, since it normalizes by the size of the union.
- It does not capture the magnitude or the direction of the vectors, so it may not reflect the strength or the polarity of the similarity.
- It also does not consider the order or the context of the words, so it may miss some syntactic or semantic variations.

Applications of Text Representation

- Text Classification:
  - For the classic problem of text classification, it is important to represent text in vector form so as to process it for classification of text.
  - Used in spam detection, sentiment analysis etc.
- Topic Modelling:
  - The topic modeling use case of NLP requires text to be represented in the correct format to be modeled into different topics.
- Autocorrect Model:
  - The autocorrect model, as the name suggests is the model that is used for correcting spelling errors, for the most part. Now for the suggestions to be accurate, we need our texts represented in a format that uses probability.
- Text Generation:
  - Much like the previous use case, text generation too requires a probabilistic text representation format.

- Thank you