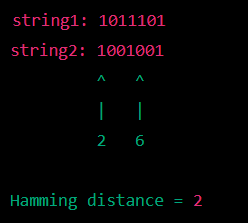
# String similarity metrics

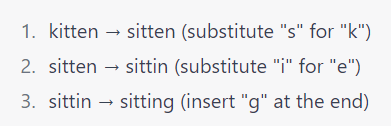
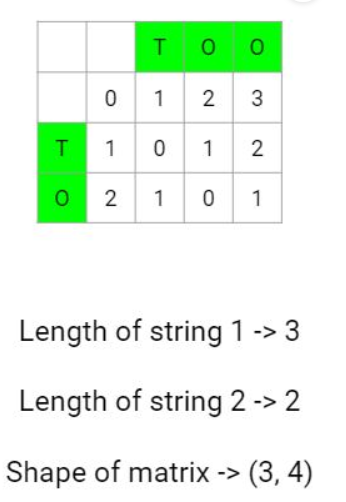
## Hamming distance

* Number of positions at which two strings differ.
* String must have same length.
* Add padding to with shorter if length is different.
* Can be used in spell-checking, plagiarism detection, and DNA sequence analysis



## Levenshtein distance or Minimum edit distance

Number of edit required for that will transform one string to other.

Answer is 3. Answer is 1.

Check if the characters are same

# fill the element with the minimum of (diagonal value, first corresponding value + 1, second corresponding value + 1)

If characters are not same

# fill the element with the minimum of corresponding values + 1

## Jaccard similarity

This measures the size of the intersection of two sets of characters divided by the size of the union of the two sets.

## Cosine similarity

This measures the cosine of the angle between two vectors in a multi-dimensional space, where each string is represented as a vector of character counts.

## Longest common subsequence

This measures the length of the longest sequence of characters that appears in both strings in the same order, but not necessarily consecutively.

Lecture 3:

Manhattan distance better then euclidean distance dealing with outlier

MD = abs difference btn axis

ED = squre root of difference

# Probablistic model:

* N-gram models are a type of Markov mode : probability of a sequence of words based on the probability of the previous n-1 words
* Latent Dirichlet Allocation (LDA) - Generative probabilistic mode, Topic modelling
* Probabilistic context-free grammar (PCFG) : Rule of grammer depends upon the grammer

## Markov Assumption:

the probability of a word depends only on the previous n-1 words

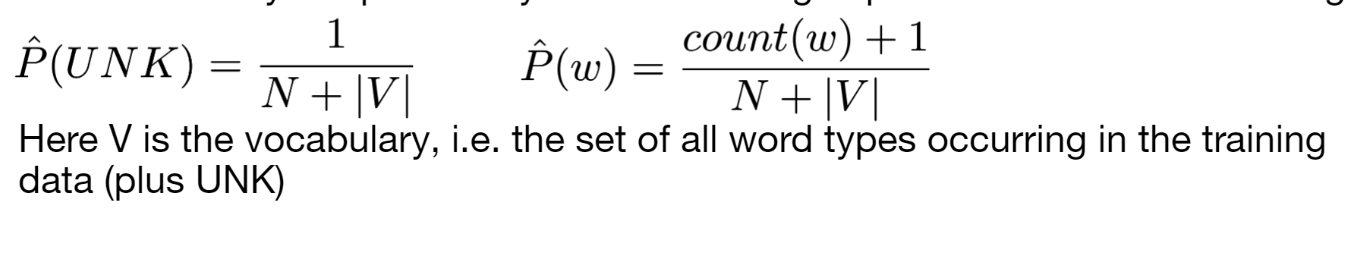
## Problem with unigram model:

* Lack of context:
* Out-of-vocabulary (OOV) words
* Overfitting
* High dimensionality

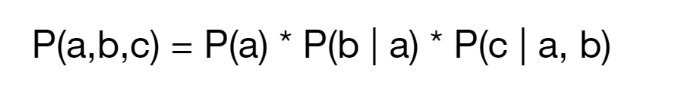
Still, it is very useful in the text classification problem.

## Laplace smoothing

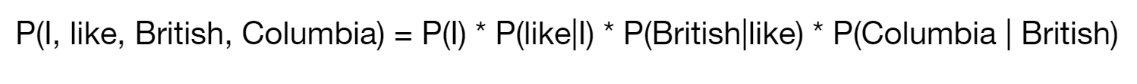
It is known as add-k smoothing. it help to give probablity to unseen data of training set.



Chain of probablity:



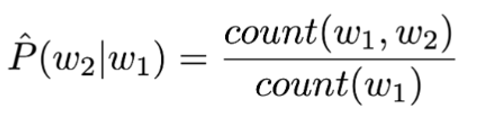
Based on the markov(unigram) :



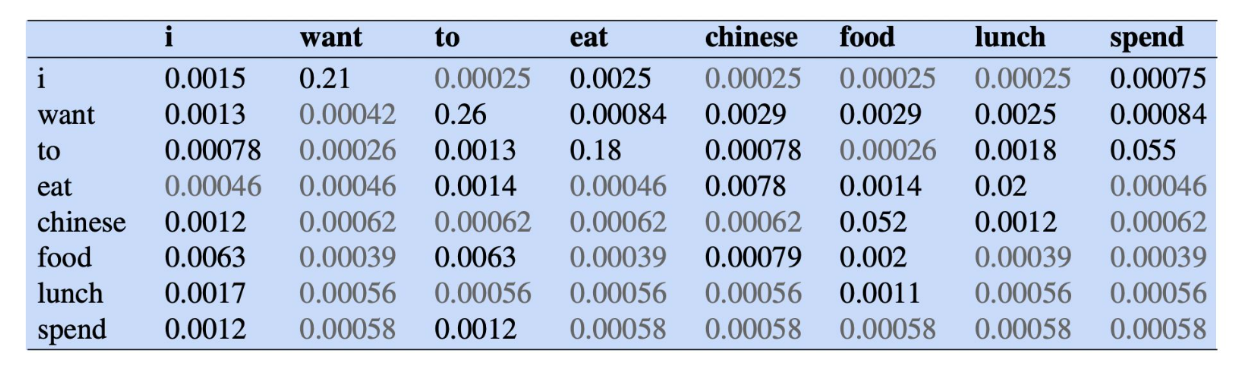
For biggram we need two word after like in 2nd prob.

## Bi gram model

It is also called as Markov chain.

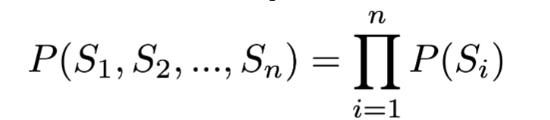


Normalized bi gram model:



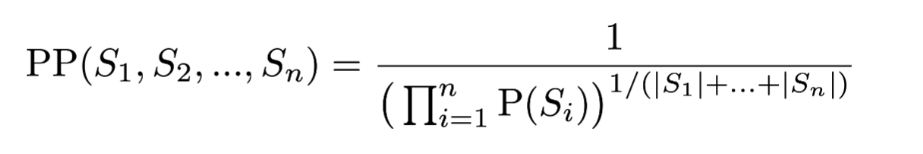
Good way to evaluate the model is test on the unseen data..

Likelihood of a collection of sentences:

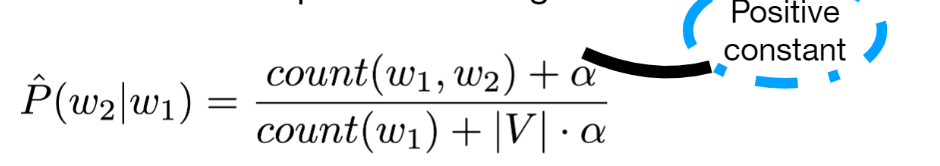


## Perplexity

* For perplexity we measure the normalized inverse above likelihood.
* Perplexity is a measure of how well a probability model is able to predict a given set of observations.
* It is evaluation matrix for the probablity model.



## Adding alpha smoothing



By using hmm with this approach there would be thousands of parameters. So to handle this stituation we can **manage with three state** :

* self.emissions, the list of word types, like “cat”, “dog”
* self.**states**, the list of possible POS tags. “noun”, “verb”
* **Emission distribution** : word is associated with the states like “noun” is probablity distribution for “cat’ and dog may higher then “the”
* 1 x size\_of\_state\_set
* P(w | state = i)
* **Transition probability** : probability of moving from one state to another
* From “noun” to “verb”
* P(state = j | state = i)
* size\_of\_state\_set x size\_of\_state\_set’

With the help of these four HMM to try to find the next word.

## **Generating from an HMM:**

* Generating new observations using new observation using all three states which are similar to used in training.
* It is new training part but we can say prediction part.

### Process steps

The process of generating from an HMM typically involves the following steps:

1. Start in a randomly chosen initial state.
2. Use the transition probabilities to determine the next state.
3. Use the emission probabilities of the current state to determine the next observation.
4. Repeat steps 2 and 3 for a desired number of steps or until a stopping criterion is reached.

Success of right next word : it is based on the **probabilities** learned from the training set.

* This can be achieved by using the **Viterbi Algorithm.**
* P(I,am,Sam; 1,2,3) = P(1|START) \* P(I|1) \* P(2|1) \* P(am|2) \* P(3|2) \* P(Sam|3)

not the same as P(I am Sam)

## **Inference Tasks**

Two states (Rainy and Sunny) and two observations (Umbrella and No Umbrella)

### **Evaluation**

Given a sequence of observations and the HMM parameters, this task calculates the likelihood of the observations being generated by the HMM.

Given observations [Umbrella, No Umbrella, No Umbrella, Umbrella, Umbrella, No Umbrella, No Umbrella].

### **Decoding**

Given a sequence of observations and the HMM parameters, this task finds the most likely sequence of states that generated the observations. This is also known as the Viterbi algorithm.

we have a sequence of observations for the week that is [Umbrella, No Umbrella, No Umbrella, Umbrella, Umbrella, No Umbrella, No Umbrella]. Given the HMM parameters, the most likely sequence of states that generated the observations is [Rainy, Sunny, Sunny, Rainy, Rainy, Sunny, Sunny].

## Hidden and visible states

### Case 1: Inferring Hidden States in an Unsupervised Manner

The states are completely unknown: In this case, the states are not given and we need to infer them from the data in an **unsupervised** manner. This can be done by using techniques such as clustering or Expectation-Maximization (EM) algorithm to group similar observations together and assign each group to a state.

### Case 2: Known Hidden States

The states aren't actually hidden: In this case, the states can be POS tags that come from an annotated corpus. This means that the states are already known and we don't have to infer them from the data.

It is straightforward to estimate model parameters: With the known states, we can use the annotated corpus to estimate the model parameters such as the initial state probabilities, the emission probabilities, and the transition probabilities.

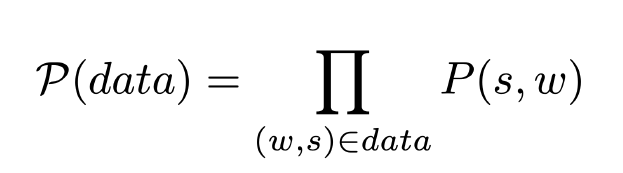
In the 2nd case, it's a **supervised** learning problem, thus it's easier to estimate the model parameters comparing to the 1st case.

## **Training with observable states**

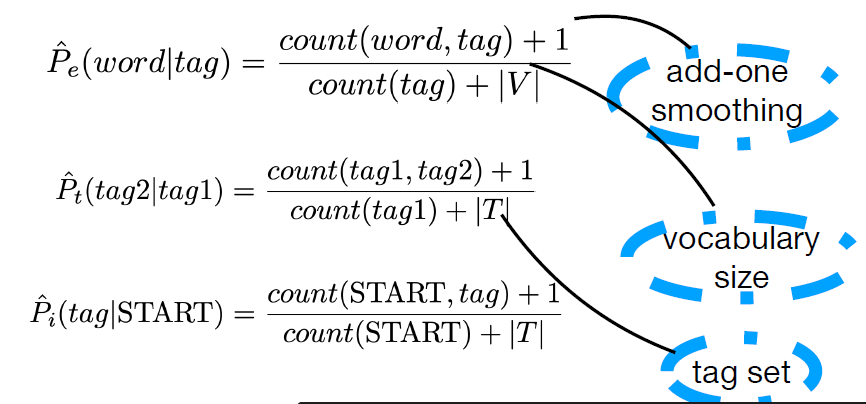
When we train with observable states (e.g. POS tags), the objective is to find model parameters which maximize the joint probability of the states and

emissions in our training corpus:

Where data is a collection of sentences w = w1 ... wk and state sequences s = s1 ... sk



This is called maximum likelihood estimation (MLE)



# Dimensionality reduction technique:

Common use cases : Information retrieval, Text classification, Text clustering

## LSA

LSA it uses singular value decomposition (SVD) on a term-document matrix to extract latent topics from text data.

* Hyperparameter: Number of dimensions, Regularization parameter

## LDA

LDA is a generative probabilistic model that identifies latent topics in text data and projects documents into a lower-dimensional space based on their topic distribution.

* Hyperparameter: n\_component, max\_iter
* It uses the joint probality of word and label.
* Beta : word distribution for the topic
* Theta : topic distribution for the document
* Topic distribution is obtained by lda\_model.transform(cvz)
* Word distribution is obtained by lda\_model.components\_

## Autoencoder:

An autoencoder is a neural network architecture that learns to encode high-dimensional input data into a lower-dimensional representation (latent space) and then decode it back to the original space.

* Hyperparameter: Latent space dimensionality, Regularization parameter, Number of layers in the network

## PCA (Principal Component Analysis):

PCA is a linear dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional space, preserving the maximum variance in the data.

* Hyperparameter: Number of dimensions

## Singular Value Decomposition (SVD):

It is a matrix factorization technique that decomposes a matrix into the product of three matrices - a singular value matrix, a left singular vector matrix, and a right singular vector matrix.

It is internally used by LSA.

## t-SNE (t-distributed Stochastic Neighbor Embedding):

t-SNE is a non-linear dimensionality reduction technique that maps data points into a low-dimensional space while preserving the relationships between them.

* Hyperparameter: Number of dimensions, Perplexity
* Use case: Word embedding visualization, Document clustering

## GloVe (Global Vectors for Word Representation):

GloVe is a word embedding technique that maps words into a continuous low-dimensional space based on their co-occurrence statistics.

* Hyperparameter: Dimensionality of the word vector, Context window size, Regularization parameter
* Use case: Word similarity calculation

## UMAP (Uniform Manifold Approximation and Projection):

UMAP it uses topological information to preserve the global structure of the data in the low-dimensional space.

* Hyperparameter: Number of dimensions, Regularization parameter
* Use case: Word embedding visualization

# SVD

SVD is reducing the dimensionality of the data into 2 D, using linear algebra-based technique. We can say decompose a matrix into its constituent parts.

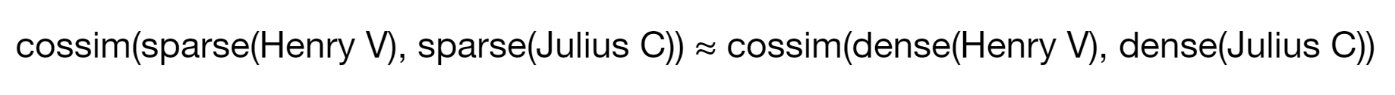
Convert sparse matrix to dense matrix

## Application:

SVD has a wide range of applications in data analysis, including:

* Image compression
* Data mining
* Natural Language Processing (NLP): SVD can be used to analyze text data, such as in topic modeling and sentiment analysis.
* Recommender systems: SVD can be used to factorize a user-item matrix, making it possible to recommend items to users based on their past behavior.

## SVD Aim: make sure compressed version have same cossim



## Rank in SVD

* Rank one matrix is product of column and row vector.
* Rank two is sum of two rank one vector.

Rank of a matrix is used to determine the number of singular values and singular vectors needed to decompose the matrix.

A matrix with a higher rank will have more linearly independent rows or columns, require more singular values and singular vectors to decompose.

Turncated SVD:

Approximate a matrix using a smaller number of its singular values and corresponding singular vectors

### Turncated SVD steps

* perform svd on original matrix it will convert into 3 matrix; u, v , s
* Select the number of singular values and singular vectors
* Form the approximation matrix
* Validate the approximation use metrics like Frobenius norm or the relative error to measure the difference between the two matrices.

We can use elbow method to find dimensionality.

# T\_SNE

T-SNE is visulizing the data in 2 or 3 D form and non-linear technique.

* maps the data points to a lower-dimensional space while preserving the local structure of the data

## T\_SNE application

T-SNE has a wide range of applications in data visualization, including:

* Visualizing high-dimensional datasets
* Exploring complex data structures
* Identifying customer segments: T-SNE can be used to visualize patterns in customer data, making it easier to identify customer segments and target marketing efforts.
* Bioinformatics: T-SNE can be used to visualize gene expression data, making it easier to identify patterns and relationships in large, complex datasets.
* Natural Language Processing (NLP): T-SNE can be used to visualize word embeddings, making it easier to understand the relationships between words in a text.

## Hyperparameter

### Perplexity

Measure expected density of the cluster.

higher perplexity is better but upto a limit. If value is too high then the it will create one cluster. It should be less then the number of the data points.

### Eplison

Higher eplison put data toghter. Smaller e is better in genreral.

SVD is primarily used for data compression and feature extraction, while T-SNE is primarily used for data visualization

# TF\_IDF:

Importance of word in the documents. We need to remove

## TF

* Number of time a word appear in a documents.
* Like dog appear 10 times in a document.

## IDF

* how common a word appear in the different documents.
* Like dog appear 100 times in 1000 documents so idf will log(100/10000)

noisy channel model?

# CountVectorizer(document term matrix)

* It is used to convert text into token
* Each row represent document
* Each column represents count

Hyperaparameter :

* Lowertype = true
* Token\_pattern = ‘\b\w+\b’ only word will considered
* Min\_df = 5 if token appear less then 5 will be ignored
* Max\_df =.1 any token appears in more than 10% of the document will be ignoed.

# Normalize function

It is used to scale a matrx so that its individual values have a unit norm.

## Type of normalization:

L1 = is known as manhattan norammlization

It normalize all values in such way that sum of square is equal to 1.

L2 = euclidean normalization

Sum of square of all the values for each row and columns equal to 1.

## Library and use

from sklearn.preprocessing import normalize

noramlize(data, axis = 1, norm = ‘l2’) # it will return normalised datasets.

# MiniBatchKMeans

* Designed to handle large datasets
* It is same as kmeans clustering
* Lloyd's algorithm to iteratively find the cluster centroids and assign each data point to the closest centroid.
* It uses mini-batches to reduce the computation time
* It take small random subsets of data to compute the centriods
* It is memory efficients

## Cluster evaluation metirx

There are different ways to measure the similarity between the cluster .

### [Adjusted rand index](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html)

It is used to compare the similarity betweeen two clusters.

Ranges : -1 ot 1

1 for similarity , -1 for dissimilarity

## NMI : normalized mutual information

Ranges : 0 to 1

LDA  
Latent dirichlet allocation is generative statstical model used in topic classification, topic discovery, and document summarization. Perplexity used to measure of how well a model fits the data.

## Generative statstical

Generative statstical model can generate new data samples that similar to training data.

They use underlying probablity distribution of the training model.

# Wellness of a model:

Perplexity: It measures how well a probability model predicts a sample. Lower perplexity indicates the model is better at predicting the sample.

Coherence: It measures the degree of semantic similarity between high-probability words within the same topic. A model with high coherence is considered to have interpretable and semantically meaningful topics.

Topic Coherence: It measures the semantic similarity between the top n words in a topic and how they relate to each other.

BLEU Score: It compares a candidate translation of text to one or more reference translations. Higher BLEU scores indicate that the candidate translation is closer to the reference translations.

ROUGE Score: It compares a candidate summary to one or more reference summaries. Higher ROUGE scores indicate that the candidate summary is closer to the reference summaries.

METEOR Score: It is an automatic evaluation metric that is based on the harmonic mean of unigram precision and recall between the candidate and reference translations.

F1 Score: It is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score. The F1 score can be interpreted as a weighted average of the precision and recall.

Accuracy: It is a measure of how well a model correctly predicts the class of a sample.

Log-Likelihood: It is a measure of the likelihood of the observed data given the model. A higher log-likelihood indicates that the model is a better fit for the data.

# **Topic Modeling**

Process to find the topics from documents in an unsupervised manner.

Also popularly known as documents to topic districution and topic to term distribution.

1. Document categorization.
2. Document summrizatiom
3. Dimensionality reduction
4. Informal retrival
5. Recommendation engines

**Techniques:**

LDA – Latent Dirichlet Allocation

NNMF - Non Negative Mattrix Factorization

LSA – Latent Semantic Allocation

## LDA

It is generative probablistics algorithm. Two important assumptoin documents are mixers of Topics and second topics are mixture of term.

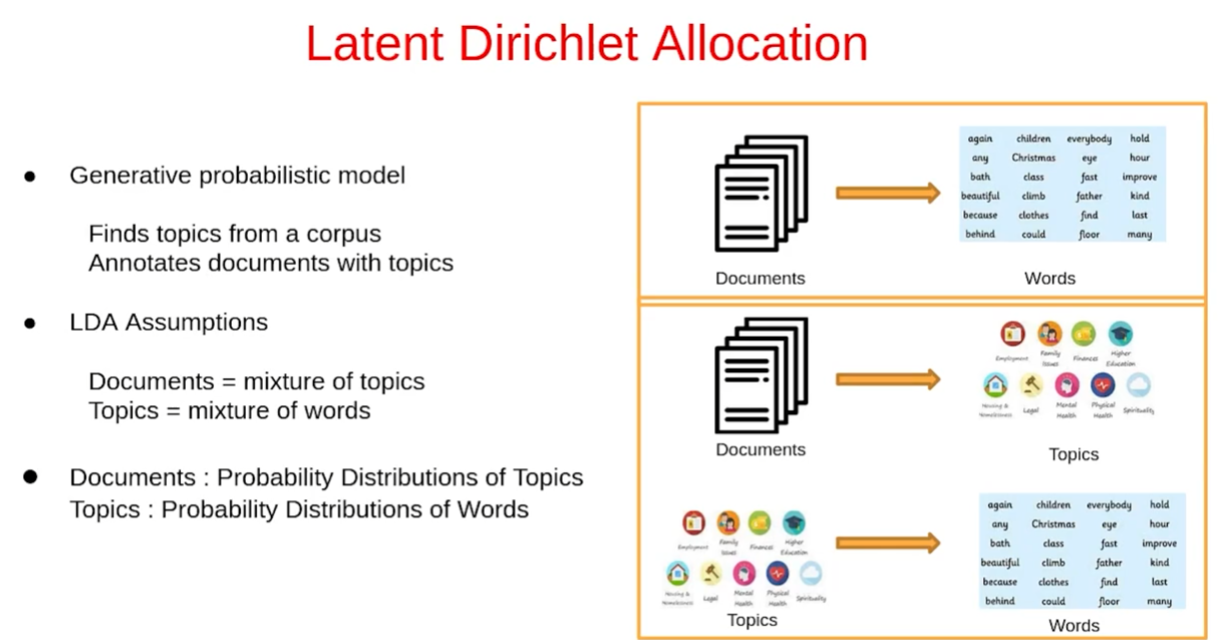
GPA – find topics from corpus and assign topics to documents

🡪 mixtures

Documents 🡪 topics 🡪 terms

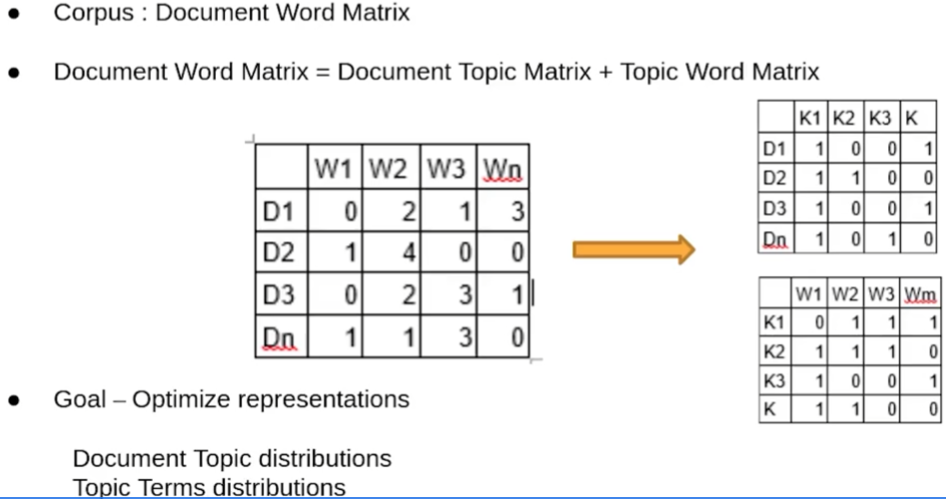
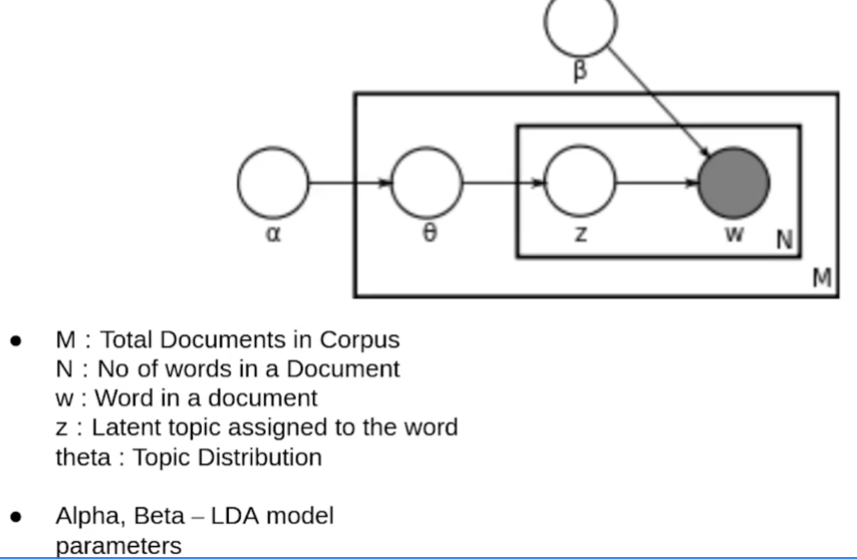
Documents = probablity distributions of topics

Topics = probablity distributions of words.



Corpus = document word matrix

document word matrix = document topic matrix + topic word matrix

Estimating the parameters

* Collect counts based on random initialization of variables (E-step)
* Pretend the counts are real, and update parameters(M-step)
* Repeat until convergence
* Markov Chain Monte Carlo/Gibbs Sampling : purpose to avoid the local minima and achieve global minima: they generate new sample sets and updating the parameters in a random fashion

Corpus : d1 = ( w1, w2,w3, ….wn)

Firstly assign random topics to each word :

d1 = ( w1(k4), w2(k3),w3(k1), ….wn(k2))

two outcomes:-

* Documents is mixture of topics.
* Topics is mixture of terms.

Topic distribution is obtained by lda\_model.transform(cvz)

Word distribution is obtained by lda\_model.components\_

# Elastic search

Step 1:

from elasticsearch\_dsl import Document, Text, Keyword, analyzer, tokenizer

brown\_analyzer = analyzer('brown', tokenizer="classic", filter=["lowercase","stop"])

Step 2:

class BrownDocument(Document):

#my code here

text = Text(analyzer=brown\_analyzer)

genre = Keyword()

#my code here

Step 3:

from elasticsearch\_dsl import Index

brown\_index = Index("brown")

brown\_index.document(BrownDocument)

brown\_index.create()

Step 4:

#provided code

from nltk.corpus import brown

for fileid in brown.fileids():

text = " ".join(brown.words(fileid))

genre = brown.categories(fileid)[0]

doc = BrownDocument(text=text, genre=genre)

doc.meta.id = fileid

doc.save()

Step 5:

s = brown\_index.search()

Step 6:

s = brown\_index.search()

s = s.query(Match(genre="government"))

s = s[:500]

response = s.execute()

for i, hit in enumerate(response):

print(i+1, hit.text[:100])

we can also use

s = s.query(~(Match(text="black") | Match(text="blue")))

step 7;

Two basic methods which are used to define the search: filter and query