

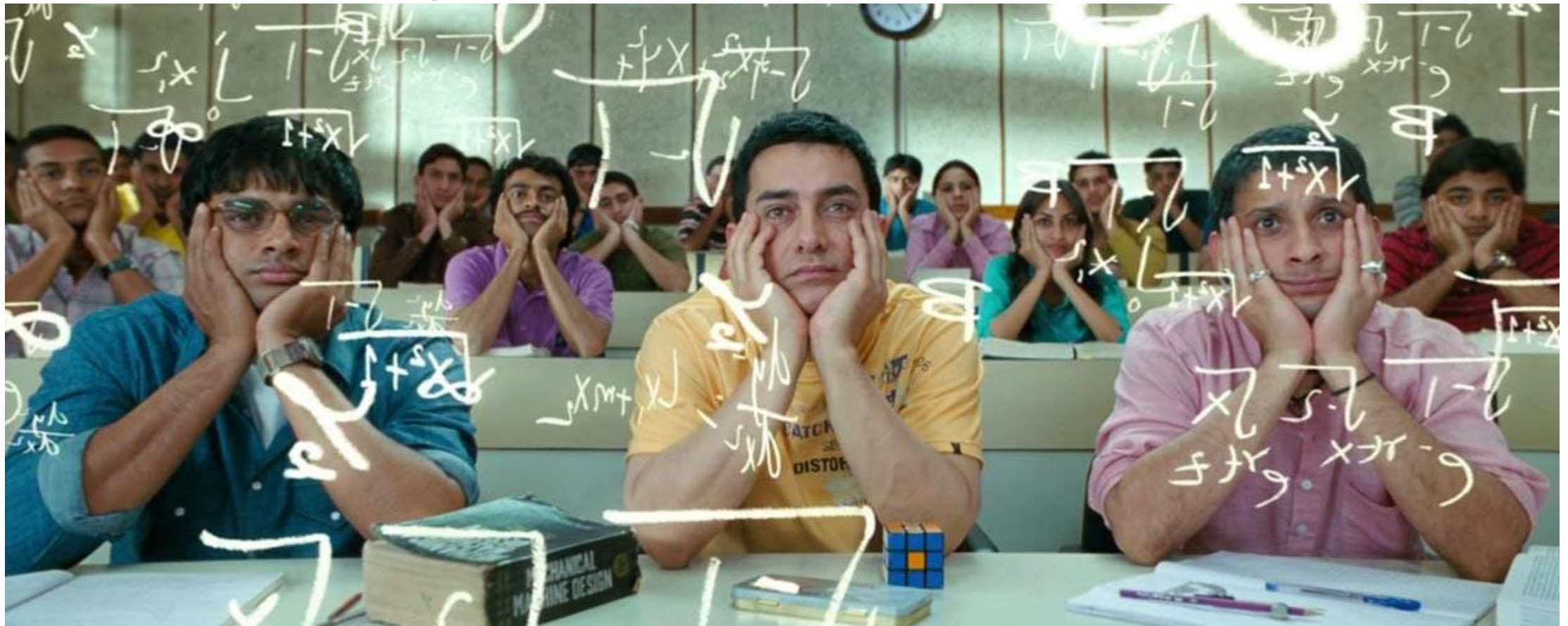
Topic Modelling using Latent Dirichlet Allocation

Arkaprava Sinha [MDS201801](#)

Subhasish Basak [MDS201803](#)

Rohan Khaitan [MDS201812](#)

What to study for
tomorrow's exam??



Problem Statement :

We summarize the aim of our project as follows:

Suggesting important topics for exam preaparation, based on past year question papers.

- Methodologies used:

LDA (Latent Dirichlet Allocation)

Motivation :

As more information becomes available, it becomes more difficult to find and discover what we need.

We need tools to help us organize, search and understand these vast amount of information.

Topic modeling provides methods for automatically organizing, understanding, searching, and summarizing large electronic archives:

1. Discover the hidden themes in the collection
2. Annotate the documents according to these themes.
3. Use annotations to organize, summarize, search, and form predictions.

Why LDA?

Latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics.


Assumptions:

We have a set of documents D_1, D_2, \dots, D_N .

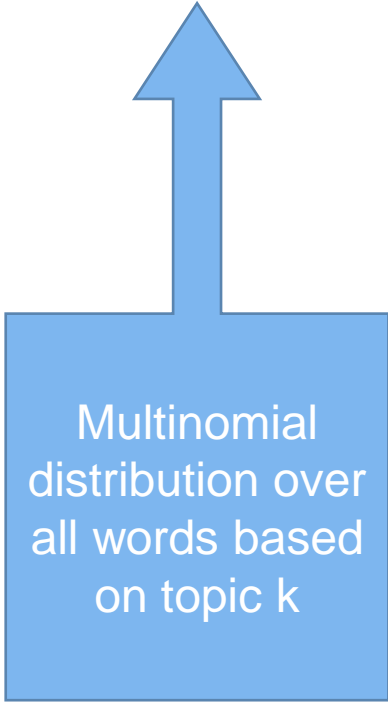
- Each document is just a collection of words or a “bag of words”. Thus, the order of the words and the grammatical role of the words (subject, object, verbs,...) are not considered in the model.
- Words like am/is/are/of/a/the/but/... can be eliminated from the documents as a preprocessing step since they don't carry any information about the “topics”.
- Infact, we can eliminate words that occur in at least 80% ~ 90% of the documents!
- Each document is composed of N “important” or “effective” words, and we want K topics.

Generative models for words in a document :

$$P(word) = \sum_{k=1}^n P(Topic_k) P(word | Topic_k)$$

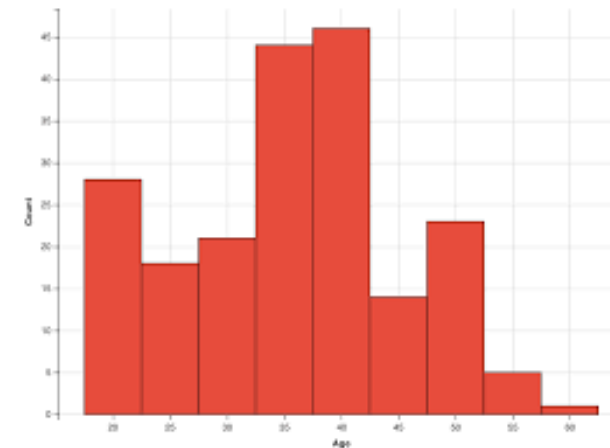
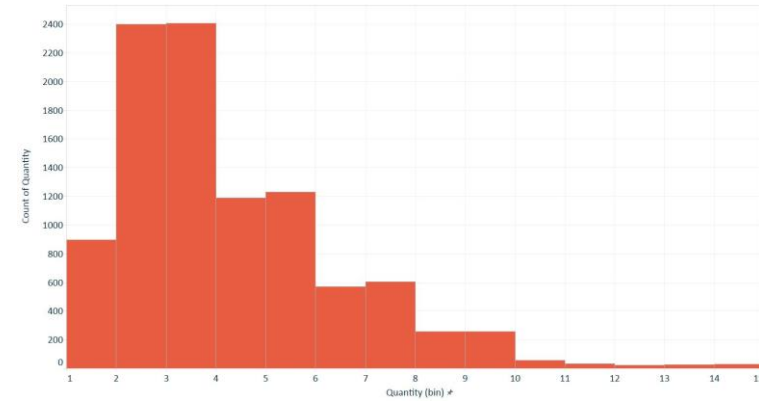
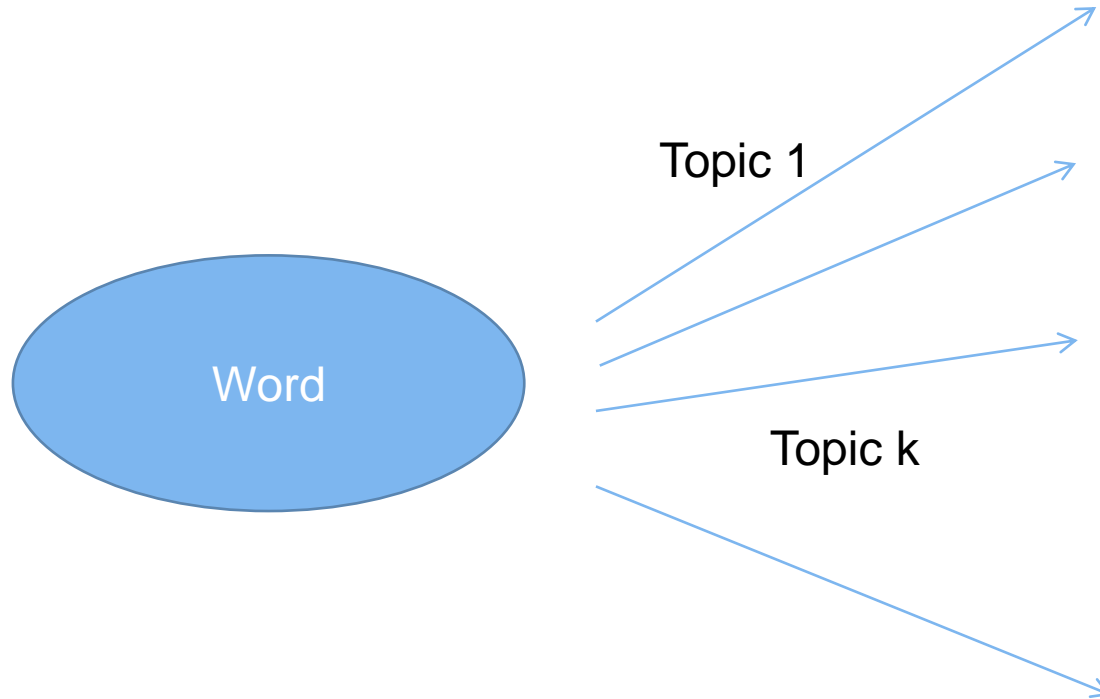


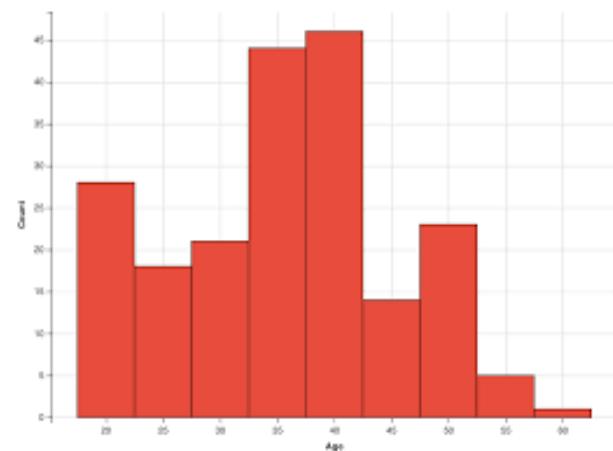
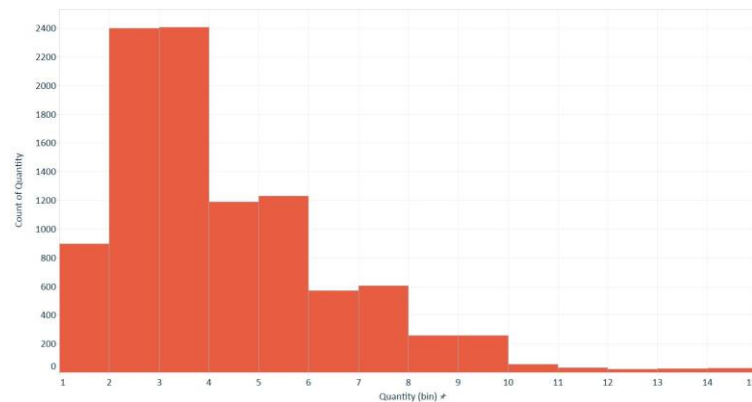
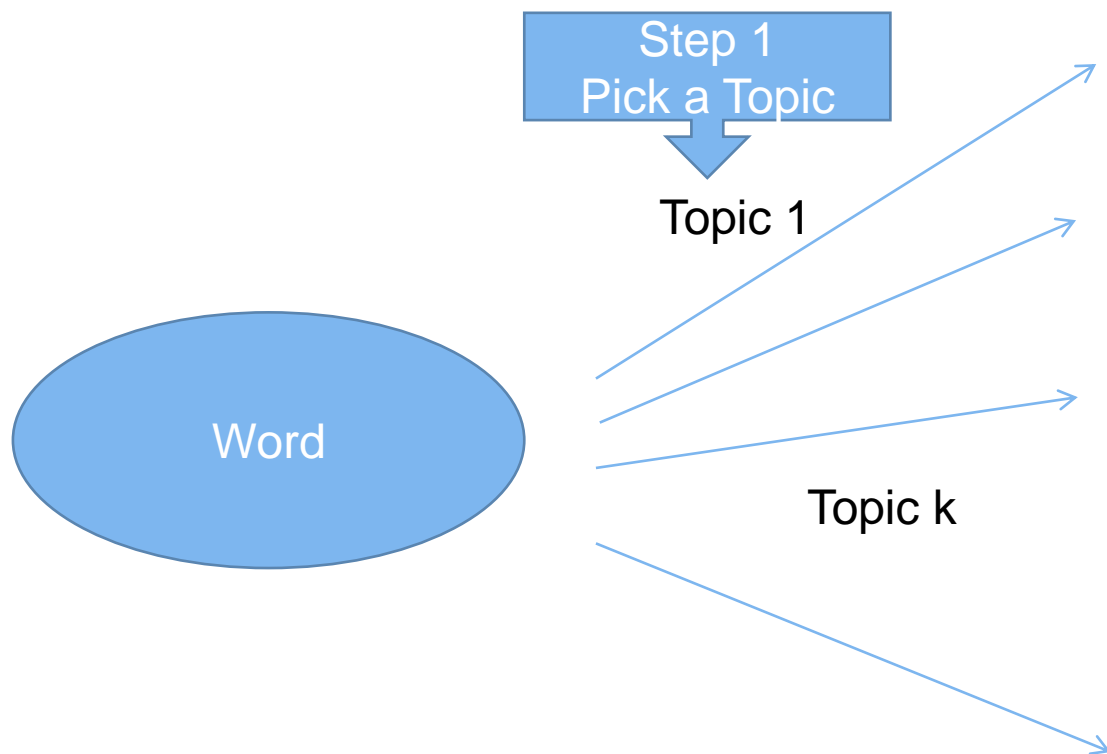
Mixture weight
for topic k

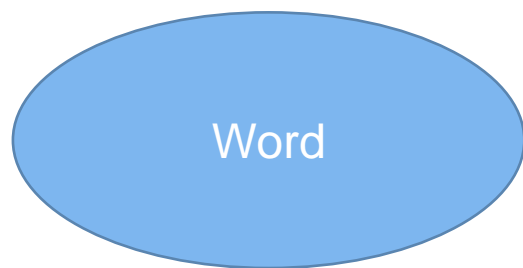


Multinomial
distribution over
all words based
on topic k

Mixture of model :



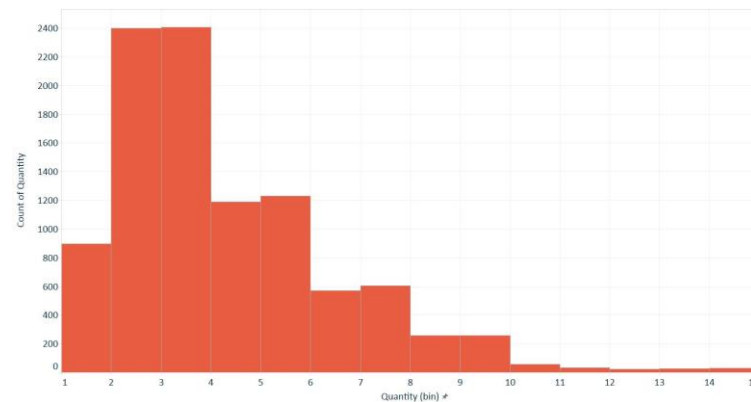




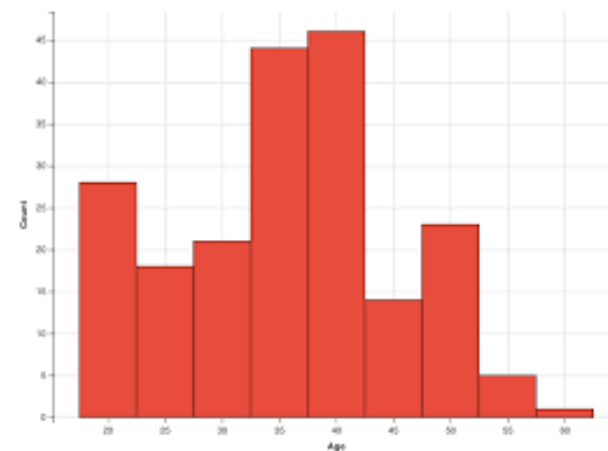
Step 1
Pick a Topic

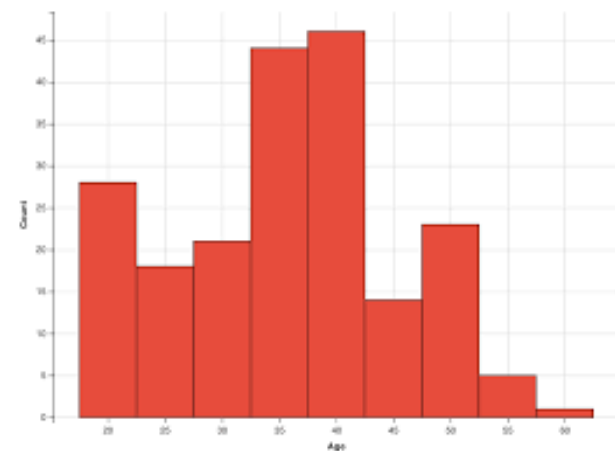
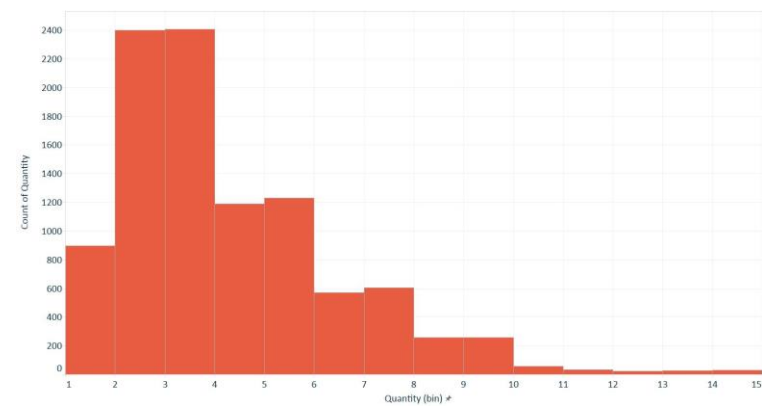
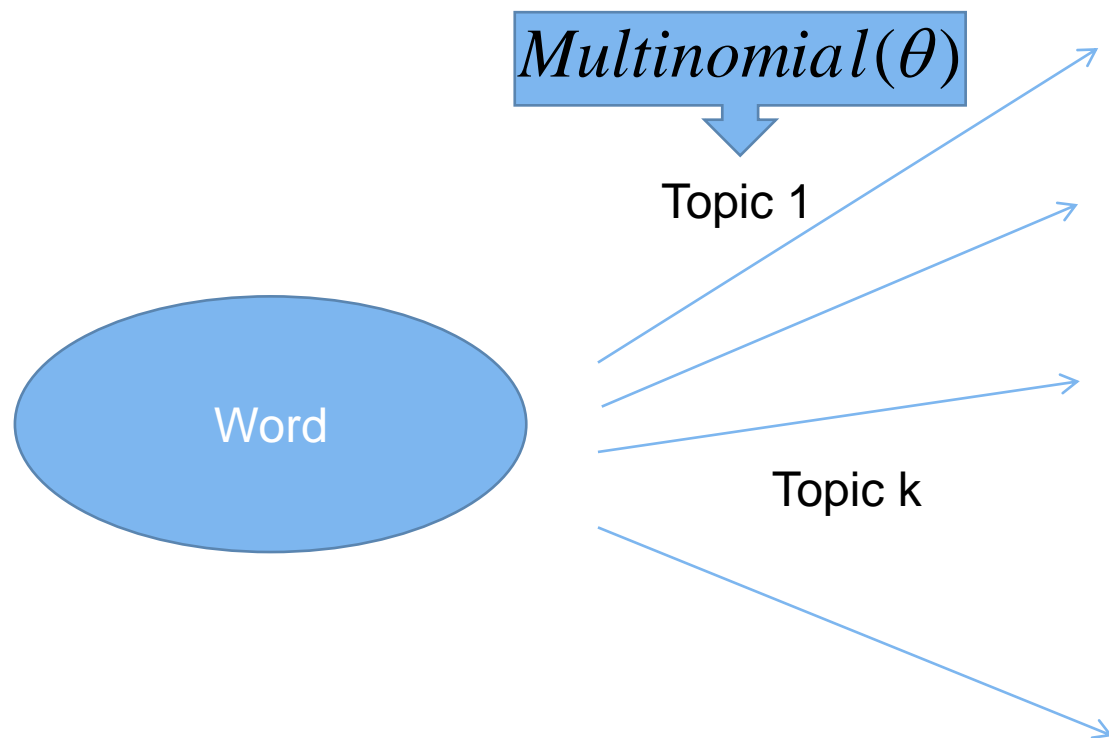
Topic 1

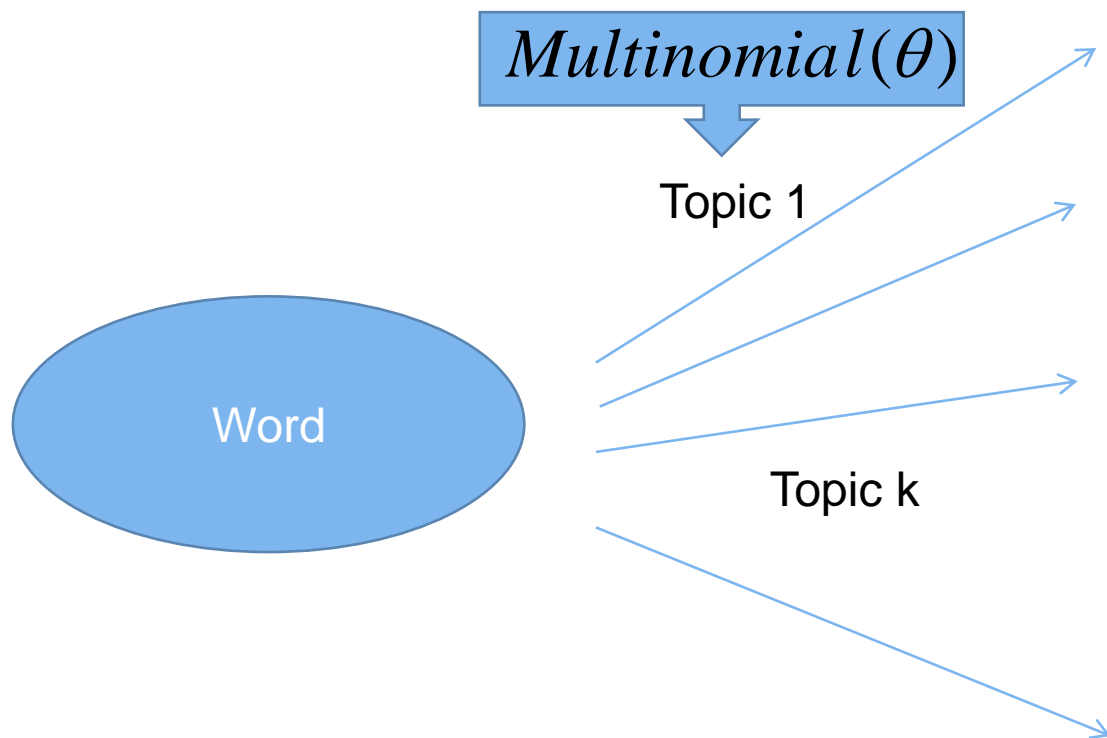
Topic k



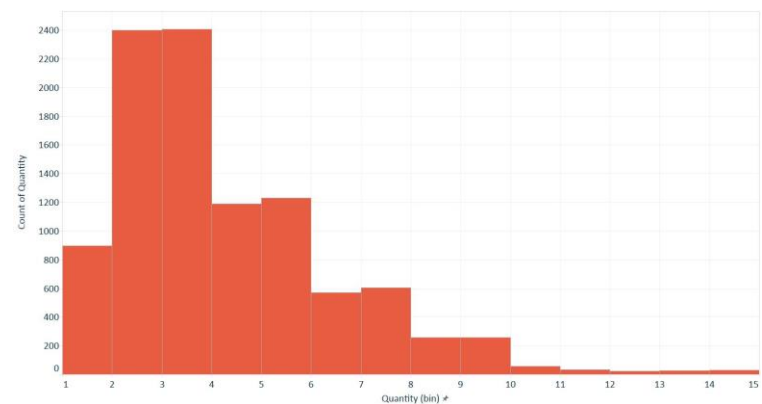
Step 2
Pick a word



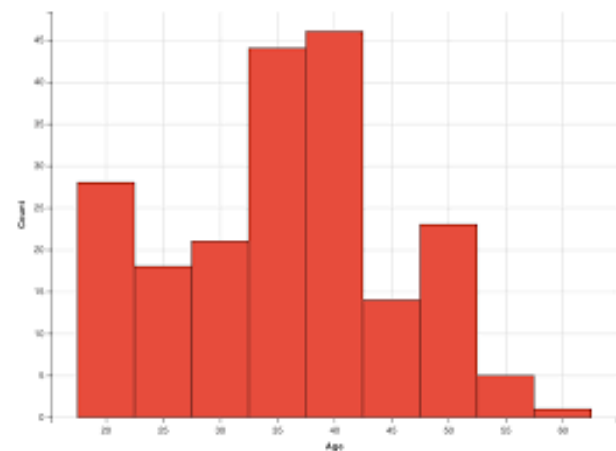




For a document d let the selected topic be Z



$Multinomial(\beta_k)$



Summarization of the Model :

$$Z \mid \theta \quad \sim \quad \textit{Multinomial}(\theta)$$

$$W \mid Z, \beta \quad \sim \quad \textit{Multinomial}(\beta_k)$$

Now the distributions of θ and β_k are approximated by **Dirichlet distribution**,

$$\beta_k \mid \eta \quad \sim \quad \textit{Dirichlet}(\eta)$$

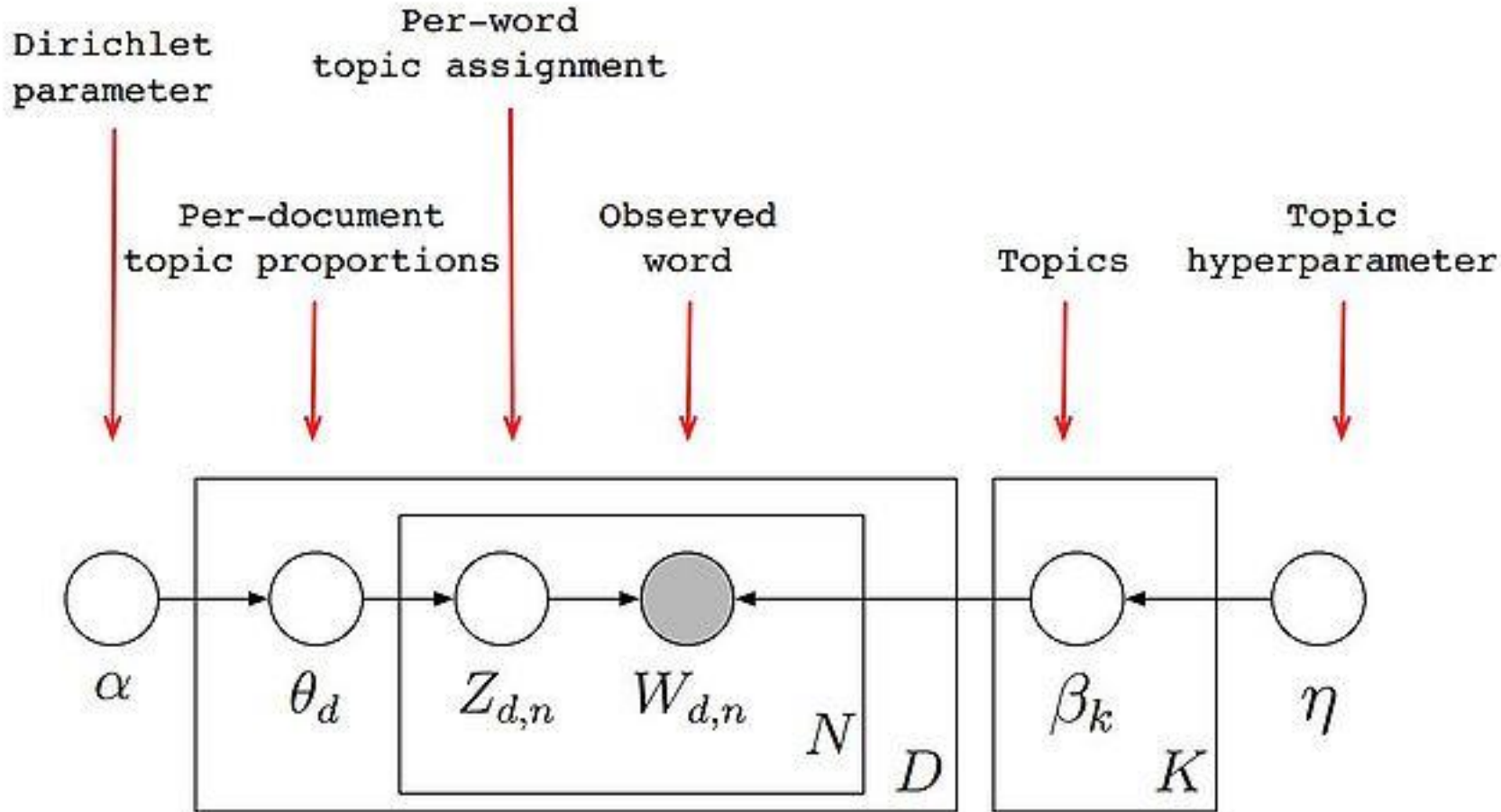
$$\theta \quad \sim \quad \textit{Dirichlet}(\alpha)$$

Understanding the generative model with an example:

In more detail, LDA represents documents as mixtures of topics that spit out words with certain probabilities. It assumes that documents are produced in the following fashion: when writing each document, you

- Decide on the number of words N the document will have (say, according to a Poisson distribution).
- Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics). For example, assuming that we have the two food and cute animal topics above, you might choose the document to consist of $1/3$ food and $2/3$ cute animals.
- Generate each word w_i in the document by:
 - First picking a topic (according to the multinomial distribution that you sampled above; for example, you might pick the food topic with $1/3$ probability and the cute animals topic with $2/3$ probability).
 - Using the topic to generate the word itself (according to the topic's multinomial distribution). For example, if we selected the food topic, we might generate the word “broccoli” with 30% probability, “bananas” with 15% probability, and so on.
- Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

Pictorially we have:



Shaded nodes are observed, and unshaded nodes are hidden.

- Multinomial distribution :

$$P(X_1 = x_1, \dots, X_K = x_K) = \frac{n!}{x_1! \dots x_K!} p_1^{x_1} \dots p_K^{x_K} \quad X_i \in \{0, \dots, n\} \quad \sum_{i=1}^K X_i = n$$

- Dirichlet distribution :

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \theta_1^{\alpha_1-1} \dots \theta_K^{\alpha_K-1} \quad \alpha = k - \text{dimensional vector} \quad \alpha_i > 0$$

variable θ can take values in the $(k - 1)$ simplex: $\theta_i > 0$ and $\sum_{i=1}^K \theta_i = 1$

The generative model follows these steps:

1. Draw each topic $\beta_i \sim \text{Dir}(\eta)$ for $i = 1, 2, \dots, k$
2. For each document:

First draw topic proportions $\theta_d \sim \text{Dir}(\alpha)$

For each word within the document:

- a) Draw $Z_{d,n} \sim \text{Multi}(\theta_d)$
- b) Draw $W_{d,n} \sim \text{Multi}(\beta_{z_{d,n}})$

Parameter estimation and inference

- Now from a set of N documents and the observed words within each document, we want to infer the posterior distribution
- There are many approximate posterior inference algorithms for this! We will briefly review Gibbs sampling here as an example.

Simple Gibbs Sampling Algorithm:

- Suppose we wish to sample $\theta_1, \theta_2 \sim p(\theta_1, \theta_2)$ but cannot use direct simulation or some other methods.
- But we can sample from $p(\theta_1|\theta_2)$ and $p(\theta_2|\theta_1)$. Then we use the following Gibbs algorithm:
 1. Initialize $(\theta_1^{(0)}, \theta_2^{(0)})$
 2. Repeat the following steps consecutively to compute $(\theta_1^{(j)}, \theta_2^{(j)})$
 - a) Sample $\theta_1^{(j)} \sim p(\theta_1 | \theta_2^{(j-1)})$
 - b) Sample $\theta_2^{(j)} \sim p(\theta_2 | \theta_1^{(j)})$

Implementation

- Initialization Step : Go through each document, and randomly assign each word in the document to one of the K topics.
- Go through each word w in d and for each topic t , compute two things:
 - 1) $p(\text{topic } t \mid \text{document } d)$ = the proportion of words in document d that are currently assigned to topic t .
 - 2) $p(\text{word } w \mid \text{topic } t)$ = the proportion of assignments to topic t over all documents that come from this word w .

Reassign w a new topic, where we choose topic t with probability

$$p(\text{topic } t \mid \text{document } d) * p(\text{word } w \mid \text{topic } t)$$

(according to our generative model, this is essentially the probability that topic t generated word w , so it makes sense that we resample the current word's topic with this probability).

Convergence

Assumption :

In other words, in this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated.

After repeating the previous step a large number of times, one will eventually reach a roughly steady state where the assignments are pretty good. So these assignments are used to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

Lets code it!!