- Section Goals
  - Understand Topic Modeling
  - Learn Latent Dirichlet Allocation
  - Implement LDA
  - Understand Non-Negative Matrix Factorization
  - Implement NMF
  - Apply LDA and NMF with a project

Topic Modeling Overview

- Before we learn about methods such as Latent Dirichlet Allocation or Non-negative Matrix Factorization, let’s understand the general concept of Topic Modeling and why it’s important!

- Topic Modeling allows for us to efficiently analyze large volumes of text by clustering documents into topics.
- A large amount of text data is **unlabeled** meaning we won't be able to apply our previous supervised learning approaches to create machine learning models for the data!

- If we have **unlabeled** data, then we can attempt to “discover” labels.
- In the case of text data, this means attempting to discover clusters of documents, grouped together by topic.

- A very important idea to keep in mind here is that we don’t know the “correct” topic or “right answer”!
- All we know is that the documents clustered together share similar topic ideas.
- It is up to the user to identify what these topics represent.

- We will begin by examining how Latent Dirichlet Allocation can attempt to discover topics for a corpus of documents!

- Johann Peter Gustav Lejeune Dirichlet was a German mathmatician in the 1800s who contributed widely to the field of modern mathematics.
- There is a probability distribution named after him “Dirichlet Distribution”.

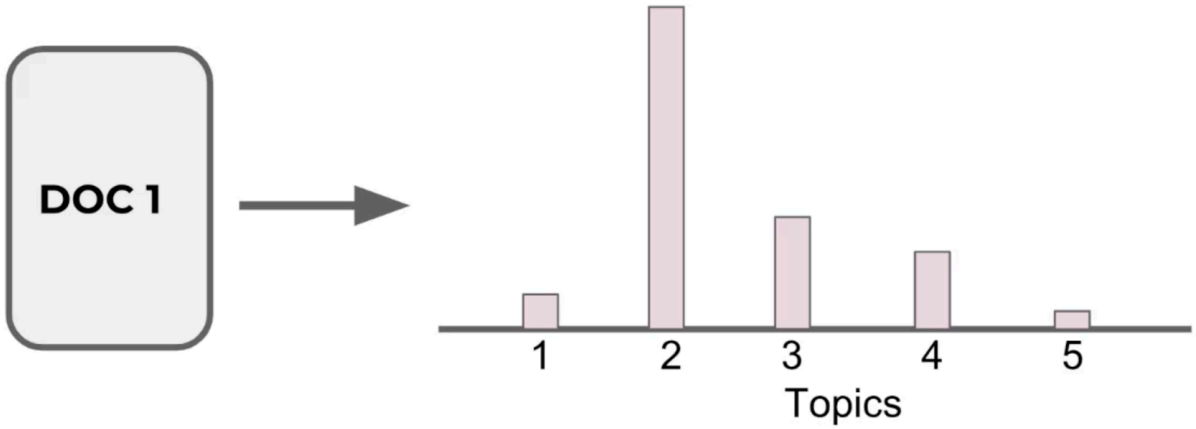
- Latent Dirichlet Allocation is based off this probability distribution.
- In 2003 LDA was first published as a graphical model for topic discovery in *Journal of Machine Learning Research* by David Blei, Andrew Ng and Michael I. Jordan.

- Let’s get a high level overview of how LDA works for topic modelling.
- I encourage you to also take a look at the original publication paper!

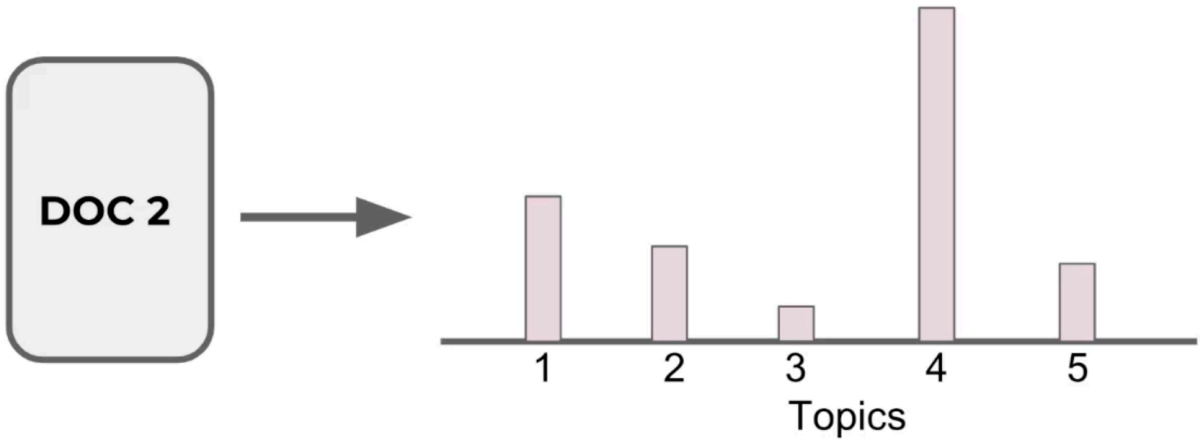
- Assumptions of LDA for Topic Modeling
  - Documents with similar topics use similar groups of words
  - Latent topics can then be found by searching for groups of words that frequently occur together in documents across the corpus.

- Assumptions of LDA for Topic Modeling
  - Documents are probability distributions over latent topics.
  - Topics themselves are probability distributions over words.

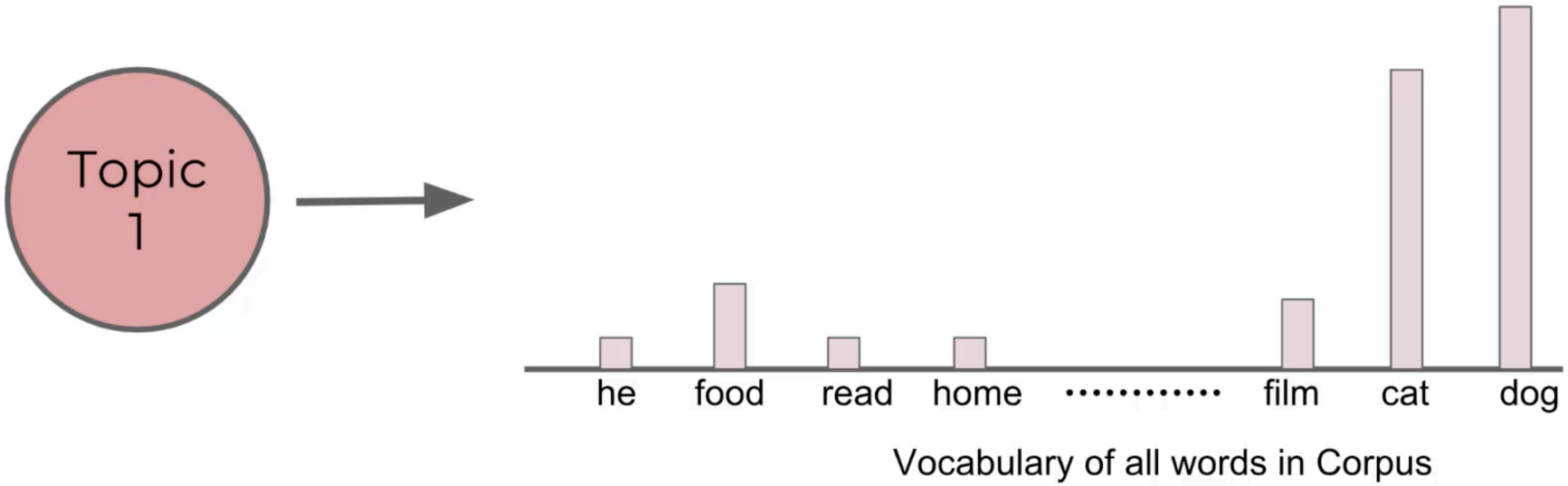
- Documents are probability distributions over latent topics.



- Documents are probability distributions over latent topics.



- Topics themselves are probability distributions over words.



- LDA represents documents as mixtures of topics that spit out words with certain probabilities.

- It assumes that documents are produced in the following fashion:
  - Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics).
  - e.g. 60% business, 20% politics, 10% food

- It assumes that documents are produced in the following fashion:
  - Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics).
  - e.g. 60% business, 20% politics, 10% food

- Generate each word in the document by:
  - First picking a topic according to the multinomial distribution that you sampled previously (60% business, 20% politics, 10% food)

- Generate each word in the document by:
  - 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 "apple" with 60% probability, "home" with 30% 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.

- Now imagine we have a set of documents.
- We’ve chosen some fixed number of K topics to discover, and want to use LDA to learn the topic representation of each document and the words associated to each topic.

- Go through each document, and randomly assign each word in the document to one of the K topics.
- This random assignment already gives you both topic representations of all the documents and word distributions of all the topics (note, these initial random topics won’t make sense).

- Now we iterate over every word in every document to improve these topics.
- For every word in every document and for each topic **t** we calculate:
  - p(topic **t** | document **d**) = the proportion of words in document **d** that are currently assigned to topic **t**

- Now we iterate over every word in every document to improve these topics.
- For every word in every document and for each topic **t** we calculate:
  - p(word **w** | 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(topic t | document d)** * **p(word w | topic t)**
- This is essentially the probability that topic t generated word w

- After repeating the previous step a large number of times, we eventually reach a roughly steady state where the assignments are acceptable.

- At the end we have each document assigned to a topic.
- We also can search for the words that have the highest probability of being assigned to a topic.

- We end up with an output such as:
  - Document assigned to Topic #4
  - Most common words (highest probability) for Topic #4:
    - ['cat','vet','birds','dog',...,'food','home']
  - It is up to the user to interpret these topics.

- Two important notes:
  - The user must decide on the amount of topics present in the document.
  - The user must interpret what the topics are.
- Let’s explore how to implement LDA with Python!