**Topic Modeling**

Topic model is one that automatically discovers topics occurring in a collection of documents

 sports. Some such words are athlete, soccer, and stadium.

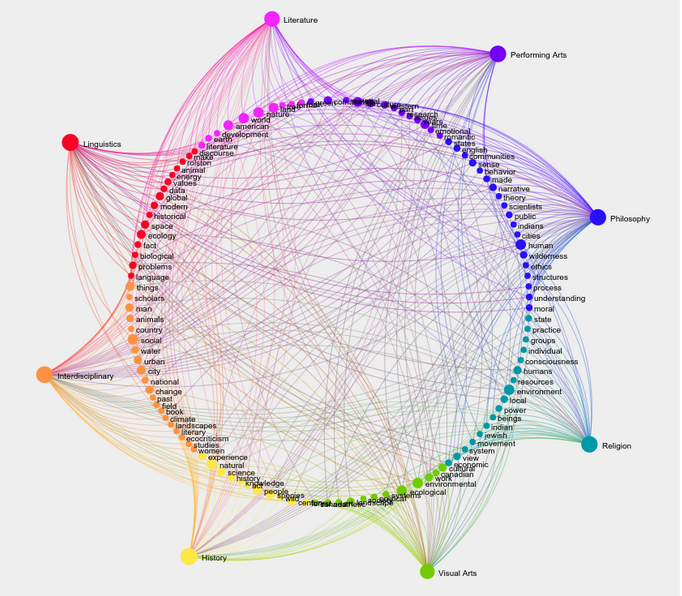
Topic models are used extensively for document clustering and organizing large collections of text data

* Topics contain a set of words
* Documents are made up of a set of topics

Use Cases

* Summarizing documents, tweets, etc., in the form of keywords based on learned topic distributions
* Detecting social media trends over a period of time
* Designing recommender systems for text

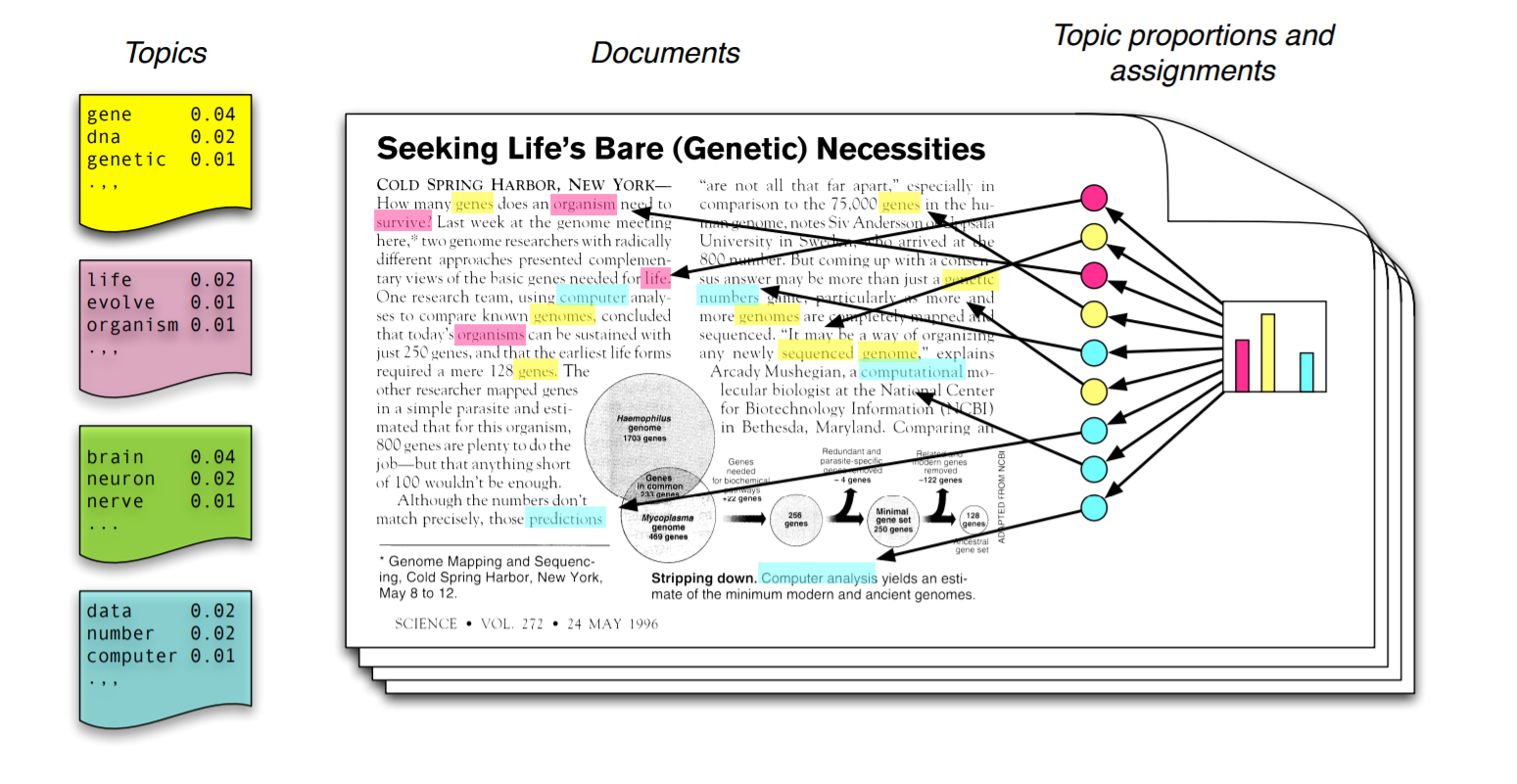
**Word cloud -**



**Latent Dirichlet Allocation** (**LDA**)

LDA is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

* Each document is modeled as a multinomial distribution of topics and each topic is modeled as a multinomial distribution of words.
* LDA assumes that the every chunk of text we feed into it will contain words that are somehow related. Therefore choosing the right corpus of data is crucial.
* It also assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution.
* Each document is a random mixture of corpus-wide topics
* Each word is drawn from one of those topics



* D1: I like to eat broccoli and bananas.
* D2: I ate a banana and salad for breakfast.



* D3: Puppies and kittens are cute.
* D4: My sister adopted a kitten yesterday.
* D5: Look at this cute hamster munching on a piece of broccoli.

**Output of LDA**

**Distribution of words across topics**

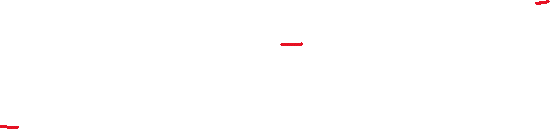
* Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching(Food)
* Topic B: 20% puppies, 20% kittens, 20% cute, 15% hamster(Pet animals)

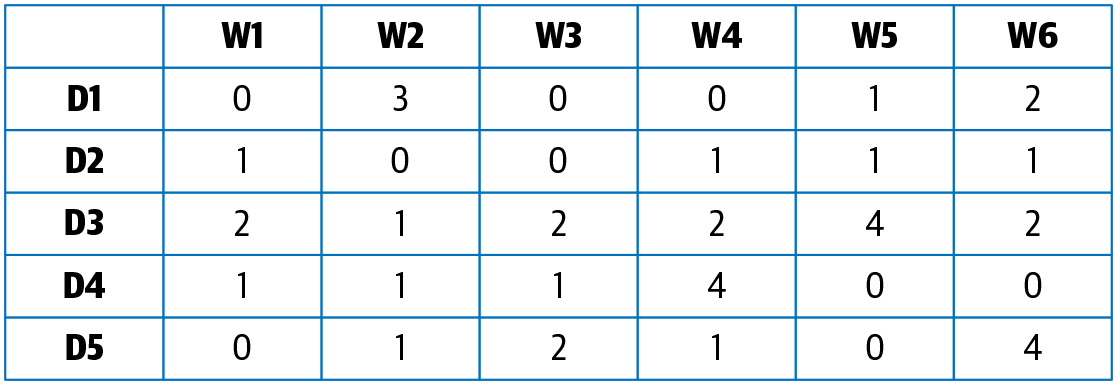
**Distribution of topics across documents**

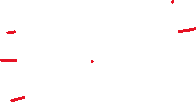
* Document 1 and 2: 100% Topic A- Food
* Document 3 and 4: 100% Topic B- pet animals
* Document 5: 60% Topic A, 40% Topic B- Food + Pet Animals

LDA tries to backtrack the generation process and figure out what topics would generate these documents in the first place. The topics are called “latent” because they’re hidden and must be discovered

LDA assumes that the documents under consideration are produced from a mixture of topics. It further assumes the following process generates these documents: at the start, we have a list of topics with a probability distribution. For every topic, there’s an associated list of words with a probability distribution. We sample k topics from topic distribution. For each of the k topics selected, we sample words from the corresponding distribution. This is how each document in the collection is generated.



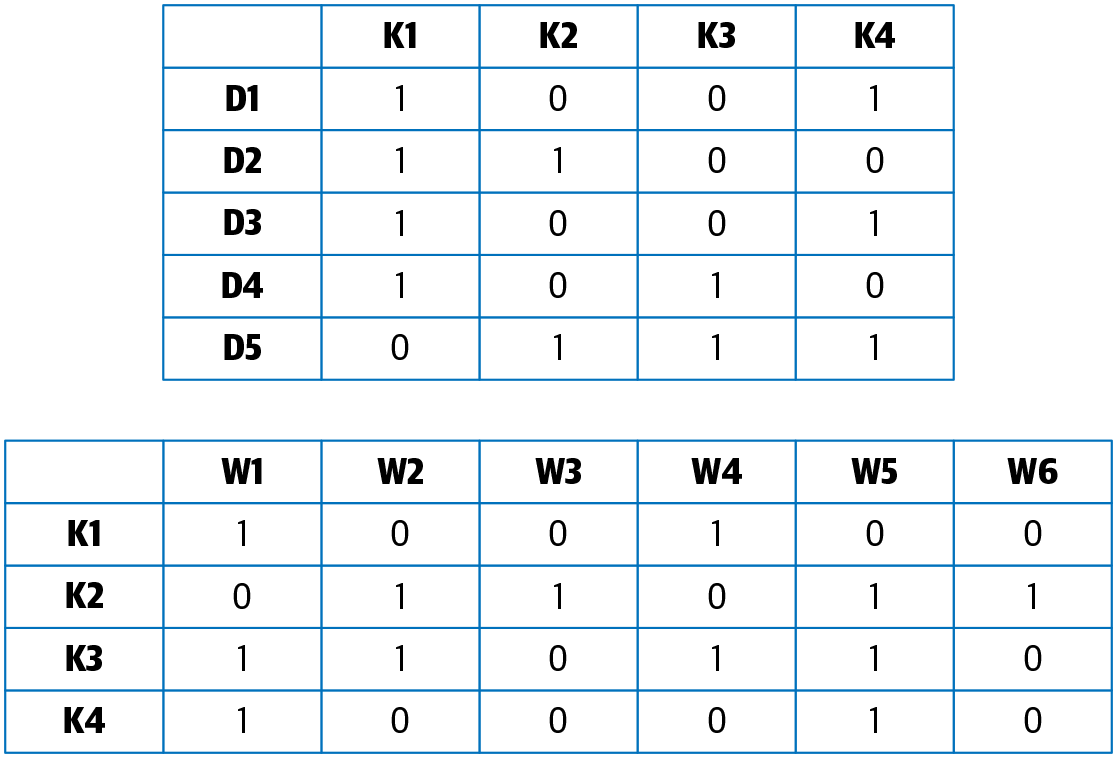




LDA factorizes M into two submatrices: M1 and M2. M1 is a **document–topics matrix** and M2 is a **topic–terms matrix**, with dimensions (M, K) and (K, N), respectively.

**K is the number of topics we are interested- K= 4**

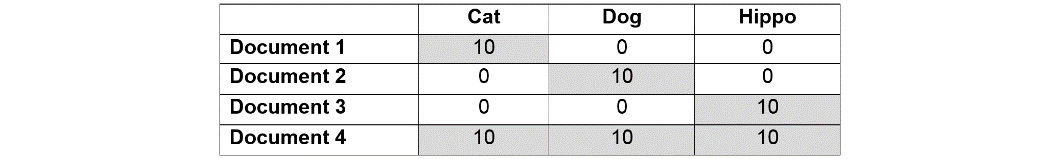






Evaluation

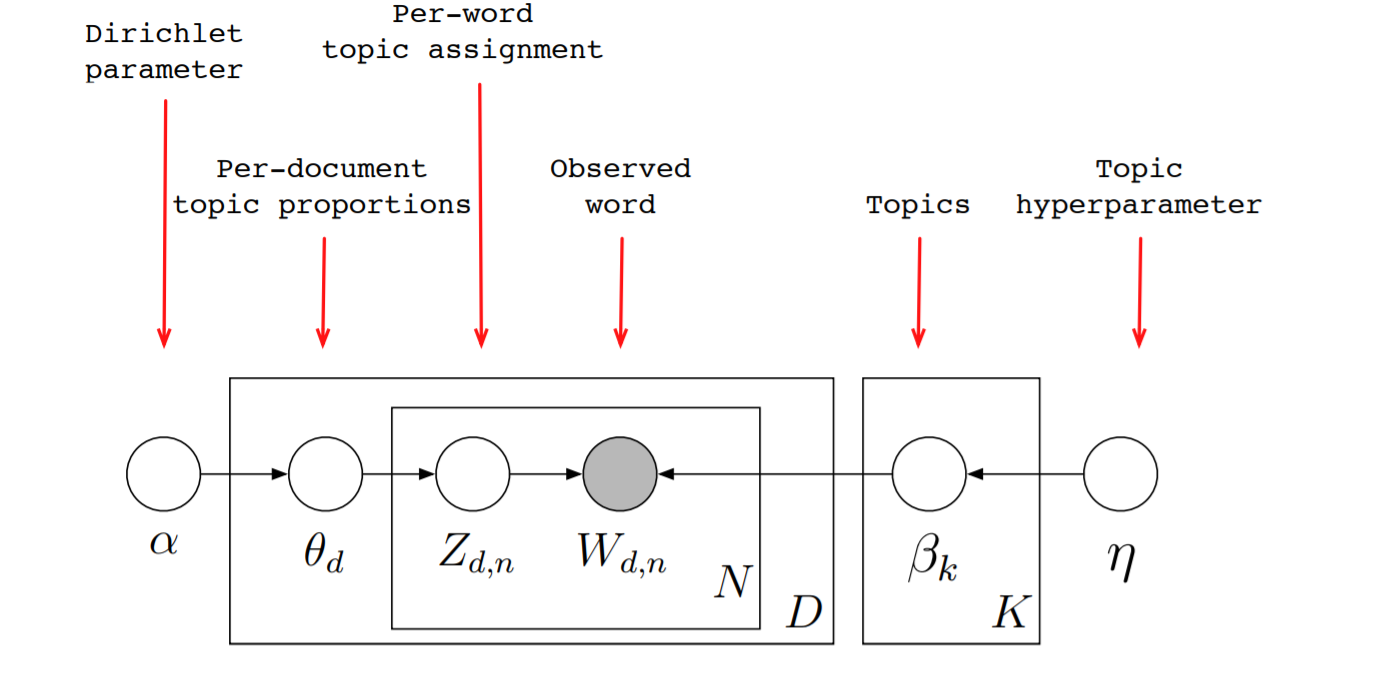
Topic–term matrix for LDA, we sort each topic from highest to lowest term weights and then select the first n terms for each topic. coherence for terms in each topic, which essentially measures how similar these words are to one another

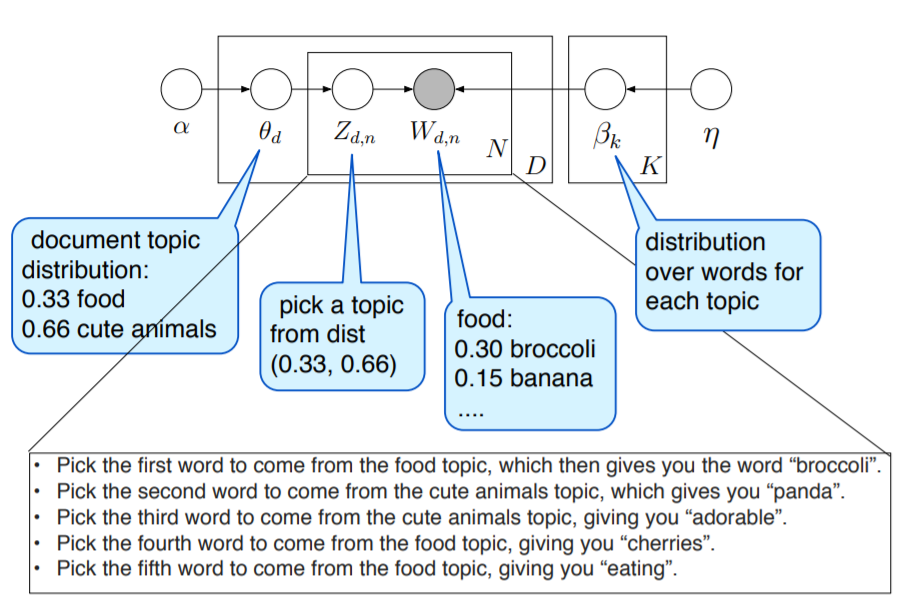


**Cat** is found **10** times in **Document 1** and **Document 4** and **0** times in documents **2** and **3**. **Document 4** contains all three words **10** times each. For its analysis, LDA maintains two probability tables. The first table tracks the probability of selecting a specific word when sampling a specific topic. The second table keeps track of the probability of selecting a specific topic when sampling a particular document:

Figure 5.10: Probability tables


sampled a word from **Topic 3**, it would likely be **Cat** (probability 99%). If you sampled **Document 4**, then there is a one-third chance of getting each of the topics, since it contains all three words in equal proportions





LDA

Introduction

Suppose you have the following set of sentences:

* I like to eat broccoli and bananas.
* I ate a banana and spinach smoothie for breakfast.
* Chinchillas and kittens are cute.
* My sister adopted a kitten yesterday.
* Look at this cute hamster munching on a piece of broccoli.

What is latent Dirichlet allocation? It’s a way of automatically discovering **topics** that these sentences contain. For example, given these sentences and asked for 2 topics, LDA might produce something like

* **Sentences 1 and 2**: 100% Topic A
* **Sentences 3 and 4**: 100% Topic B
* **Sentence 5**: 60% Topic A, 40% Topic B
* **Topic A**: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, … (at which point, you could interpret topic A to be about food)
* **Topic B**: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, … (at which point, you could interpret topic B to be about cute animals)

The question, of course, is: how does LDA perform this discovery?

LDA Model

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.

Example

Let’s make an example. According to the above process, when generating some particular document D, you might

* Pick 5 to be the number of words in D.
* Decide that D will be 1/2 about food and 1/2 about cute animals.
* Pick the first word to come from the food topic, which then gives you the word “broccoli”.
* Pick the second word to come from the cute animals topic, which gives you “panda”.
* Pick the third word to come from the cute animals topic, giving you “adorable”.
* Pick the fourth word to come from the food topic, giving you “cherries”.
* Pick the fifth word to come from the food topic, giving you “eating”.

So the document generated under the LDA model will be “broccoli panda adorable cherries eating” (note that LDA is a bag-of-words model).

Learning

So now suppose you have a set of documents. You’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. How do you do this? One way (known as collapsed Gibbs sampling) is the following:

* Go through each document, and randomly assign each word in the document to one of the K topics.
* Notice that this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).
* So to improve on them, for each document d…
  + Go through each word w in d…
    - And for each topic t, compute two things: 1) p(topic t | document d) = the proportion of words in document d that are currently assigned to topic t, and 2) 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) (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). (Also, I’m glossing over a couple of things here, in particular the use of priors/pseudocounts in these probabilities.)
    - 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, you’ll eventually reach a roughly steady state where your assignments are pretty good. So use these assignments 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).

Layman’s Explanation

In case the discussion above was a little eye-glazing, here’s another way to look at LDA in a different domain.

Suppose you’ve just moved to a new city. You’re a hipster and an anime fan, so you want to know where the other hipsters and anime geeks tend to hang out. Of course, as a hipster, you know you can’t just *ask*, so what do you do?

Here’s the scenario: you scope out a bunch of different establishments (**documents**) across town, making note of the people (**words**) hanging out in each of them (e.g., Alice hangs out at the mall and at the park, Bob hangs out at the movie theater and the park, and so on). Crucially, you don’t know the typical interest groups (**topics**) of each establishment, nor do you know the different interests of each person.

So you pick some number K of categories to learn (i.e., you want to learn the K most important kinds of categories people fall into), and start by making a guess as to why you see people where you do. For example, you initially guess that Alice is at the mall because people with interests in X like to hang out there; when you see her at the park, you guess it’s because her friends with interests in Y like to hang out there; when you see Bob at the movie theater, you randomly guess it’s because the Z people in this city really like to watch movies; and so on.

Of course, your random guesses are very likely to be incorrect (they’re random guesses, after all!), so you want to improve on them. One way of doing so is to:

* Pick a place and a person (e.g., Alice at the mall).
* Why is Alice likely to be at the mall? Probably because other people at the mall with the same interests sent her a message telling her to come.
* In other words, the more people with interests in X there are at the mall and the stronger Alice is associated with interest X (at all the other places she goes to), the more likely it is that Alice is at the mall because of interest X.
* So make a new guess as to why Alice is at the mall, choosing an interest with some probability according to how likely you think it is.

Go through each place and person over and over again. Your guesses keep getting better and better (after all, if you notice that lots of geeks hang out at the bookstore, and you suspect that Alice is pretty geeky herself, then it’s a good bet that Alice is at the bookstore because her geek friends told her to go there; and now that you have a better idea of why Alice is probably at the bookstore, you can use this knowledge in turn to improve your guesses as to why everyone else is where they are), and eventually you can stop updating. Then take a snapshot (or multiple snapshots) of your guesses, and use it to get all the information you want:

* For each category, you can count the people assigned to that category to figure out what people have this particular interest. By looking at the people themselves, you can interpret the category as well (e.g., if category X contains lots of tall people wearing jerseys and carrying around basketballs, you might interpret X as the “basketball players” group).
* For each place P and interest category C, you can compute the proportions of people at P because of C (under the current set of assignments), and these give you a representation of P. For example, you might learn that the people who hang out at Barnes & Noble consist of 10% hipsters, 50% anime fans, 10% jocks, and 30% college students.