LDA:

When you’re reading an article, review, or book, extracting meaning is the point. But when you must ‘read’ thousands of articles, reviews, or books extracting meaning becomes a bit of a challenge. It can be overwhelming to consider extracting and cataloging and tagging topics while reading – and considering that some words can overlap topics. Enter: Latent Dirichlet Allocation or LDA. The model takes in a dictionary of words – or all the words used across several documents or in a specific case for my most recent project, customer reviews.

This model is a mixture model – it considers that there are subpopulations in an overall population without requiring an observed dataset belong to the same subpopulation to which an individual observation belongs to. It makes statistical inferences about the properties of the sub

Placement occurs via multidimensional scaling onto a 2d plot.

In a recent project using NLP to segment product reviews, terms that were useful in predicting highest and lowest rated reviews like “awesome”, “loved”, “rubbish”, and “worst” weren’t informative in conveying exactly what the underlying content was. Once the obvious adjectives were removed and clustering was implemented to evaluate ‘topics’ that were being surfaced in the reviews. Clustering is a method of grouping data based on its relative numeric similarity. Trying to visualize what terms were in each of these clusters became challenging for two reasons:

certain terms could potentially belong to two clusters (example)

The visualization options are either barcharts or word clouds for term probabilities for each topic. Pie charts could theoretically be implemented based on word frequency and scatterplots could inform metadata. Actually viewing the terms in how they related to each other along with relevance and topic exclusivity the degree to which its occurrences are limited.

A classical clustering algorithm (like k-means or hierarchical clustering) gives you one label per document.

Topic modeling gives you a probabilistic composition of the document (so a document has a set of weighted labels). In addition, it gives you topics that are probability distributions over words.

Note that both procedures are unsupervised learning and far from being perfect, no matter how impressing the results may look at first sight. Apply them to dataset you understand well first

The tool by genism

<https://pyldavis.readthedocs.io/en/latest/modules/API.html>

1. the [distribution of the] number of words per topic is handled by **eta**
2. the [distribution of the] number of topics per document is handled by **alpha**
   1. Alpha-parameter of the model. Higher alpha priors for topics result in an even distribution of topics within a document. Low alpha priors ensure that the inference process distributes the probability mass on a few topics for each document.

Chuang et al. (2012b)

Saliency is weighted overall frequency

recommend saliency as a thresholding method for selecting which terms are included in the visualization, and they further use a seriation method for ordering the most salient terms to highlight

differences between topics.

Distinctiveness marginal distribution of topics

Choosing the number of topics is a bit of trial and error. By determining the perplexity of a held-out set of documents. The lower the perplexity, the better the fit.

<https://www.mathworks.com/help/textanalytics/ug/choose-number-of-topics-for-LDA-model.html>

<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#17howtofindtheoptimalnumberoftopicsforlda>

Taddy (2011) lift, defined as the ratio of a term’s probability within a topic to its marginal probability across the corpus. This generally decreases the rankings of globally frequent terms, which can be helpful. We find that it can be noisy, however, by giving high rankings to very rare terms that occur in only a single topic, for instance. While such terms may contain useful topical content, if they are very rare the topic may remain difficult to interpret.

Relevance – ranking terms within topics

Bigrams and tri grams

<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#9createbigramandtrigrammodels>

In [statistics](https://en.wikipedia.org/wiki/Statistics), a **mixture model** is a [probabilistic model](https://en.wikipedia.org/wiki/Probabilistic_model) for representing the presence of [subpopulations](https://en.wikipedia.org/wiki/Subpopulation) within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs. Formally a mixture model corresponds to the [mixture distribution](https://en.wikipedia.org/wiki/Mixture_distribution) that represents the [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution) of observations in the overall population. However, while problems associated with "mixture distributions" relate to deriving the properties of the overall population from those of the sub-populations, "mixture models" are used to make [statistical inferences](https://en.wikipedia.org/wiki/Statistical_inference) about the properties of the sub-populations given only observations on the pooled population, without sub-population identity information.

Some ways of implementing mixture models involve steps that attribute postulated sub-population-identities to individual observations (or weights towards such sub-populations), in which case these can be regarded as types of [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) or [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) procedures. However, not all inference procedures involve such steps.

<https://nbviewer.jupyter.org/github/bmabey/pyLDAvis/blob/master/notebooks/pyLDAvis_overview.ipynb>