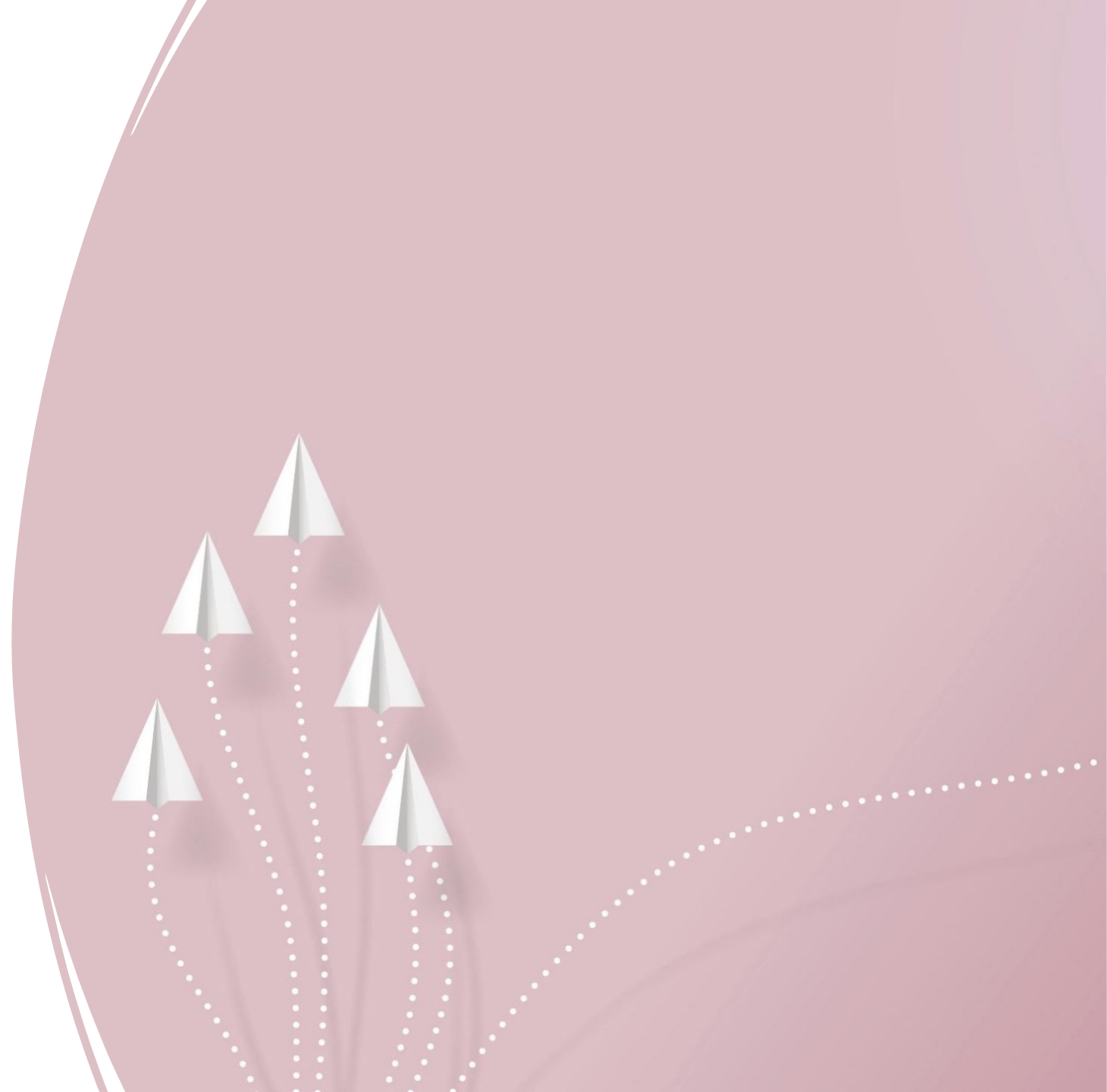


Topic Modeling

Modeling Topics across a corpus of documents using Latent Semantic Analysis and Latent Dirichlet Allocation

Peter Worth, Winter 2023



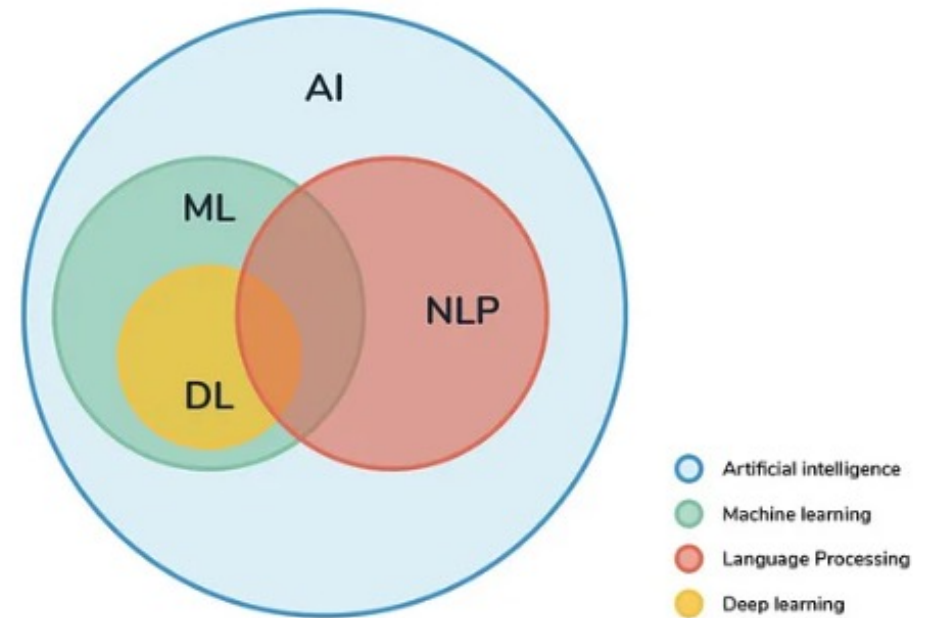
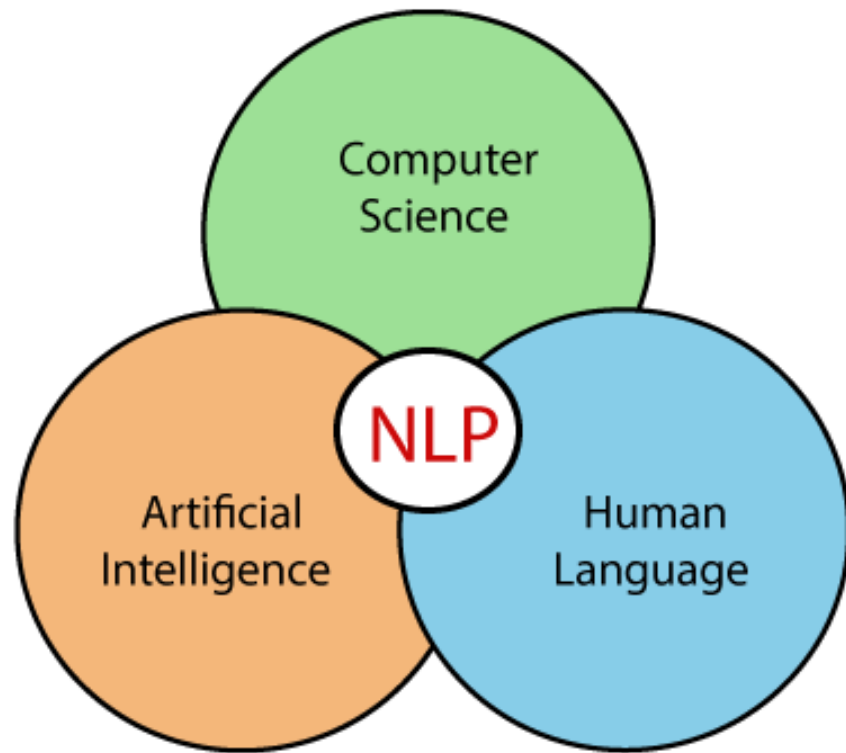
Contents

- ❑ Topic Modeling Overview
- ❑ Clustering Techniques (K-Means, Hierarchical and NMF)
- ❑ Latent Semantic Analysis (LSA or LSI)
- ❑ Probabilistic Latent Semantic Analysis (pLSA)
- ❑ Latent Dirichlet Allocation (LDA)

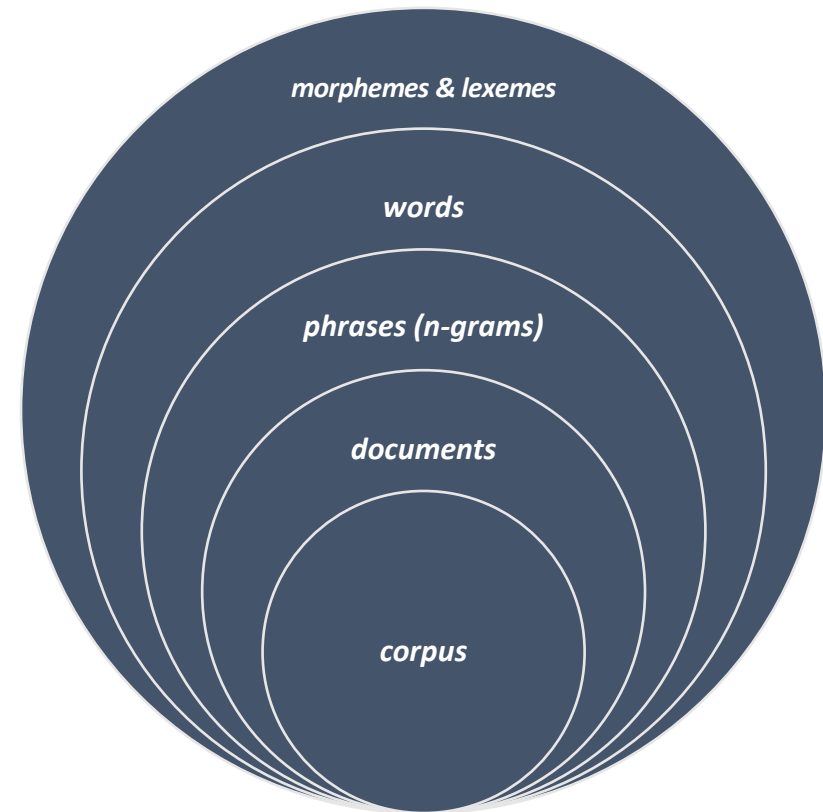
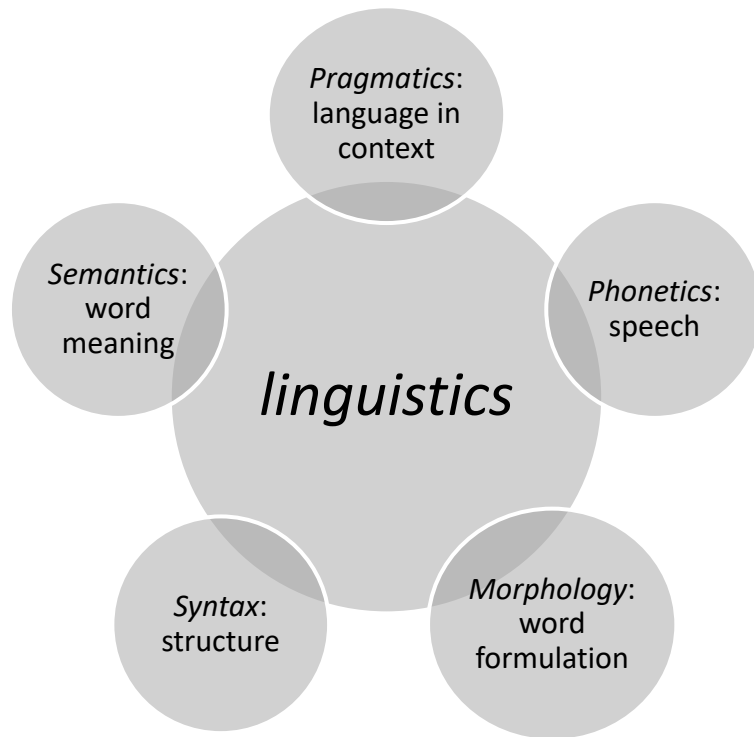
Contextualizing Topic Modeling within NLP

AI, ML and NLP

Machine Learning and NLP



Linguistics & NLP



NLP Applications

product suggestions

social media feeds

predictive text

speech recognition

facial recognition

auto-driving cars

chatbots, virtual assistants

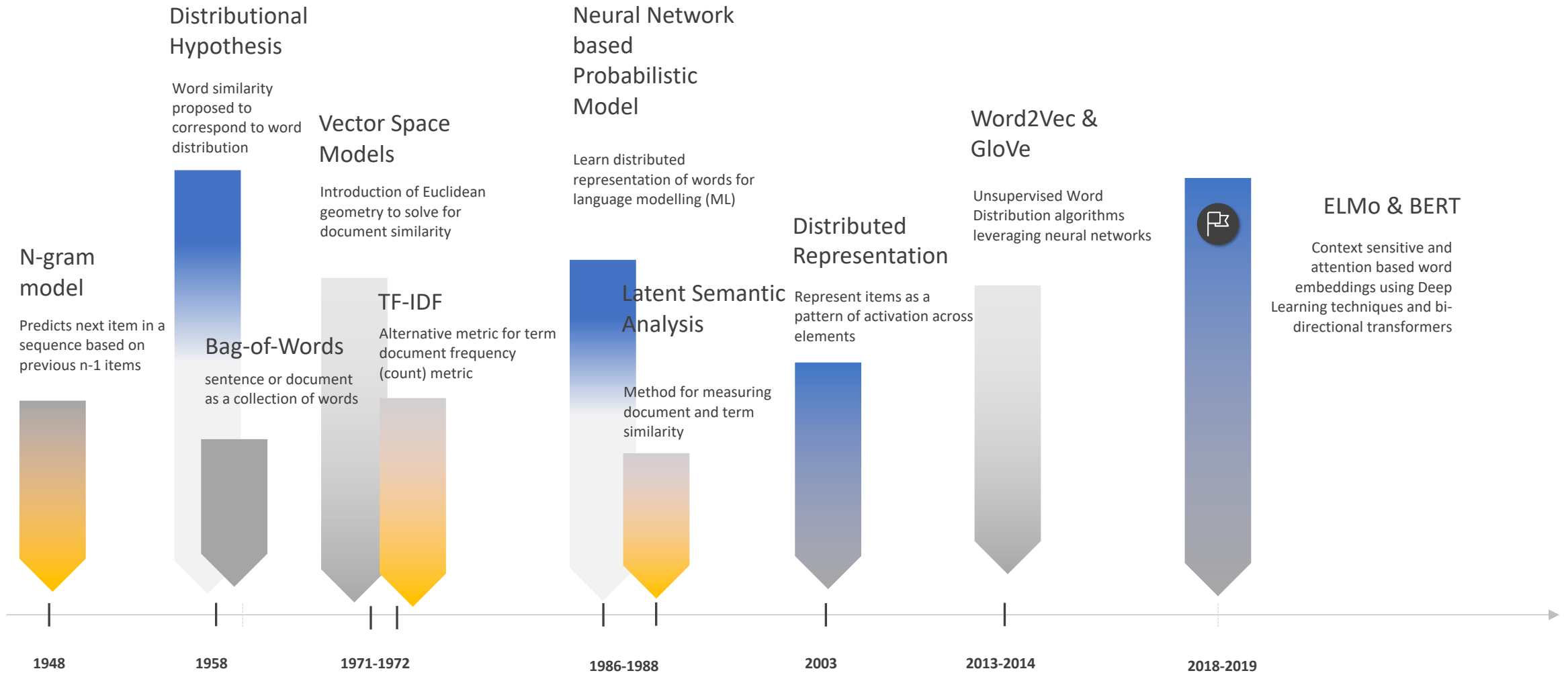
stock or weather forecasting

fraud detection

email spam filters

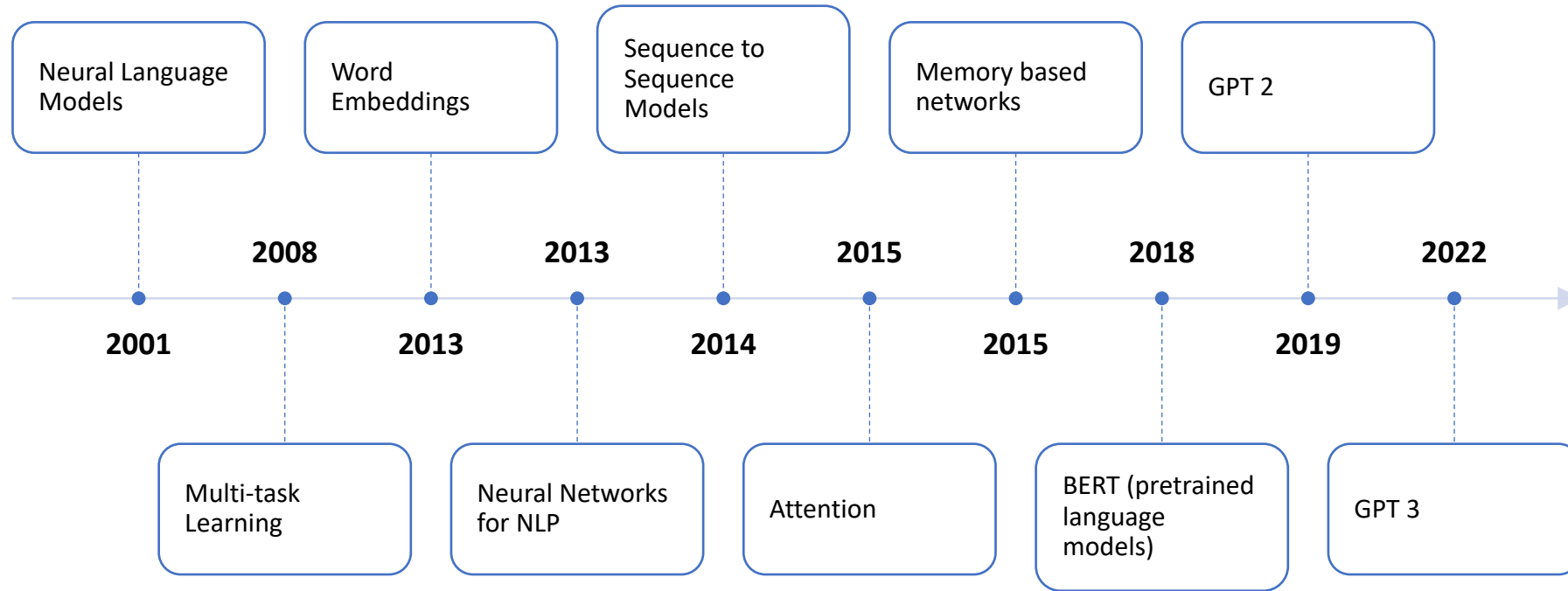
navigation technology (maps)

NLP Research & Development Timeline



Neural Networks and NLP History

Evolution of Natural Language Models



Notation and Terminology

- ❑ **Latent variables** aim to capture abstract notions such as topics
- ❑ **Word** is a basic unit of discrete data, defined as an item from a vocabulary indexed by $\{1, \dots, V\}$
Words is unit-basis vectors that have a single component equal to one and all other components equal to zero. v -th word in the vocabulary is represented by a V -vector w such that $w^v = 1$ and $w^u = 0$ for $u \neq v$.
- ❑ **Document** is a sequence of N words denoted by $\mathbf{w} = (w_1, w_2, \dots, w_N)$
- ❑ **Corpus** is a collection of M documents denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M\}$

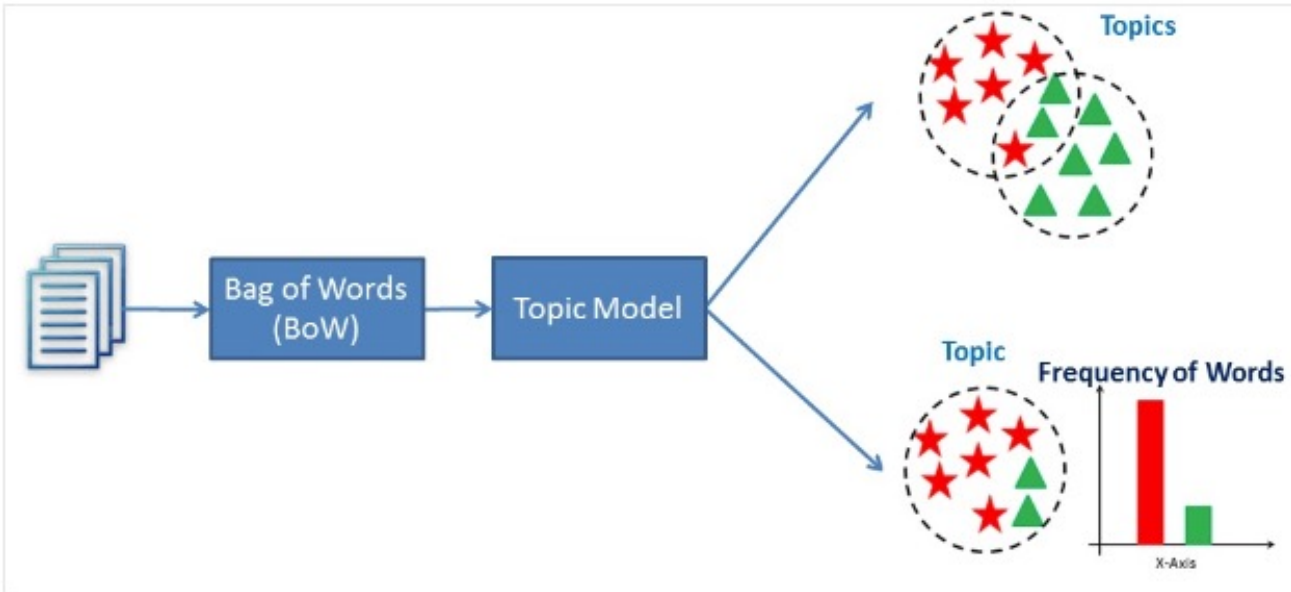
Text corpora modeling

- ☐ **Goal**
 - Finding short descriptions of the members of the collections (e.g. Finding topics from documents in corpora)
 - In particular, descriptions preserve essential statistical relationships
- ☐ **Application areas**
 - classification, novelty detection, summarization and collaborative filtering
- ☐ **Relevant approaches for text modeling**
 - *tf-idf* scheme, LSI, pLSI and latent Dirichlet allocation (LDA)

Topic Modeling

All topic models are based on the same basic assumption:

- each **document** consists of a mixture of **topics**, and
- each **topic** consists of a collection of **words**.



In other words, topic models are built around the idea that the semantics of our document are actually being governed by some hidden, or “latent,” variables that we are not observing.

As a result, the goal of topic modeling is to uncover these latent variables — *topics* — that shape the meaning of our document and corpus.

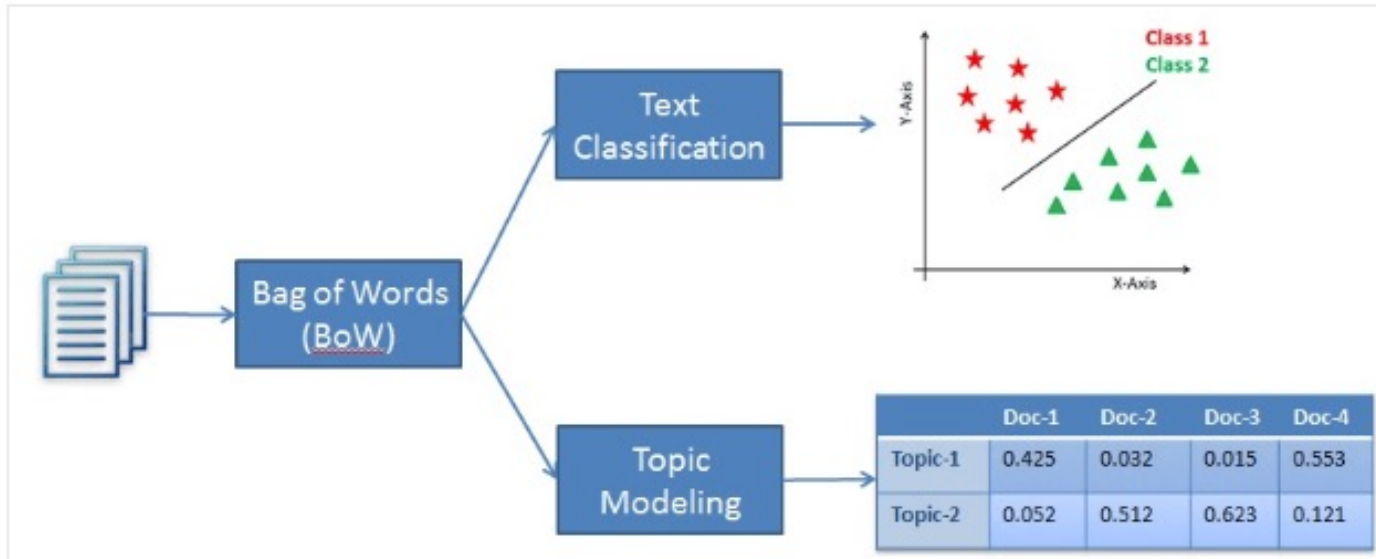
Topic Modeling automatically discover the hidden themes from given documents.

It is an unsupervised text analytics algorithm that is used for finding the group of words from the given document. These group of words represents a topic.

There is a possibility that, a single document can associate with multiple themes. for example, a group words such as 'patient', 'doctor', 'disease', 'cancer', ad 'health' will represents topic 'healthcare'.

Topic Modeling is a different game compared to rule-based text searching that uses regular expressions.

Text Classification vs Topic Modeling



Text classification is a supervised machine learning problem, where a text document or article is classified into a pre-defined set of classes.

Topic modeling is the process of discovering groups of co-occurring words in text documents. These groups of co-occurring related words make "topics". It is a form of unsupervised learning, so the set of possible topics are unknown.

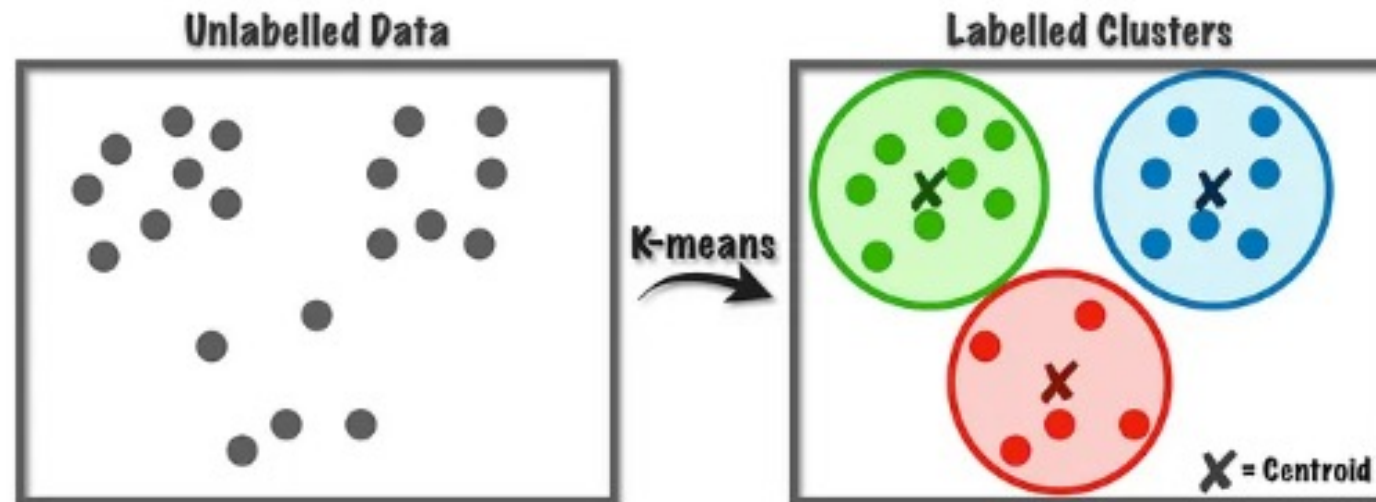
Topic modeling can be used to solve the text classification problem. Topic modeling will identify the topics present in a document, while text classification classifies the text into a single class.

Text Clustering Techniques

K-Means, Hierarchical and NMF Clustering

K-Means Clustering

- K-means is an unsupervised clustering algorithm designed to partition unlabelled data into a certain number (that's the " K ") of distinct groupings. In other words, k-means finds observations that share important characteristics and classifies them together into clusters. A good clustering solution is one that finds clusters such that the observations within each cluster are more similar than the clusters themselves.



K-Means Algorithm

We randomly takes position for K clusters and then find point nearest to it then we take mean of a cluster and then again changes the center and continues until converges.

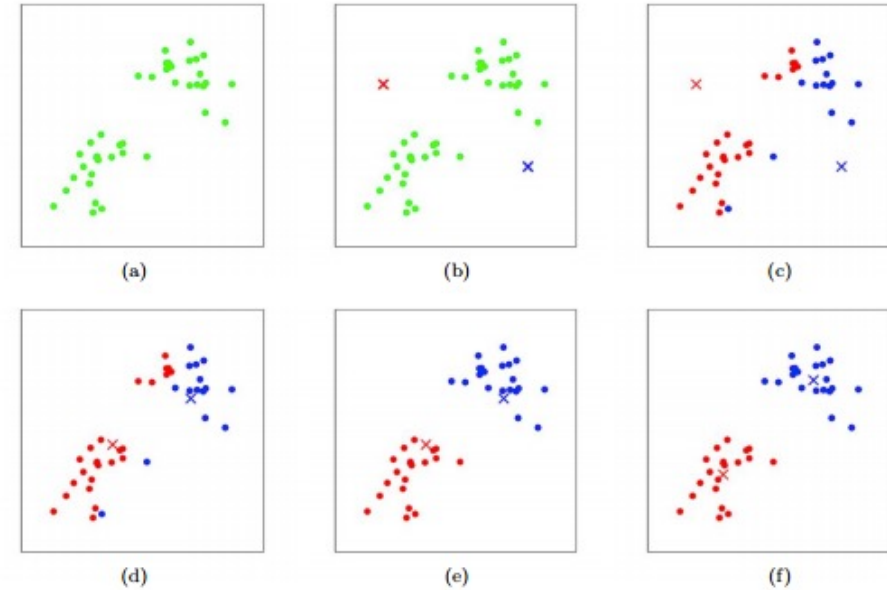


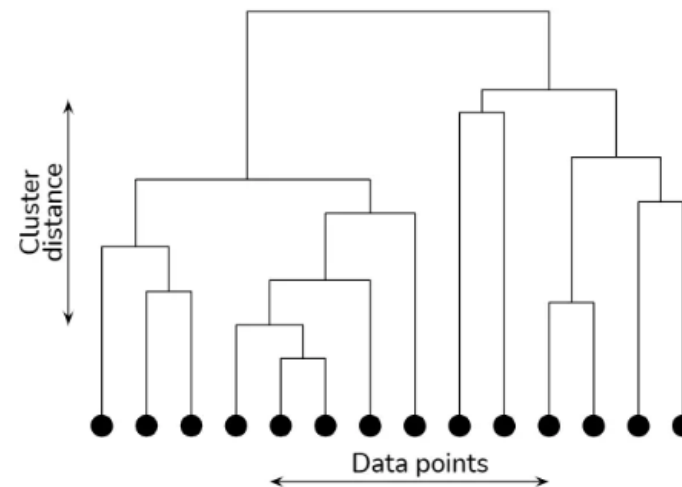
Figure 1: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster centroid (shown by "painting" the training examples the same color as the cluster centroid to which is assigned); then we move each cluster centroid to the mean of the points assigned to it. Images courtesy of Michael Jordan.

Hierarchical Clustering

- *Hierarchical clustering is a type of clustering algorithm used in unsupervised machine learning, which groups similar data points together based on their distance or similarity. It involves organizing data into a tree-like structure, known as a dendrogram, which represents the relationships between the data points.*

Dendrogram

The sole concept of hierarchical clustering lies in just the construction and analysis of a dendrogram. A dendrogram is a tree-like structure that explains the relationship between all the data points in the system.

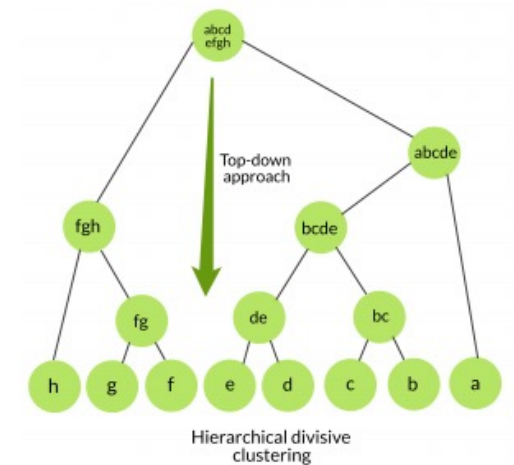
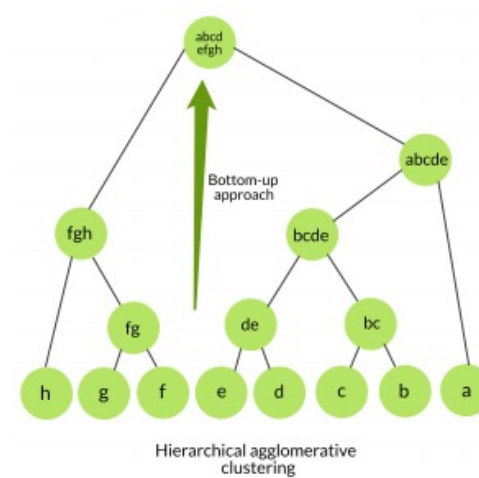


Dendrogram with data points on the x-axis and cluster distance on the y-axis (Image by Author)

Type of Hierarchical Clustering

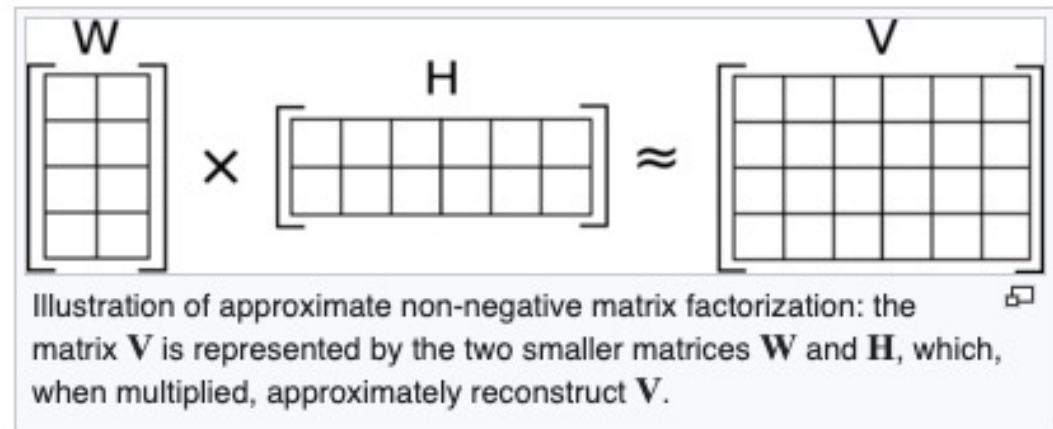
To measure the similarity or distance between clusters, various metrics can be used, such as Euclidean distance, Manhattan distance, or cosine similarity. Additionally, different linkage methods, like single, complete, average, and Ward's linkage, help determine how clusters are merged or split.

1. **Agglomerative clustering:** In this approach, the algorithm starts with each data point as a separate cluster and iteratively merges the closest pair of clusters until all data points belong to a single cluster.
2. **Divisive clustering:** In contrast to agglomerative clustering, divisive clustering begins with a single cluster containing all data points and recursively splits the clusters into smaller ones until each data point is in its own cluster.




Non-negative matrix factorization (NMF)

Non-negative matrix factorization (NMF or NNMF), also **non-negative matrix approximation**^{[1][2]} is a group of **algorithms** in **multivariate analysis** and **linear algebra** where a **matrix V** is **factorized** into (usually) two matrices **W** and **H** , with the property that all three matrices have no negative elements. This non-negativity makes the resulting matrices easier to inspect. Also, in applications such as processing of audio spectrograms or muscular activity, non-negativity is inherent to the data being considered. Since the problem is not exactly solvable in general, it is commonly approximated numerically.



Latent Semantic Analysis (LSA)



Discovering latent topics in a corpus of documents

Latent Semantic Analysis and TF-IDF

The diagram shows the formula $w_{i,j} = t f_{i,j} \times \log \frac{N}{df_j}$ with arrows pointing to its components and their meanings:

- $w_{i,j}$ is labeled "tf-idf score" (red text).
- $t f_{i,j}$ is labeled "# occurrences of term in document" (green text).
- N is labeled "# total documents" (blue text).
- df_j is labeled "# documents containing word" (purple text).

- Latent Semantic Analysis, or LSA, is one of the foundational techniques in topic modeling. The core idea is to take a matrix of what we have — documents and terms — and decompose it into a separate document-topic matrix and a topic-term matrix.
- The first step is generating our document-term matrix. Given m documents and n words in our vocabulary, we can construct an $m \times n$ matrix A in which each row represents a document and each column represents a word.
- In the simplest version of LSA, each entry can simply be a raw count of the number of times the j -th word appeared in the i -th document. In practice, however, raw counts do not work particularly well because they do not account for the *significance* of each word in the document. [For example, the word “nuclear” probably informs us more about the topic(s) of a given document than the word “test.”]
- Consequently, LSA models typically replace raw counts in the document-term matrix with a **tf-idf score**. Tf-idf, or term frequency-inverse document frequency, assigns a weight for term j in document i as follows:

Details of tf-idf

- Term Frequency - Inverse Document Frequency: $\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$
- Term Frequency $\text{tf}(t, d)$: How frequently a word occurs in a document
- Inverse Document Frequency $\text{idf}(t, D)$: The inverse document frequency for any given term

$$\text{tf}(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)}$$

$$\text{idf}(t, D) = \ln \left(\frac{|D|}{|\{d \in D : t \in d\}|} \right)$$

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

$f_d(t)$:= frequency of term t in document d

D := corpus of documents

High tf-idf scored words

-> occur frequently

-> provide more important information

Details of LSI (also called LSA)

- ❑ Analysis of latent semantics in a corpora of text (by using Singular Value Decomposition)
- ❑ A collection of documents can be represented as a term-document matrix
- ❑ Similarity of doc-doc, term-term, term-doc can be measured by cosine similarity $\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$
- ❑ Not easy to find polysemy and synonymy
- ❑ LSI transforms the original data in a different space so that two documents/words about the same concept are mapped close

$$C = U \Sigma V^T$$

documents

C

transformed word-document co-occurrence matrix

words

=

dimensions

U

word space

words

dimensions

Σ

weights

dimensions

dimensions

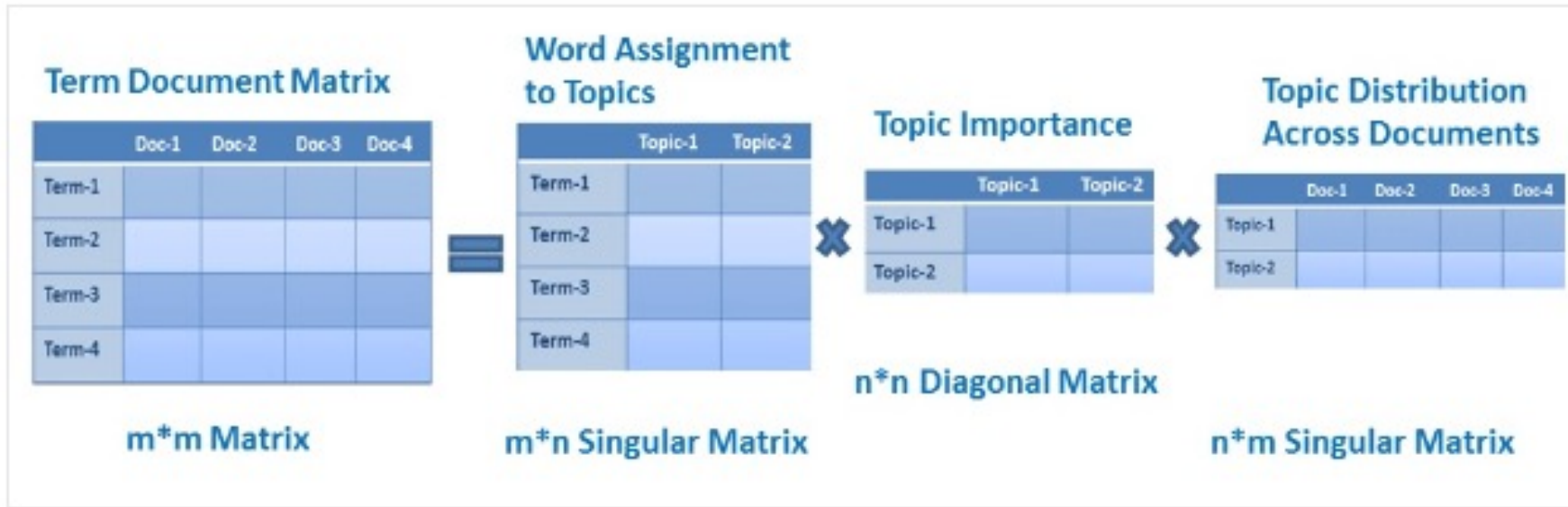
V^T

document space

documents

GRIFFITHS, STEYVERS, AND TENENBAUM

Latent Semantic Analysis



LSA (Latent Semantic Analysis) also known as LSI (Latent Semantic Index) LSA uses bag of word(BoW) model, which results in a term-document matrix(occurrence of terms in a document). Rows represent terms and columns represent documents.

LSA learns latent topics by performing a matrix decomposition on the document-term matrix using Singular value decomposition.

LSA is typically used as a dimension reduction or noise reducing technique.

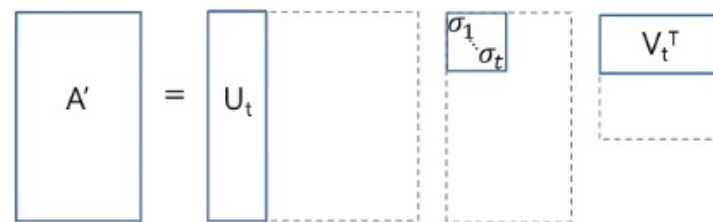
Singular Value Decomposition (SVD)

- This dimensionality reduction can be performed using **truncated SVD**. SVD, or singular value decomposition, is a technique in linear algebra that factorizes any matrix M into the product of 3 separate matrices: $M=U*S*V$, where S is a diagonal matrix of the [singular values](#) of M .

- Critically, truncated SVD reduces dimensionality by selecting only the t largest singular values, and only keeping the first t columns of U and V . In this case, t is a hyperparameter we can select and adjust to reflect the number of topics we want to find

$$A \approx U_t S_t V_t^T$$

Intuitively, think of this as only keeping the t most significant dimensions in our transformed space.



In this case, $U \in \mathbb{R}^{(m \times t)}$ emerges as our document-topic matrix, and $V \in \mathbb{R}^{(n \times t)}$ becomes our term-topic matrix. In both U and V , the columns correspond to one of our t topics. In U , rows represent document vectors expressed in terms of topics; in V , rows represent term vectors expressed in terms of topics.

Singular Value Decomposition (SVD)

- With these document vectors and term vectors, we can now easily apply measures such as cosine similarity to evaluate:
- the similarity of different documents
- the similarity of different words
- the similarity of terms (or “queries”) and documents (which becomes useful in information retrieval, when we want to retrieve passages most relevant to our search query).

Code

In sklearn, a simple implementation of LSA might look something like this:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import Pipeline

documents = ["doc1.txt", "doc2.txt", "doc3.txt"]

# raw documents to tf-idf matrix:
vectorizer = TfidfVectorizer(stop_words='english',
                             use_idf=True,
                             smooth_idf=True)

# SVD to reduce dimensionality:
svd_model = TruncatedSVD(n_components=100,          // num dimensions
                         algorithm='randomized',
                         n_iter=10)

# pipeline of tf-idf + SVD, fit to and applied to documents:
svd_transformer = Pipeline([('tfidf', vectorizer),
                             ('svd', svd_model)])

svd_matrix = svd_transformer.fit_transform(documents)

# svd_matrix can later be used to compare documents, compare words,
# or compare queries with documents
```

Determining Optimum # of Topics

- Consider each topic as a cluster and find out the effectiveness of a cluster using the Silhouette coefficient
- Topic coherence measure is a realistic measure for identifying the number of topics.
- **Topic Coherence** measure is a widely used metric to evaluate topic models. It uses the latent variable models. Each generated topic has a list of words.
- In topic coherence measure, you will find average/median of pairwise word similarity scores of the words in a topic. The high value of topic coherence score model will be considered as a good topic model.

Probabilistic Latent Semantic Analysis (pLSA)

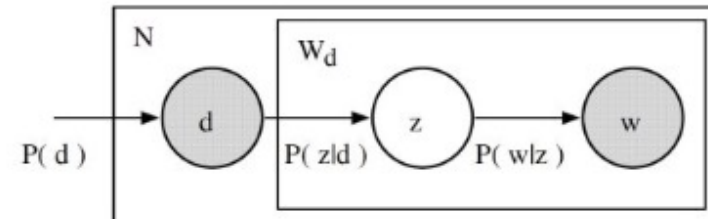
Using probabilities to determine the document-term matrix

pLSA

- pLSA, or Probabilistic Latent Semantic Analysis, uses a probabilistic method instead of SVD to tackle the problem.
- The core idea is to find a probabilistic model with latent topics that can *generate* the data we observe in our document-term matrix.
- In particular, we want a model $P(D,W)$ such that for any document d and word w , $P(d,w)$ corresponds to that entry in the document-term matrix.

Recall the basic assumption of topic models: each document consists of a mixture of topics, and each topic consists of a collection of words. pLSA adds a probabilistic spin to these assumptions:

- given a document d , topic z is present in that document with probability $P(z|d)$
- given a topic z , word w is drawn from z with probability $P(w|z)$



Details of pLSI

$$\begin{aligned} P(d, w) &= P(d)P(w|d), \text{ where} \\ P(w|d) &= \sum_{z \in \mathcal{Z}} P(w|z)P(z|d) . \\ P(d, w) &= \sum_{z \in \mathcal{Z}} P(z)P(w|z)P(d|z) . \end{aligned}$$

- ❑ pLSI models each word in a document as a *sample* from mixture model
- ❑ The mixture components of mixture model are multinomial random variables that can be viewed as representations of ‘topics’. Thus each word is generated from a *single topic*
- ❑ Each document is represented as a list of mixing proportions for these mixture components “topics”.

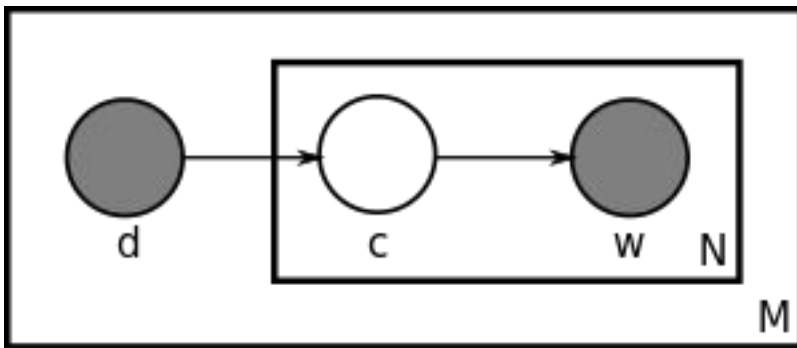


Plate notation representing the pLSA model

d : the document index variable

c : a word's topic drawn from the document's topic distribution $P(c|d)$

w : a word drawn from the word distribution of this word's topic $P(w|c)$

d and **w** : observable variables

topic c : a latent variable (represented as z in the above)

Two assumptions

- ❑ **Bag-of-words** (Fundamental probabilistic assumption for dimensionality reduction)

The order of words in document can be neglected.

This is an assumption of exchangeability for the words in a document.

- ❑ **Exchangeability** (Documents are exchangeable)

The specific ordering of the documents in a corpus can be neglected

- ❑ Exchangeability is not equivalent to an assumption that the random variables are independent and identically distributed, it is rather “*Conditionally independent and identically distributed*”. This condition is with respect to an underlying latent parameter of a probability distribution.

Latent Dirichlet Allocation

□ **Generative probabilistic model of a corpus**

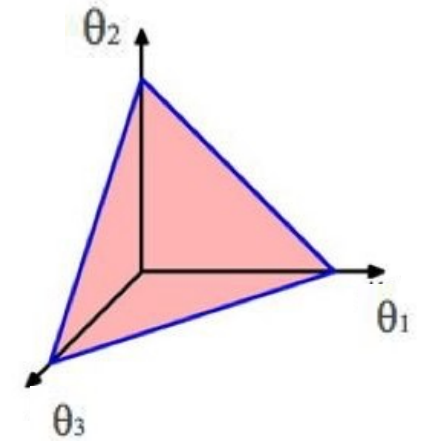
The basic idea is that documents are represented as random mixtures over latent topics
Each topic is characterized by a distribution over words.

□ **LDA generative process for each document w in a corpus D :**

1. Choose $N \sim \text{Poisson}(\xi)$
2. Choose $\theta \sim \text{Dir}(\alpha)$
3. For each of the N words w_n :
 - (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
 - (b) Choose a word w_n from $p(w_n | z_n, \beta)$, a multinomial probability conditioned on the topic z_n .

□ **Several simplifying assumption :** **[1]** The dimensionality k of the Dirichlet distribution (and thus the dimensionality of topic variable z) is assumed to be known and fixed **[2]** The word probabilities are parameterized by $k \times V$ matrix β where $\beta_{ij} = p(w^j = 1 | z^i = 1)$ which for now we treat as a fixed quantity to be estimated. **[3]** Poisson assumption is not critical to anything. **[4]** N is independent to all other data generating θ and z

LDA (continue)



- k -dimensional Dirichlet random variable θ can take values in $(k-1)$ -simplex

(k vector θ lies in the $k-1$ simplex if $\theta_i \geq 0, \sum_{i=1}^k \theta_i = 1$)

- The probability density on this simplex : $p(\theta | \alpha) = \frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)} \theta_1^{\alpha_1-1} \dots \theta_k^{\alpha_k-1}$
- Parameter α is a k -vector with components with $\alpha_i > 0$, and where $\Gamma(x)$ is gamma function.
- Given the parameters α and β , the joint distribution of a topic mixture θ , a set of N topics z , and a set of N words w is given by:

$$p(\theta, z, w | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta)$$

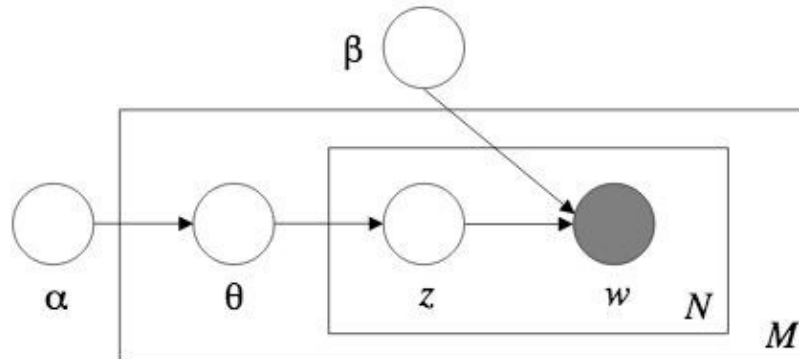
LDA (continue)

- Where $p(z_n | \theta)$ is simply θ_i for the unique i such that $z_n^i = 1$
Integrating over θ and summing over z , we obtain the marginal distribution of a document

$$p(\mathbf{w} | \alpha, \beta) = \int p(\theta | \alpha) \left(\prod_{n=1}^N \sum_{z_n} p(z_n | \theta) p(w_n | z_n, \beta) \right) d\theta.$$

- Probability of a corpus by taking the product of the marginal probabilities of single documents

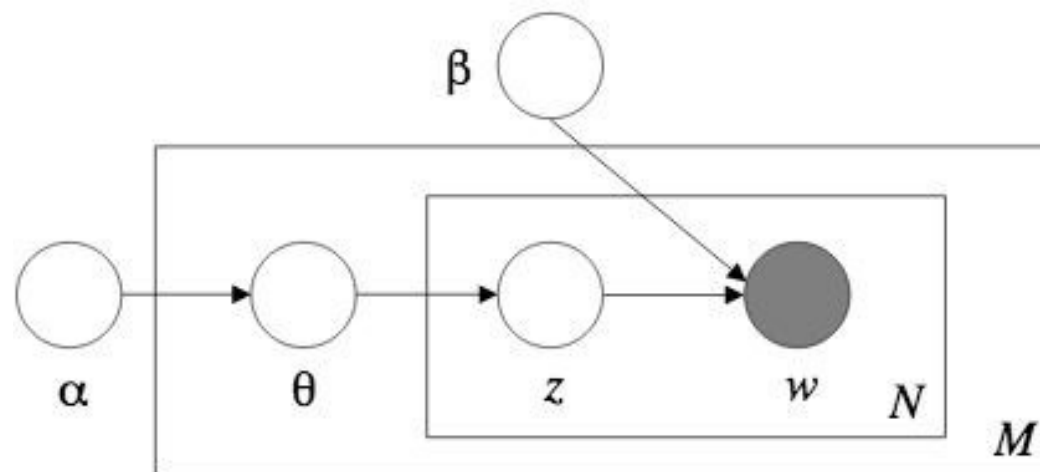
$$p(D | \alpha, \beta) = \prod_{d=1}^M \int p(\theta_d | \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} | \theta_d) p(w_{dn} | z_{dn}, \beta) \right) d\theta_d.$$



Graphical Model Representation of LDA

Three levels to the LDA representation

1. The parameter α and β are corpus level parameters, assumed to be sampled once in the process of generating a corpus
2. Variable θ_d are document-level variables, sampled per document
3. Variables z_{dn} and w_{dn} are word-level variables and sampled once for each word in each document



Computing Joint Probability

Intuitively, the right-hand side of this equation is telling us *how likely it is see some document*, and then based upon the distribution of topics of that document, *how likely it is to find a certain word within that document*.

In this case, $P(D)$, $P(Z|D)$, and $P(W|Z)$ are the parameters of our model. $P(D)$ can be determined directly from our corpus. $P(Z|D)$ and $P(W|Z)$ are modeled as multinomial distributions, and can be trained using the expectation-maximization algorithm (EM). Without going into a full mathematical treatment of the algorithm, EM is a method of finding the likeliest parameter estimates for a model which depends on unobserved, latent variables (in our case, the topics).

Formally, the joint probability of seeing a given document and word together is:

$$P(D, W) = P(D) \sum_Z P(Z|D)P(W|Z)$$

Intuitively, the right-hand side of this equation is telling us *how likely it is see some document*, and then based upon the distribution of topics of that document, *how likely it is to find a certain word within that document*.

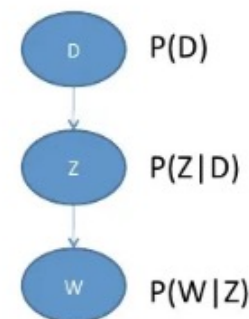
pLSA vs. LSA

Interestingly, $P(D,W)$ can be equivalently parameterized using a different set of 3 parameters:

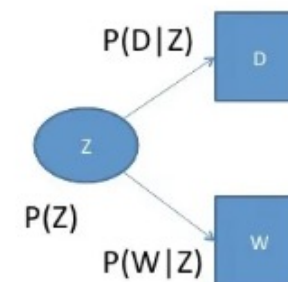
$$P(D,W) = \sum_Z P(Z)P(D|Z)P(W|Z)$$

We can understand this equivalency by looking at the model as a generative process. In our first parameterization, we were starting with the document with $P(d)$, and then generating the topic with $P(z|d)$, and then generating the word with $P(w|z)$. In *this* parameterization, we are starting with the topic with $P(z)$, and then independently generating the document with $P(d|z)$ and the word with $P(w|z)$.

- Start with document



- Start with topic



The reason this new parameterization is so interesting is because we can see a direct parallel between our pLSA model and our LSA model:

$$P(D,W) = \sum_Z \underbrace{P(Z)}_{\text{diagonal}} \underbrace{P(D|Z)}_{\text{document-topic}} \underbrace{P(W|Z)}_{\text{term-topic}}$$
$$A \approx \underbrace{U}_U \underbrace{S}_S \underbrace{V^T}_V$$

where the probability of our topic $P(Z)$ corresponds to the diagonal matrix of our singular topic probabilities, the probability of our document given the topic $P(D|Z)$ corresponds to our document-topic matrix U , and the probability of our word given the topic $P(W|Z)$ corresponds to our term-topic matrix V .

pLSA Summary

- So what does that tell us? Although it looks quite different and approaches the problem in a very different way, pLSA really just adds a probabilistic treatment of topics and words on top of LSA. It is a far more flexible model, but still has a few problems. In particular:
- Because we have no parameters to model $P(D)$, we don't know how to assign probabilities to new documents
- The number of parameters for pLSA grows linearly with the number of documents we have, so it is prone to overfitting
- We will not look at any code for pLSA because it is rarely used on its own. In general, when people are looking for a topic model beyond the baseline performance LSA gives, they turn to LDA. LDA, the most common type of topic model, extends PLSA to address these issues.

Latent Dirichlet Allocation (LDA)

The Bayesian version of pLSA

LDA Explained

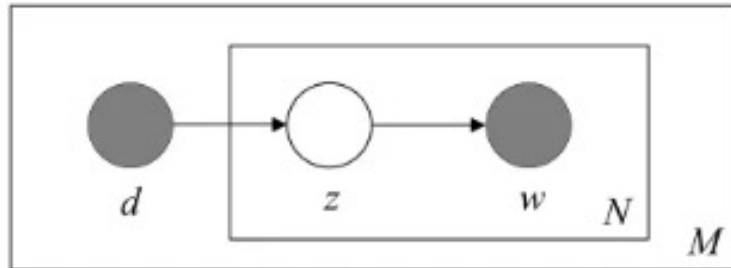
Consider the very relevant example of comparing probability distributions of topic mixtures. Let's say the corpus we are looking at has documents from 3 very different subject areas. If we want to model this, the *type* of distribution we want will be one that very heavily weights one specific topic, and doesn't give much weight to the rest at all. If we have 3 topics, then some specific *probability distributions* we'd likely see are:

- **Mixture X:** 90% topic A, 5% topic B, 5% topic C
- **Mixture Y:** 5% topic A, 90% topic B, 5% topic C
- **Mixture Z:** 5% topic A, 5% topic B, 90% topic C

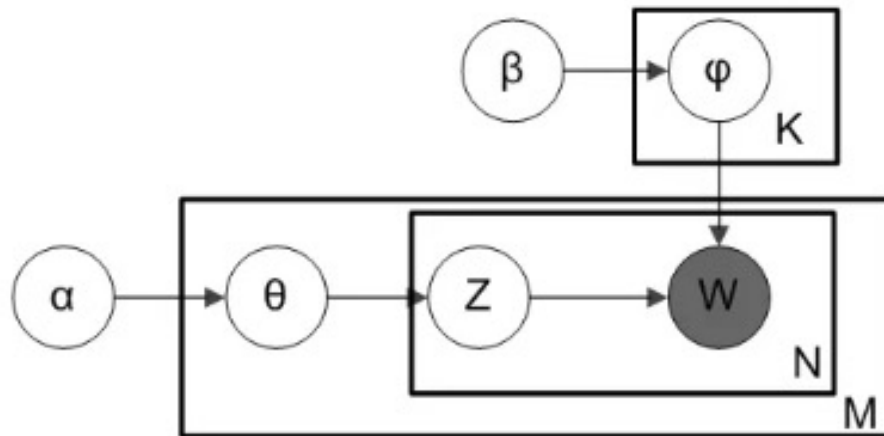
- If we draw a random probability distribution from this dirichlet distribution, parameterized by large weights on a single topic, we would likely get a distribution that strongly resembles either mixture X, mixture Y, or mixture Z. It would be very unlikely for us to sample a distribution that is 33% topic A, 33% topic B, and 33% topic C.

- That's essentially what a dirichlet distribution provides: a way of sampling probability distributions of a specific type.

That's essentially what a dirichlet distribution provides: a way of sampling probability distributions of a specific type. Recall the model for pLSA:



In pLSA, we sample a document, then a topic based on that document, then a word based on that topic. Here is the model for LDA:



LDA vs pLSA

- From a dirichlet distribution $\text{Dir}(\alpha)$, we draw a random sample representing the *topic distribution*, or topic mixture, of a particular document. This topic distribution is θ . From θ , we select a particular topic Z based on the distribution.
- Next, from another dirichlet distribution $\text{Dir}(\beta)$, we select a random sample representing the *word distribution* of the topic Z . This word distribution is ϕ . From ϕ , we choose the word w .

LDA Summary

- LDA typically works better than pLSA because it can generalize to new documents easily.
- In pLSA, the document probability is a fixed point in the dataset. If we haven't seen a document, we don't have that data point.
- In LDA, the dataset serves as training data for the dirichlet distribution of document-topic distributions. If we haven't seen a document, we can easily sample from the dirichlet distribution and move forward from there.
- With LDA, we can extract human-interpretable topics from a document corpus, where each topic is characterized by the words they are most strongly associated with.
- Furthermore, given a new document, we can obtain a vector representing its *topic mixture*, e.g. 5% topic 1, 70% topic 2, 10% topic 3, etc. These vectors are often very useful for downstream applications.

Code

LDA is easily the most popular (and typically most effective) topic modeling technique out there. It's available in [gensim](#) for easy use:

```
from gensim.corpora.Dictionary import load_from_text, doc2bow
from gensim.corpora import MmCorpus
from gensim.models.ldamodel import LdaModel

document = "This is some document..."

# load id->word mapping (the dictionary)
id2word = load_from_text('wiki_en_wordids.txt')

# load corpus iterator
mm = MmCorpus('wiki_en_tfidf.mm')

# extract 100 LDA topics, updating once every 10,000
lda = LdaModel(corpus=mm, id2word=id2word, num_topics=100,
               update_every=1, chunksize=10000, passes=1)

# use LDA model: transform new doc to bag-of-words, then apply lda
doc_bow = doc2bow(document.split())
doc_lda = lda[doc_bow]

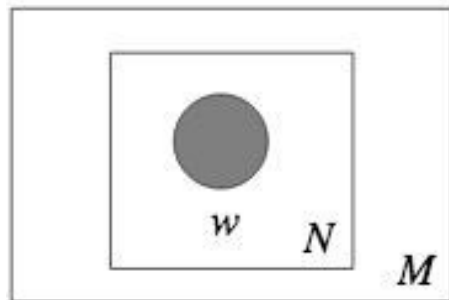
# doc_lda is vector of length num_topics representing weighted
# presence of each topic in the doc
```

Comparison : LDA and other latent variable models

(a) Unigram model

The words of every document are drawn independently from single multinomial distribution.

$$p(\mathbf{w}) = \prod_{n=1}^N p(w_n).$$

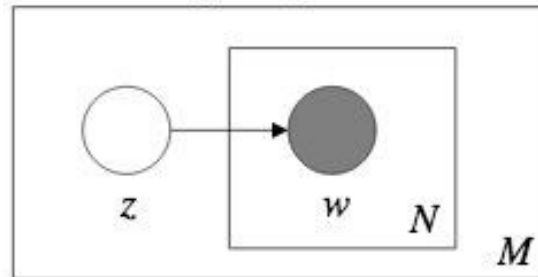


(a) unigram

(b) Mixture of unigram

Augment the unigram model with a discrete random topic variable z and obtain a mixture of unigrams model. Each document is generated by the first choosing topic z and then generating N words independently from the conditional multinomial $p(w|z)$.

$$p(\mathbf{w}) = \sum_z p(z) \prod_{n=1}^N p(w_n|z).$$

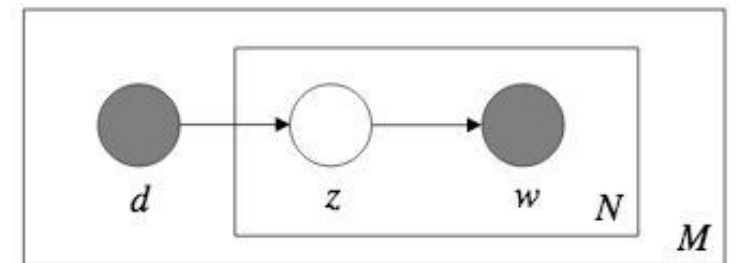


(b) mixture of unigrams

(c) pLSI model

Posits that a document label d and a word w_n are conditionally independent given an unobserved topic z .

$$p(d, w_n) = p(d) \sum_z p(w_n|z)p(z|d)$$



(c) pLSI/aspect model

Drawback of pLSI

- ❑ pLSI does capture the possibility that a document may contain multiple topics
 - $p(z|d)$ serves as the mixture weights of the topics for a particular document d
 - However, d is a dummy index into the list of documents in the training set. Thus d is a multinomial random variable restricted in training documents

- ❑ pLSI is not well-defined generative model of documents
 - There is no natural way to assign probability to a previously unseen document.
 - The parameters for a k -topic pLSI model are K multinomial distributions of size V and M mixtures over the k -hidden topics. This gives $kV + kM$ parameters and therefore linear growth in M . The linear growth in parameters suggests that the model is prone to overfitting.

LDA overcomes pLSI problem

- ❑ By treating the topic mixture weights as a k-parameter hidden random variable rather than a large set of individual parameters which are explicitly linked to the training set

$$p(d, w_n) = p(d) \sum_z p(w_n | z) p(z | d) \quad \text{pLSI}$$

- ❑ LDA is a well-defined generative model and generalize easily to new documents. Furthermore, the $k+kV$ parameters in k-topic LDA model do not grow with the size of training corpus, therefore doesn't suffer from the issue of overfitting.

$$p(\theta, \mathbf{z}, \mathbf{w} | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta) \quad \text{LDA}$$

sources

- <https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05#:~:text=Latent%20Semantic%20Analysis%2C%20or%20LSA,and%20a%20topic%2Dterm%20matrix.>
- <https://www.datacamp.com/tutorial/discovering-hidden-topics-python>
- <https://www.slideshare.net/SoojungHong2/latent-dirichletallocation-presentation>