Topic Modeling: an overview.

Author
Bulzoni Federico

Tutor
Dott. Marzano Enrico

Università degli Studi di Udine
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Corso di Laurea Magistrale in Informatica

2022-02-03

Abstract

In the Natural Language Processing (NLP) field, we refer to **Topic Modeling** as the **set of techniques for discovering latent topics in a collection of documents**. In this cycle of seminars we will explore the **history** of Topic Modeling, giving an overview that starts from the ancestor techniques and ends to nowadays state-of-the-art techniques. This work takes place in the R&D path developed by **Gap srlu** in the context of conversational automation.

Table of contents

Table of contents

Introduction

About Gap srlu

- Gap srlu started an **R&D path** in 2012
- **Focus on**: Data Warehousing, Decision Support System, Machine Learning and Speech Analysis
- **Cooperation with** Prof. Angelo Montanari (DSAV Lab) UNIUD and Prof. Guido Sciavicco UNIFE
- **Academic activities**: publications, seminars, internships, theses, PhD, project conception & funding
- Since 2017 the company is working on **Conversational Automation** and NLP
- Gap designed and developed a **CA platform** called CareN, operational since 2018
- **The present work**: an investigation on Topic Modeling as a well founded semantic viewpoint in understanding texts

The topic modeling task

In the Natural Language Processing (NLP) field, we refer to **Topic Modeling** as the **set of techniques for discovering latent topics in a collection of documents**. [3]

What topics are?

In all the techniques that we are going to explore we can see topics as **latent variables** that describes the given documents. Topics capture some common structure between the given documents, giving us:

What topics are?

In all the techniques that we are going to explore we can see topics as **latent variables** that describes the given documents. Topics capture some common structure between the given documents, giving us:

- the ability of **organizing the documents**,

What topics are?

In all the techniques that we are going to explore we can see topics as **latent variables** that describes the given documents. Topics capture some common structure between the given documents, giving us:

- the ability of **organizing the documents**,
- **information about the content** of the documents.

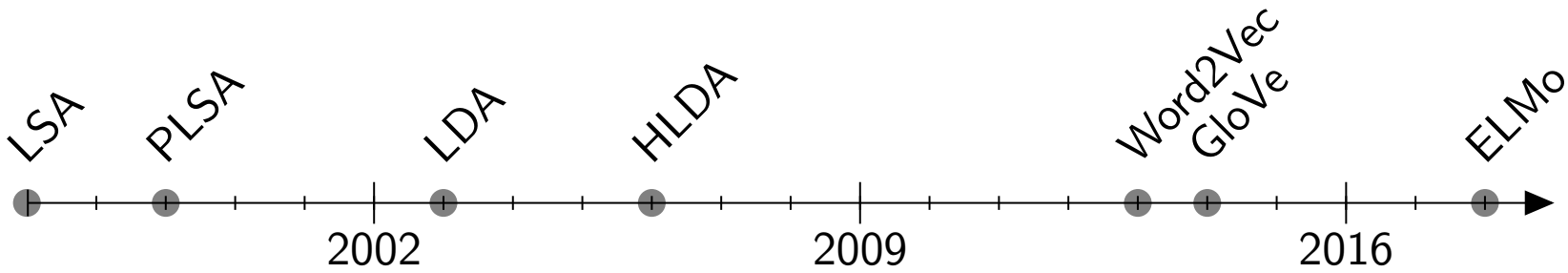
Topic modeling techniques

Topic Modeling techniques are a **family of statistical unsupervised method**, for discovering **latent topics** among a collection of input documents.

Program

1. **Topic Modeling: the ancestor techniques**: Topic Modeling connection with dimensionality reduction: PCA, SVD,LSA.
2. **Topic Modeling: the probabilistic turn**: Topic Modeling in a probabilistic framework: PLSA, LDA, LDA's extensions.
3. **Topic Modeling: word embeddings, the contextual representation problem**: From words to numbers: W2V, ELMO, Sentence Embedding.

The topic modeling timeline



Dimensionality reduction and topic modeling

The dimensionality reduction task

We refer to **dimensionality reduction** as the task of projecting data from a high-dimensional space into a **lower-dimensional space** (w.r.t. the original one), so that the projected data keep the meaningful properties of the original data while ignoring meaningless relationships between data.

An analogy with topic modeling

- **Topic modeling** goal: **finding short descriptions** of text corpora that enable efficient processing of large collections, while **preserving the essential statistical relationships** between the data.

An analogy with topic modeling

- **Topic modeling** goal: **finding short descriptions** of text corpora that enable efficient processing of large collections, while **preserving the essential statistical relationships** between the data.
- **Dimensionality reduction** goal: find a **low-dimensional representation** of the data, based on small number of interesting dimensions, **that captures as much of the information as possible**.

Principal Component Analysis (PCA)

Goal: finding new variables that are linear combinations of those in the original dataset, that successively maximize variance and that are uncorrelated with each other.This new variables are called **principal components** (PCs).

History

Principal Component Analysis (PCA) was invented in **1901** by **Karl Pearson** [7], and it was later independently developed and named by **Harold Hotelling** in the **1930s** [9].

How principal components can be computed?

The principal components of a dataset X, with M features and N observations can be computed via **spectral decomposition** of the **sample covariance matrix** C. X is a matrix where each column X_m with $m \in [1, M]$ represents the observed values for the m-th feature on the N observations.

Sample covariance - 1

The **sample covariance** quantifies the joint variability between two observed variables.
Let $\mathrm{x}_i = (x_{1,i}, \ldots, x_{N,i})$ and $\mathrm{x}_j = (x_{1,j}, \ldots, x_{N,j})$ be the values of the i-th and j-th features on the N observations; $\overline{x_i}$ and $\overline{x_j}$ be the sample mean for $\mathrm{x}_i, \mathrm{x}_j$ respectively. The sample covariance between x_i and x_j is:

$$c_{i,j} = \frac{1}{N-1} \sum_{n=1}^{N} (x_{n,i} - \overline{x_i})(x_{n,j} - \overline{x_j})$$

Sample covariance -2

Notice that if the two features have both **zero mean**, $\overline{x_i} = 0 = \overline{x_j}$ then, the sample covariance is:

$$c_{i,j} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{n,i}x_{n,j}) = \frac{1}{N-1}(\mathrm{x}_i \cdot \mathrm{x}_j)$$

.

Assuming that our dataset X have all the features with zero-mean, the **sample covariance matrix** C can be computed as:

$$\mathrm{C} = \frac{\mathrm{X}^T\mathrm{X}}{N-1}$$

Finding the first PC I

The first Principal Component is a linear combination of the column of X, $\sum_{m=1}^{M} a_m x_m = \mathrm{X}\mathrm{a}$ that maximize variance. The variance of such a linear combination, is given by

$$\begin{aligned} var(\mathrm{X}\mathrm{a}) =& \frac{(\mathrm{X}\mathrm{a})^T(\mathrm{X}\mathrm{a})}{N-1} \\ =& \mathrm{a}^T \frac{\mathrm{X}^T\mathrm{X}}{N-1}\mathrm{a} \\ =& \mathrm{a}^T\mathrm{C}\mathrm{a} \end{aligned} \tag{1}$$

moreover, we require the vector a to have unitary norm, namely:

$$\sum_{m=1}^{M} a_m^2 = \mathrm{a}^T\mathrm{a} = 1$$

Finding the first PC II

This constrained maximization problem is equivalent to maximizing:

$$a^T Ca - \lambda(a^T a - 1) \tag{2}$$

where λ is a Lagrange multiplier, its solution is given by:

$$Ca = \lambda a$$

Thus, we are searching for the **autovector** a of the sample-covariance matrix C with the largest associated **autovalue** λ.

Finding the first PC III

The vector a is called **loading vector** of the first principal component, the **first principal component** is the projection of X on the loading vector a, hence Xa.

Finding the subsequent PCs

We need to introduce an **orthogonality constraint** for finding the loading vectors $\mathrm{a}_2, \ldots, \mathrm{a}_M$ of the following principal components $\mathrm{Xa}_2, \ldots, \mathrm{Xa}_M$. Since C is a real-symmetric matrix it has exactly M eigenvalues $\lambda_1, \ldots, \lambda_M$ and we can define the corresponding eigenvectors to form an orthonormal set of vectors $\mathrm{a}_1, \ldots, \mathrm{a}_M$, namely, for all $i, j \in [1, M]$:

$$\begin{cases} \mathrm{a}_i^T \mathrm{a}_j = 1 & \text{if } i = j \\ \mathrm{a}_i^T \mathrm{a}_j = 0 & \text{otherwise} \end{cases}$$

Result

The new orthonormal linear transformation found by PCA, is then $(a_1, \ldots, a_M)$, the vector of **loadings**, it can be view in matricial form as

$$A = \begin{pmatrix} a_{1,1} & \ldots & a_{1,M} \\ a_{2,1} & \ldots & a_{2,M} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \ldots & a_{M,M} \end{pmatrix}$$

The projected dataset is then given by:

$$S = XA$$

S containes the re-oriented data, the columns of S_X are the **principal components**, while the single entries are called **scores**. For instance, $S_{i,1}$, $i \in [1, N]$ is the score of the i-th observation on the first principal component.

Dimensionality reduction with PCA

For the purpose of dimensionality reduction we can project the original data onto the first $K < M$ principal components. For do that we build a matrix $\mathrm{A_K}$ were we simply set to zero the columns of A associated with the $M - K$ smaller eigenvalues.
The projection of X on the first K principal components is then:

$$\mathrm{S}_K = \mathrm{X}\mathrm{A}_K.$$

Singular Value Decomposition (SVD) I

Definition 2.1. Singular Value Decomposition (SVD) Any matrix $X \in \mathbb{C}^{N \times M}$ can be rewritten as the product of three matrices:

$$X = U \Sigma V^*$$

where $U \in \mathbb{C}^{N \times N}$ is a unitary matrix, $\Sigma \in \mathbb{C}^{N \times M}$ is a diagonal matrix and $V \in \mathbb{C}^{M \times M}$ is a unitary matrix; V^* is the conjugate transpose of V. The number of singular values in Σ is equal to the **rank** of X, $r(X)$.

Singular Value Decomposition (SVD) II

Restricting to the case of SVD applied to a real matrix $X \in \mathbb{R}^{N \times M}$. we have that:

$$X = U \Sigma V^T \tag{3}$$

where U and V are orthonormal matrices.

Singular Value Decomposition (SVD) III

- U columns are the **left-singular vectors** of X, they are the **eigenvectors** of XX^T.
- V columns are the **right-singular vectors** of X, they are the **eigenvectors** of X^TX.
- Σ diagonal entries are the **singular values** of X, they are the square root of the non-zero eigenvalues of XX^T (and of X^TX).
- The diagonal entries of Σ and the corresponding left-singular vectors in U and right-singular vectors in V are arranged by **descending order of singular values**.

Relationship with PCA I

If X is a matrix with mean-centered columns, then, the orthonormal autovectors of $X^T X$ and of the sample covariance matrix $C = X^T X/(N-1)$ are the same. It follows immediately that the matrix of right-singular vectors V corresponds to the matrix of **principal components loadings** A.

Relationship with PCA II

The **autovalues of the sample covariance matrix** C are the square of the singular values of X divided by $(N-1)$.
Moreover, we have that the **matrix of principal components** $S \in \mathbb{R}^{N \times M}$ corresponds to:

$$\begin{aligned} S =& XA \\ =& XV \\ =& U\Sigma(V^T V) \\ =& U\Sigma \end{aligned} \tag{4}$$

Dimensionality reduction with SVD I

Given the SVD decomposition of a matrix $\mathrm{X} = \mathrm{U}\Sigma\mathrm{V}^T \in \mathbb{R}^{N\times M}$ we can perform dimensionality reduction on X by keeping only the largest $K < r(\mathrm{X})$ and setting the others equal to zero obtaining Σ_K and the approximated matrix

$$\widetilde{\mathrm{X}}_K = \mathrm{U}\Sigma_K\mathrm{V}^T \tag{5}$$

Dimensionality reduction with SVD II

Theorem 2.1 (Eckart-Young-Mirsky). *For either the 2-norm* $\|\cdot\|_2$ *or the Frobenius-norm* $\|\cdot\|_F$

$$\|\mathrm{X} - \widetilde{\mathrm{X}}_K\| \leq \|\mathrm{X} - \mathrm{Y}\| \text{ for all the matrices } \mathrm{Y} \text{ of rank } K$$

LSA: Basic definitions I

Before starting at exploring how LSA works we give some basic definitions that will be useful in the subsequent analysys.

Definition 2.2 (Corpora $\mathcal{D}$, document d and word w)**.** A **Corpora** $\mathcal{D}$ is a set of M documents

$$\mathcal{D} = \{\mathrm{d}_1, \ldots, \mathrm{d}_M\}$$

A **document** d is a D-tuple of words

$$\mathrm{d} = (w_1, \ldots, w_D)$$

A **word** w in our context is the basic unit of discrete data.

LSA: Basic definitions II

Definition 2.3 (Bag-of-Words (BoW) representation of a document)**.** Given a vocabulary of words $\mathcal{W} = \{w_1, \ldots, w_N\}$ and a document $\mathrm{d} = (w_{d,1}, \ldots, w_{d,M})$ its BoW representation $\mathrm{b_d}$ is a N-dimensional vector such that $\mathrm{b}_{\mathrm{d},n}$ for all $n \in [1, N]$ is equal to the number of occurrences of word w_n in document d.

LSA: Basic definitions III

Definition 2.4 (Term-document matrix)**.** Let $\mathcal{D} = \{d_1, \ldots, d_M\}$ be a set of documents. Let $N \in \mathbb{N}$ be the number of different words contained in $\mathcal{D}$, and let $\mathcal{W} = \{w_1, \ldots, w_N\}$ be the set of different words contained in $\mathcal{D}$. The **term-document** matrix $X \in \mathbb{R}^{N \times M}$ of $\mathcal{D}$ is a matrix where each row is associated to a word in $\mathcal{W}$ and each column is associated to a document in $\mathcal{D}$, and contains its bag-of-words representation.

Example of term-document matrix

Let $\mathcal{D} = \{d_1, d_2, d_3\}$ where:

$$d_1 = \textit{Dogs eat bones.}$$
$$d_2 = \textit{Cats eat fishes.}$$
$$d_3 = \textit{Computers don't eat fishes and don't eat bones.}$$

The corresponding term-document matrix X is:

$$X = \begin{array}{c} \\ \text{dogs} \\ \text{eat} \\ \text{bones} \\ \text{cats} \\ \text{fishes} \\ \text{computers} \\ \text{don't} \\ \text{and} \end{array} \begin{array}{c} \begin{array}{ccc} d_1 & d_2 & d_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Latent Semantic Analysis (LSA) - 1997

We recall that the main goal of topic modeling is to find **topics**, namely, latent variables, that describe relationships between the data.
In our case, topics describe **relationships between words and documents**.

Latent Semantic Analysis (LSA) - 1997

We recall that the main goal of topic modeling is to find **topics**, namely, latent variables, that describe relationships between the data.
In our case, topics describe **relationships between words and documents**.

LSA's idea: apply SVD to the term-document matrix of the given corpora in order to find this latent variables.

History

LSA was originally presented in **1997** by *Thomas K. Landauer* and *Susan T. Dumais* in the paper [11].

How LSA works? I

Given a corpora $\mathcal{D} = \{d_1, \ldots, d_M\}$, LSA applies dimensionality reduction on the correspondent term-document matrix X using the **Singular Value Decomposition** and keeping only the largest K singular values.
In this case K is a **hyperparameter** we can select and adjust to reflect the number of topics we expect to be in our documents.

How LSA works? II

$$\widetilde{X}_K = U\Sigma_K V^T$$

The components of the reconstructed matrix are:

- V, the **document-topic matrix**.
- Σ_K is the diagonal matrix where σ_k, $k \in [1, K]$ represents the **relevance** of the k-th topic in the considered set of documents.
- U the **term-topic matrix**.

How LSA works? III

LSA represents the meaning of a word as a kind of average of the meaning of all the passages in which it appears, and the meaning of a passage as a kind of average of the meaning of all the words it contains [10].

This works as when we read a passage, where we don't know the meaning of a word, for example:

Kitties eat fishes.

We may don't know the meaning of *kitties*, but by the fact that the word is used in a **similar context** as the word *cats*, it may suggest that they have **similar meanings**.

Technical memoranda example I

Consider the following text passages, they are technical memoranda, **five about HCI** (Human Computer Interaction), and **four about graph theory**:

- c_1: *Human* machine *interface* for ABC *computer* applications.
- c_2: A *survey* of *user* opinion of *computer system response time*.
- c_3: The *EPS user interface* management *system*.
- c_4: *System* and *human system* engineering testing of *EPS*.
- c_5: Relation of *user* perceived *response time* to error measurement.

- m_1: The generation of random, binary, ordered *trees*.
- m_2: The intersection *graph* of paths in *trees*.
- m_3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering.
- m_4: *Graph minors*: A *survey*.

Technical memoranda example II

We consider only words used at least in two different documents, and we esclude stopwords (e.g: the, a, etc.) We get the following $N \times M$, $N = 12, M = 9$ term-document matrix X:

$$X = \begin{array}{c} \\ \text{human} \\ \text{interface} \\ \text{computer} \\ \text{user} \\ \text{system} \\ \text{response} \\ \text{time} \\ \text{EPS} \\ \text{survey} \\ \text{trees} \\ \text{graph} \\ \text{minors} \end{array} \begin{array}{c} \begin{array}{ccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Technical memoranda example III

Applying SVD decomposition on the matrix X and retaining only the two most significant dimensions (which have the two highest singular values) we obtain the reconstructed matrix $\widetilde{X}_2$:

	c_1	c_2	c_3	c_4	c_5	m_1	m_2	m_3	m_4
human	0.16	0.40	0.38	0.47	0.18	−0.05	−0.12	−0.16	−0.09
interface	0.14	0.37	0.33	0.40	0.16	−0.03	−0.07	−0.10	−0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	−0.07	−0.15	−0.21	−0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	−0.07	−0.14	−0.20	−0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	−0.06	0.23	−0.14	−0.27	0.14	0.24	0.55	0.77	0.66
graph	−0.06	0.34	−0.15	−0.30	0.20	0.31	0.69	0.98	0.85
minors	−0.04	0.25	−0.10	−0.21	0.15	0.22	0.50	0.71	0.62

Technical memoranda example IV

Look at the values for *survey* and *trees* in column m_4, the word *trees* **did not appear in this document**, but by the fact that contain *graph* and *minors*, which are higly related with *trees*, the zero entry has been replaced with 0.66. On the other hand, the value 1 for *survey* has been replaced with 0.42 **reflecting the fact that is unusual in this context and then it should be counted as unimportant in characterizing the passage**.

Technical memoranda example V

Now consider the rows in the original and in the reconstructed matrix for terms: *human* and *user*. **Intuitively they are highly related**, but if we compute the **cosine similarity** in the original matrix between *human* and *user*, we obtain

$$sim_{cos}(\text{human}, \text{user}) = 0$$

indicating decorrelation.
On the other hand if we compute the cosine similarity between human' and user' of the reconstructed matrix we obtain:

$$sim_{cos}(\text{human}', \text{user}') = 0.88$$

indicating intermediate-high similarity.

Technical memoranda example VI

The same considerations hold also for **similarities between documents**. Consider documents c_1 and c_2 in the original matrix their cosine similarity is:

$$sim_{cos}(c_1, c_2) = 0.23$$

indicating low similarity.
But considering c_1', c_2' of the reconstructed matrix we obtain:

$$sim_{cos}(c_1', c_2') = 0.91$$

indicating high similarity.

Considerations

LSA brings the first idea of *topic modeling*, topics are **latent variables** that best describes the documents considered in a lower-dimensional semantic space (w.r.t. the semantic space of all documents). It's fast to perform since it's a matrix factorization, and is a simple approach that helps to bring out similarities between words and documents. However with LSA we are not able of extract some useful information about the topic founds, it **lacks of interpretability**, analyzing the found **document-topic** and **term-topic** matrices we can not say anything about the topics theirselves.

Probabilistic topic modeling

Recap

- **Topic modeling task**: discover **latent topics** in a collection of documents.

Recap

- **Topic modeling task**: discover **latent topics** in a collection of documents.
- **Topics**: **latent variables** that capture some **common structure** among the given documents.

Recap

- **Topic modeling task**: discover **latent topics** in a collection of documents.
- **Topics**: **latent variables** that capture some **common structure** among the given documents.
- **Topic modeling techniques**: family of **unsupervised methods** for discovering latent topics among a collection of documents.

Recap

- **Topic modeling task**: discover **latent topics** in a collection of documents.
- **Topics**: **latent variables** that capture some **common structure** among the given documents.
- **Topic modeling techniques**: family of **unsupervised methods** for discovering latent topics among a collection of documents.
- **Latent Semantic Analysis (LSA)**: the first topic modeling technique called LSA is based on the **matrix decomposition** of the **term-document matrix** followed by a **dimensionality reduction** step. We have seen how this simple technique helps to bring out similarities between words and documents.

Topics as probability distributions

Idea: topics are probability distributions over words.

Topics as probability distributions

Idea: topics are probability distributions over words. **Example**: if we find a topic *animals* we would expect that words related to animals, as *dog* have higher probability associated than words as *parliament*.

Documents as mixture of topics

Idea: documents are mixtures of topics, hence, they can be viewed as probability distributions over topics.

Documents as mixture of topics

Idea: documents are mixtures of topics, hence, they can be viewed as probability distributions over topics. **Example**: if we take the newspaper *La Gazzetta dello Sport* we would expect that the prevalent topic in it is *sport*, with a small percentage of other topics as *gossip* and *politics*. We can say for example that the newspaper is 80% *sport*, 15% *gossip* and 5% *politcs*.

Generative model

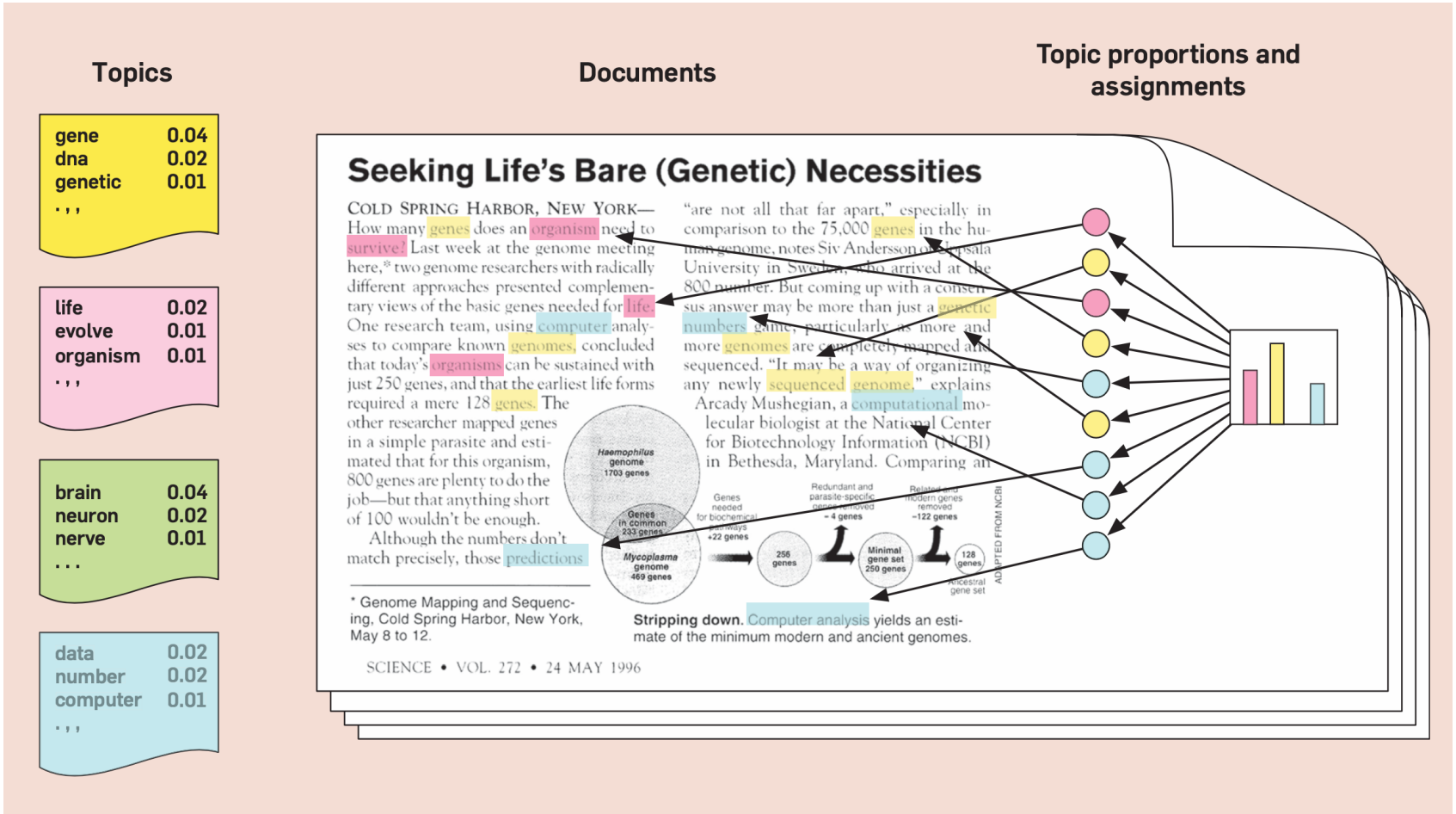
Probabilistic topic models assume a **generative model** for the documents.

Generative model

Probabilistic topic models assume a **generative model** for the documents. **Example**: For a document d with N_{d} words:

1. pick a topic with probability $P(z \,|\mathrm{d})$
2. pick N_{d} words with probabilities $P(w \,|z)$

Probabilistic topic modeling output



Interpretability and polysemy handling

LSA's major problem: **lack of interpretability** of the term-topic and document-topic matrices.

Interpretability and polysemy handling

LSA's major problem: **lack of interpretability** of the term-topic and document-topic matrices. With probabilistic topic modeling we overcome this problem since probability distributions are intuitively interpretable by a human.

Interpretability and polysemy handling

LSA's major problem: **lack of interpretability** of the term-topic and document-topic matrices. With probabilistic topic modeling we overcome this problem since probability distributions are intuitively interpretable by a human.

Polysemy's handling: if a word has more than one meaning this can be captured by probabilistic topic models since that word will appear with high probability in the topics associated to its meanings.

Interpretability and polysemy handling

LSA's major problem: **lack of interpretability** of the term-topic and document-topic matrices. With probabilistic topic modeling we overcome this problem since probability distributions are intuitively interpretable by a human.

Polysemy's handling: if a word has more than one meaning this can be captured by probabilistic topic models since that word will appear with high probability in the topics associated to its meanings.
Example: word *mouse* topics: $\{$*animals*, *computer's I/O devices*$\}$.

Training I

Given a generative model for the documents, and a set of training documents the task is then to find suitable values for the **parameters** in the generative model, such that the model **maximize the likelihood** of generate the given training documents.

Training II

Given a set of input documents $\mathcal{C}$ and a set of parameters for the generative model θ we want to maximize the probability of having $\mathcal{C}$ given θ:

$$\arg\max_{\theta} P(\mathcal{C} \mid \theta)$$

Training III

Definition 3.1 (Bayes' rule)**.** Let $\mathcal{C}$ be our training data and θ be the parameters of the considered probabilistic model, then we have that:

$$P(\theta \mid \mathcal{C}) = \frac{P(\theta)P(\mathcal{C} \mid \theta)}{P(\mathcal{C})} \tag{6}$$

where:

- $P(\theta \mid \mathcal{C})$: **posterior distribution**,
- $P(\mathcal{C} \mid \theta)$: **likelihood distribution**,
- $P(\theta)$: **prior distribution**
- $P(\mathcal{C})$: **evidence**

Training IV

Problem: the **evidence** is the marginal probability of the training data, which is the probability of seeing the observed corpus under any possible configuration of the parameters θ:

$$P(\mathcal{C}) = \int_{\theta} P(\mathcal{C} \mid \theta) P(\theta) \, d\theta$$

that is **intractable** to compute in most of the cases.

Training V

It follows, that since the posterior distribution can not be computed exactly, we need to approximate it, this can be done with:

- **MCMC Sampling-approach** (e.g. Gibbs Sampling),
- **EM algorithm**
  - **Variational methods**

MCMC Sampling approach

- **Markov Chain Monte Carlo sampling approach** consists in the idea of randomly pick some samples from a **Markov chain** whose stationary distribution is the intractable posterior distribution.

MCMC Sampling approach

- **Markov Chain Monte Carlo sampling approach** consists in the idea of randomly pick some samples from a **Markov chain** whose stationary distribution is the intractable posterior distribution.
- The Markov chain is such that, the next sample that is drawn from the posterior distribution is dependent upon the last sample that was drawn and the observed data.

MCMC Sampling approach

- **Markov Chain Monte Carlo sampling approach** consists in the idea of randomly pick some samples from a **Markov chain** whose stationary distribution is the intractable posterior distribution.
- The Markov chain is such that, the next sample that is drawn from the posterior distribution is dependent upon the last sample that was drawn and the observed data.
- Then we can pick a **subset of the samples** in order to estimate the actual posterior distribution.

MCMC Sampling approach

- **Markov Chain Monte Carlo sampling approach** consists in the idea of randomly pick some samples from a **Markov chain** whose stationary distribution is the intractable posterior distribution.
- The Markov chain is such that, the next sample that is drawn from the posterior distribution is dependent upon the last sample that was drawn and the observed data.
- Then we can pick a **subset of the samples** in order to estimate the actual posterior distribution.
- **Examples**: Gibbs sampling, Metropolis-Hastings sampling.

EM algorithm I

The **Expectation-Maximization algorithm** is an iterative procedure for finding a **local maximum of a latent variable model likelihood**. Let X be the observable data, Y be the latent variables and θ be the parameters of the latent variable model. We want to maximize:

$$l(\theta) = \log P(\mathrm{X}\ |\theta) = \log \int_{\mathrm{Y}} P(\mathrm{X}, \mathrm{Y}\ |\theta)\, d\mathrm{Y}$$

The idea is to define a **probability distribution over the latent variables** $Q(\mathrm{Y})$ and use it for obtaining a lower bound $\mathcal{F}(Q, \theta)$ on the latent variable model log-likelihood.

EM algorithm II

By some math, using **Jensen's inequality** it turns out that:

$$l(\theta) = \mathcal{F}(Q, \theta) + KL(Q(\mathrm{Y}) \,\|P(\mathrm{Y} \,|\mathrm{X}, \theta))$$

where $KL(Q(\mathrm{Y}) \,\|P(\mathrm{Y} \,|\mathrm{X}, \theta))$ is the **Kullback-Leiber divergence** of approximating $P(\mathrm{Y} \,|\mathrm{X}, \theta))$ using $Q(\mathrm{Y})$.
Maximizing $\mathcal{F}(Q, \theta)$ corresponds hence to **minimizing the divergence** between $P(\mathrm{Y} \,|\mathrm{X}, \theta))$ and $Q(\mathrm{Y})$.

EM algorithm III

Initially, the values of the parameteres θ are randomly chosen, then at each iteration, until convergence:

1. **E-step**: optimize $\mathcal{F}(Q, \theta)$ wrt distribution over hidden variables holding parameters fixed:

$$Q^k(Y) = \arg\max_{Q(Y)} \mathcal{F}(Q(Y), \theta^{k-1})$$

2. **M-step**: maximize $\mathcal{F}(Q, \theta)$ wrt parameters holding hidden distribution fixed:

$$\theta^k = \arg\max_{\theta} \mathcal{F}(Q^k(Y), \theta)$$

Probabilistic Latent Semantic Analysis (PLSA) - 1999

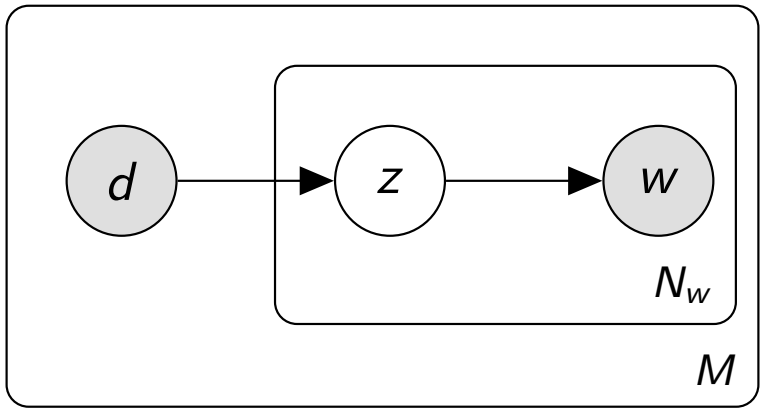
Probabilistic Latent Semantic Analysis (PLSA) [8] was presented in 1999 by *Thomas Hoffman*. It was the first method at introducing the idea of defining a **generative model** for documents where we have that **topics** are probability distributions over words and **documents** are **mixture of topics**.

Generative process

The **generative process** of PLSA assumes that given K **latent topics** $\{z_1, \ldots, z_K\}$ a document d is generated:

1. picking d with probability $P(\text{d})$,
2. each word w in d is generated by:
   - 2.1 picking a topic z with probability $P(z\,|\text{d})$,
   - 2.2 picking word w with probability $P(w\,|z)$

Graphical model



Shaded circles indicate observed variables, while unshaded circles represent latent variables. Plates represent repetitions, the number of repetitions of the process is indicated on the right below of the plate.

Assumptions

The presented model, called **aspect model** is a statistical **mixture model** which is based on two **independence assumptions**:

- the **bag-of-words** assumption and
- the **conditional independence** assumption.

Bag-of-words

The joint variable (d, w) is **independently sampled**.
Intuitively this correspond to consider each document as an unordered collection of words (bag-of-words model). A consequence is that the joint distribution of the observed data can be seen as the product of the distributions of each observation:

$$P(\mathcal{C}) = \prod_{\mathrm{d} \in \mathcal{C}} \prod_{w \in \mathrm{d}} P(d, w)$$

.

Conditional independence

Words and documents are assumed to be conditionally independent given the topic. In other words, conditioned on topic z, words w are generated independently of the specific document d. It follows that:

$$P(w, d|z) = P(w|z)P(d|z)$$

or, equivalently:

$$P(w|d, z) = P(w|z)$$

Joint distribution - model definition

Following the generative model, the **probability of seeing a given document d and a given word w together** is:

$$P(d, w) = P(d)P(w|d)$$

where:

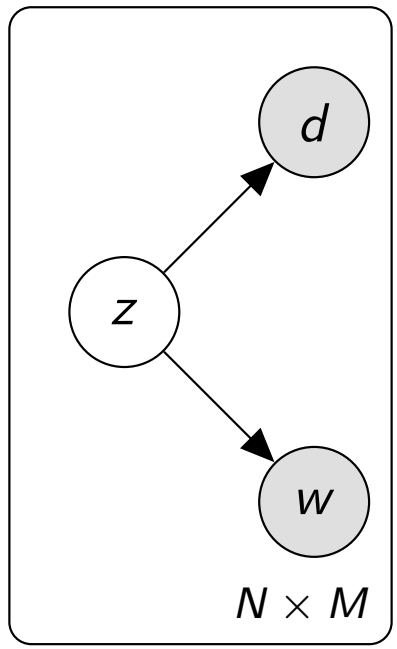
$$P(w|d) = \sum_z P(z|d)P(w|z)$$

Another point of view

Using the conditional independence assumption, the previous formula, can be rewritten as:

$$P(d, w) = \sum_z P(w|z)P(d|z)P(z)$$

The corresponding generative model is:



Correspondance with LSA

We can exploit this new **parametrization** to see the **link between PLSA and LSA**. If we look to this equation:

$$P(d, w) = \sum_z P(w|z)P(d|z)P(z)$$

$P(w|z)$ corresponds to the **term-topic** matrix U, $P(z)$ corresponds to the **diagonal matrix of singular values for topics** Σ and finally $P(d|z)$ corresponds to the **document-topic** matrix V.

Model fitting I

The latent parameters of the model are $P(w|z)$ and $P(z|d)$, they are $(N-1) \times K$ and $(K-1) \times M$ respectively. Hence, the total number of parameters **grows linearly with the size of the corpus**, making the **model predisposed to overfitting**.

Model fitting II

In the original article [8] Hoffman presents a **cross-validation** procedure called **Tempered Expectaction Maximization** for inferring the latent parameters of PLSA accounting for the fact that the model is predisposed to overfitting.

Unseen documents

PLSA exhibits another major problem, we have not a method for generating unseen documents with the trained model since **we have only the topic proportions for the training documents**.

Unseen documents

PLSA exhibits another major problem, we have not a method for generating unseen documents with the trained model since **we have only the topic proportions for the training documents**.
Huffman proposes a so-called **fold-in** technique that consists in keeping the topics discovered on the training documents and compute by an EM-procedure the topic proportions for the unseen document.

Considerations I

- (+): PLSA offers **interpretable results**, overcoming the major problem of LSA.
- (+): It offers a more **solid statistical basis** than LSA, defining a generative model for documents.

Considerations II

- (-): it is **prone to overfitting**, since the number of parameters of the model increase linearly with the size of the training corpus.
- (-): PLSA does not make any assumptions about how the topic proportions are generated, making it **difficult to test the generalizability** of the model to new documents.
- (-): there is **no natural way of assigning topic proportions to unseen documents**.

Latent Dirichlet Allocation (LDA) - 2003 I

In the context of population genetics, LDA was proposed by *J. K. Pritchard*, *M. Stephens* and *P. Donnelly* in 2000. *David Blei*, *Andrew Ng* and *Michael I. Jordan* rediscovered independently it in 2003 [5] as a topic modeling technique.

Latent Dirichlet Allocation (LDA) - 2003 II

Latent Dirichlet Allocation (**LDA**) overcome PLSA's main problem: the lack of a generative model for the mixing proportions of topics in documents.

Latent Dirichlet Allocation (LDA) - 2003 III

Idea: **documents are represented as random mixtures over latent topics**, where each topic (as for PLSA) is a probability distribution over words.

Distributions of topics over documents are drawn by a **Dirichlet distribution**; a K-dimensional Dirichlet random variable can take values in the $(K-1)$-simplex, so each component lies in $[0,1]$ and they sum to 1... Probability distribution!

Dirichlet distribution I

The Dirichlet distribution is a **distribution on probability distributions**, $\vec{\theta} = (\theta_1, \ldots, \theta_K)$, $\theta_k \geq 0$ and $\sum_k \theta_k = 1$. It is controlled by a vector of non-negative K-values $\vec{\alpha} = (\alpha_1, \ldots, \alpha_K)$:

$$p(\vec{\theta}|\vec{\alpha}) = \frac{1}{B(\vec{\alpha})} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \tag{7}$$

where the normalizing costant is the **multivariate beta function**.

Dirichlet distribution II

The Dirichlet distribution is the **conjugate prior** distribution of the categorical distribution and more generally of the **multinomial distribution**.

Definition 3.2. Conjugate prior

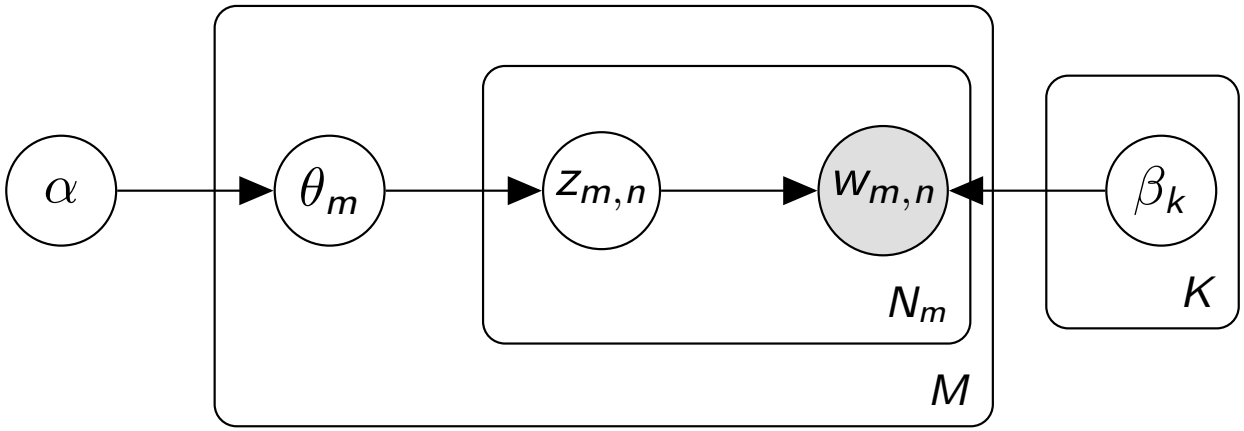
$$P(\theta \mid \mathcal{C}) = \frac{P(\theta)P(\mathcal{C} \mid \theta)}{P(\mathcal{C})} \tag{8}$$

If the posterior $P(\theta \mid \mathcal{C})$ is in the same probability distribution as the prior $P(\theta)$ then they are called **conjugate distributions**, and the prior is called a **conjugate prior** for the likelihood function $P(\mathcal{C} \mid \theta)$.

Dirichlet distribution III

In such case, starting from what we know about the parameter prior to observing the data point, we then can update our knowledge based on the data point and end up with a new distribution of the same form as the old one.
This means that we can successively **update our knowledge of a parameter** by incorporating new observations one at a time, **without running into mathematical difficulties**.

Graphical model



where:

- $\alpha = (\alpha_1, \ldots, \alpha_K)$ is a **Dirichlet parameter**.
- θ_m is the **topic distribution over document** m.
- $z_{m,n} \in [1, K]$ is the **topic assignment** for the n-th word in document m.
- $w_{m,n}$ is the n-th **word** in document m.
- β_k is the **word distribution over topic** k.

Generative process

LDA assumes the following generative process, for each document $m \in \{1, \ldots, M\}$:

1. θ_m is chosen from a Dirichlet distribution with parameter α;
2. for each of the N_m words in document m:
   - 2.1 the topic assignment $z_{m,n}$ for the n-th word is chosen from θ_m,
   - 2.2 the n-th word $w_{m,n}$ is chosen from the distribution $\beta_{z_{m,n}}$.

Focus on β_k and $z_{m,n}$ I

Each distribution β_k over the N words of the dictionary can be seen as a **column of an** $N \times M$ **matrix** β where at each row corresponds a word of the dictionary. $z_{m,n}$ **indicates which distribution** β_k we have to look at for word $w_{m,n}$, graphically:

$$\beta = \begin{array}{c} \\ w_1 \\ w_2 \\ \vdots \\ w_n = w_{m,n} \\ \vdots \\ w_N \end{array} \begin{array}{c} \begin{array}{cccccc} \beta_1 & \beta_2 & \dots & \beta_k = \beta_{z_{m,n}} & \dots & \beta_K \end{array} \\ \begin{pmatrix} 0.05 & 0.10 & 0.03 & 0.21 & 0.12 & 0.17 \\ 0.10 & 0.10 & 0.07 & 0.09 & 0.16 & 0.03 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.01 & 0.03 & 0.12 & 0.06 & 0.04 & 0.20 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.02 & 0.11 & 0.07 & 0.01 & 0.02 & 0.10 \end{pmatrix} \end{array}$$

Focus on β_k and $z_{m,n}$ II

By looking at the previous example we can conclude that the probability of seeing word $w_{m,n}$ in document m given the topic assignment $z_{m,n}$ and the word over topic distributions $\beta_{1...K} = \beta$ is:

$$P(w_{m,n}|z_{m,n}, \beta) = P(w_{m,n}|\beta_{z_{m,n}})$$

In our example:

$$P(w_{m,n}|z_{m,n}, \beta) = P(w_n|\beta_k) = 0.06$$

Joint distribution - model definition

By the previous example we can easily understand the joint distribution model definition for LDA. Given the Dirichlet parameter α and the word over topic distributions $\beta_{1...K}$, the probability of choose word $w_{m,n}$ in document m is:

$$P(w_{m,n}|\alpha,\beta) = P(\theta_m|\alpha)\prod_{k=1}^{K} P(z_k|\theta_m)P(w_{m,n}|\beta_{z_k})$$

How α influences the Dirichlet? I

With α we can influence how topics are more likely to be distributed over documents. $\alpha = (\alpha_1, \ldots, \alpha_K)$ is a K-**dimensional vector of positive values**, we define

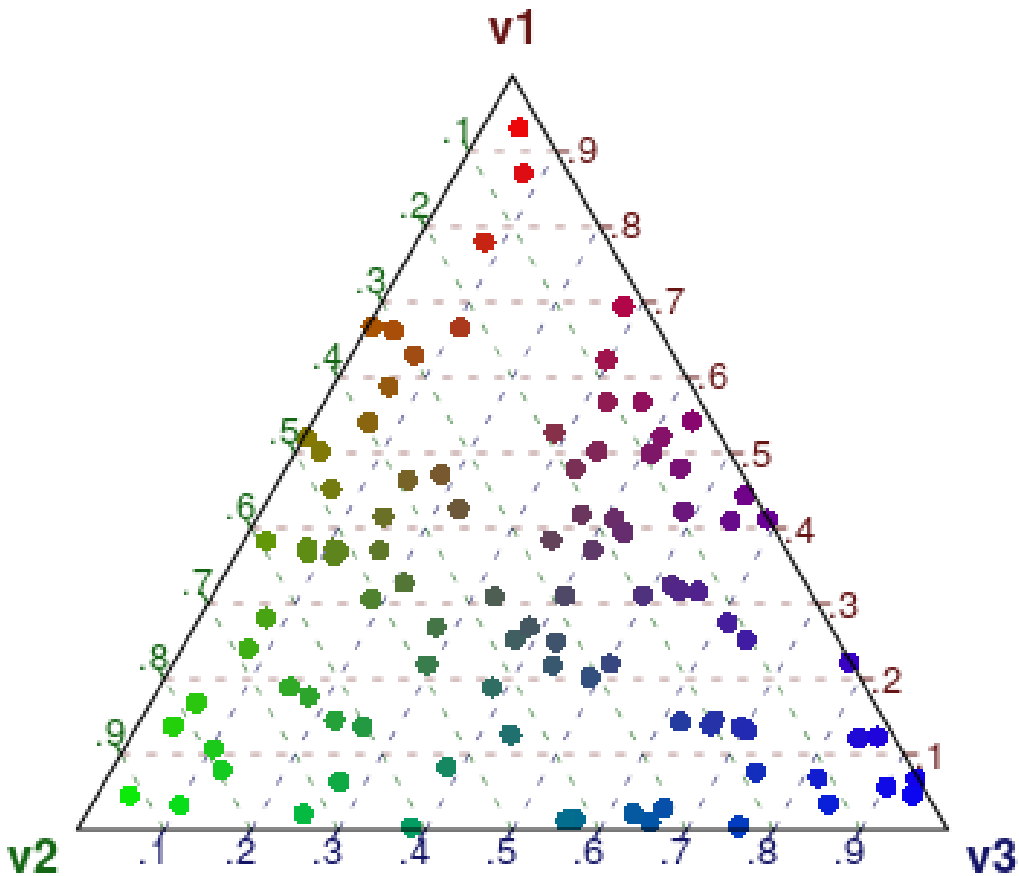
$$s = \sum_{k=1}^{K} \alpha_k$$

as the **scaling parameter**, Higher the scaling parameter is, more concentrated around the expected values the components will be.
On the other hand the **expected value** for the k-th component is given by:

$$E(\theta_k|\alpha) = \frac{\alpha_k}{s}$$

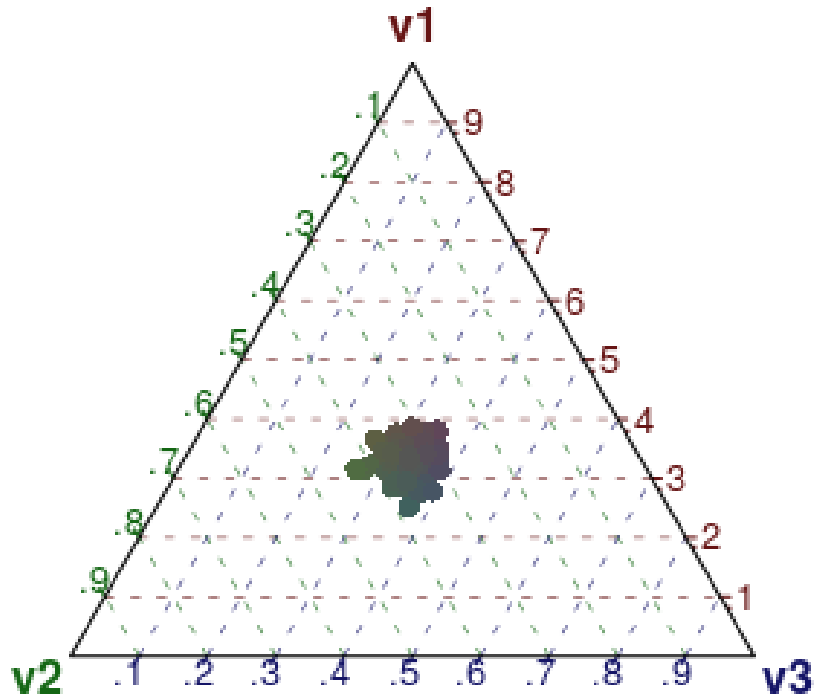
How α influences the Dirichlet? II

Uniform Dirichlet $\alpha = (1, 1, 1)$: with $\alpha = (1, \ldots, 1)$ we obtain the **uniform dirichlet**, where all the points of the symplex have the same probability to be draw.



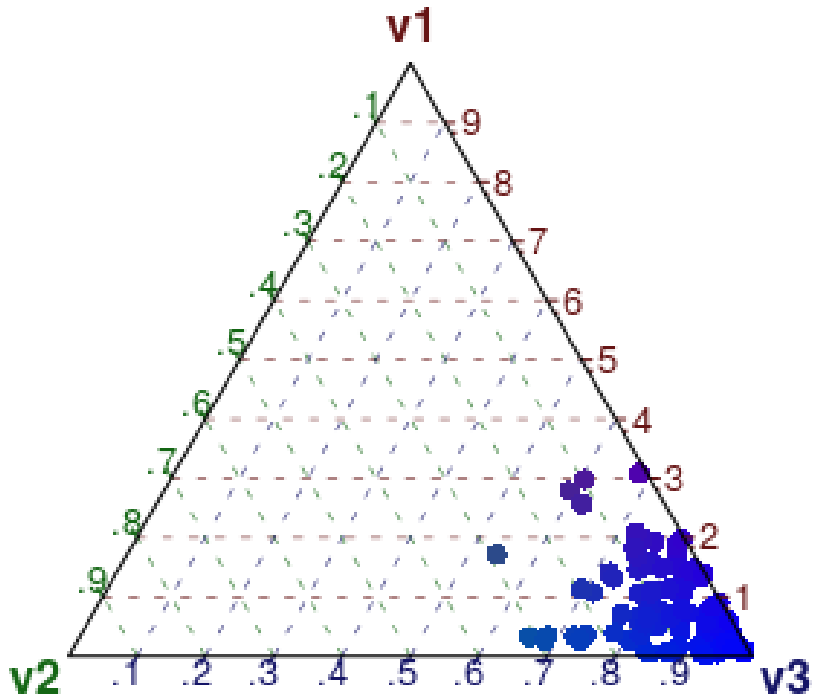
How α influences the Dirichlet? III

Centered Dirichlet $\alpha = (100, 100, 100)$: In this example we set all the α-components to 100, by that we expect that the **chosen points will be more concentrated** around the center of the symplex.



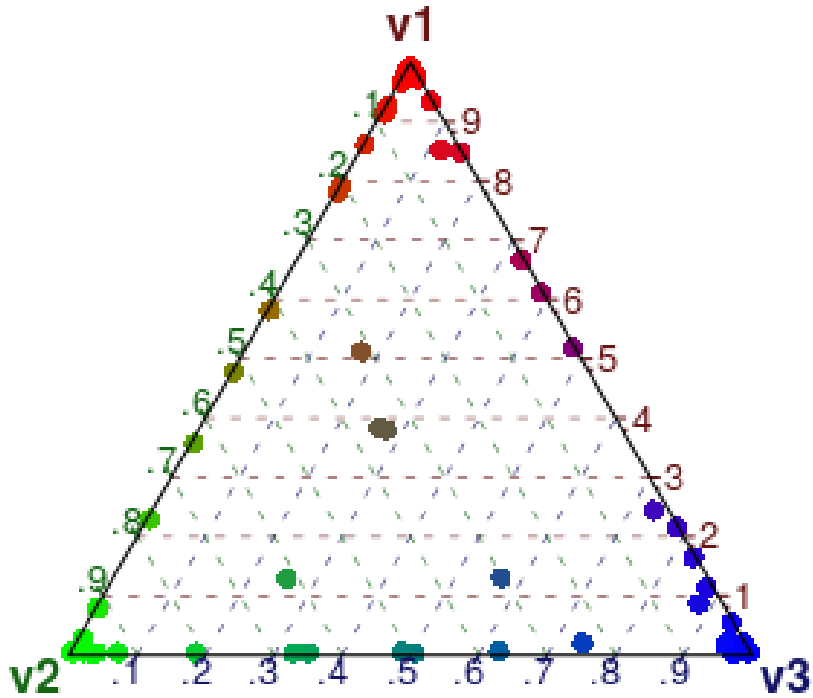
How α influences the Dirichlet? IV

Unfair Dirichlet $\alpha = (1, 1, 10)$: the α-value for the third component is greater than the other one, so we expect that the chosen points will be more **concentrated around the third component** $v3$.



How α influences the Dirichlet? V

Sparse Dirichlet $\alpha = (0.1, 0.1, 0.1)$: but what happens if we set the α-**components** to be **lower than** 1? This is what happens if we set $\alpha = (0.1, 0.1, 0.1)$.



Observations

- When the α-**components** are **greater than one**, the **distribution** tends to be **concentrated around the center** of the symplex.
- On the other hand, with α-**components lesser than one**, the **distribution** tends to be **concentrated around the corners** of the symplex.

Practical consequences

Given a dataset:

- if we suppose (or know) that is more **likely to see only one topic for each document** then we can use α with **components lesser than one** in order to specify this prior knowledge;
- if we suppose that is more **likely that each document contains a mixture of topics** than we will set α with **components greater than one**.

Finding the best number of topics I

There are some strategies to find the best number of topics, clearly the choice of K is in someway driven from the application domain knowledge, that gives us a starting point. Starting from the guessed K we can try different values for it, and then choose the most suitable one by looking at the **topic coherence** and at **quantitative measures**.

Finding the best number of topics II

Looking for the topic coherence of the model is a **human-task**, we have to inspect the distribution of words over topics and decide if from an human point of view they make sense.

The most common quantitative measure used for evaluating the goodness of a topic model, is the **perplexity**, namely a measure of the "surprise" of the model when we give to it new data.

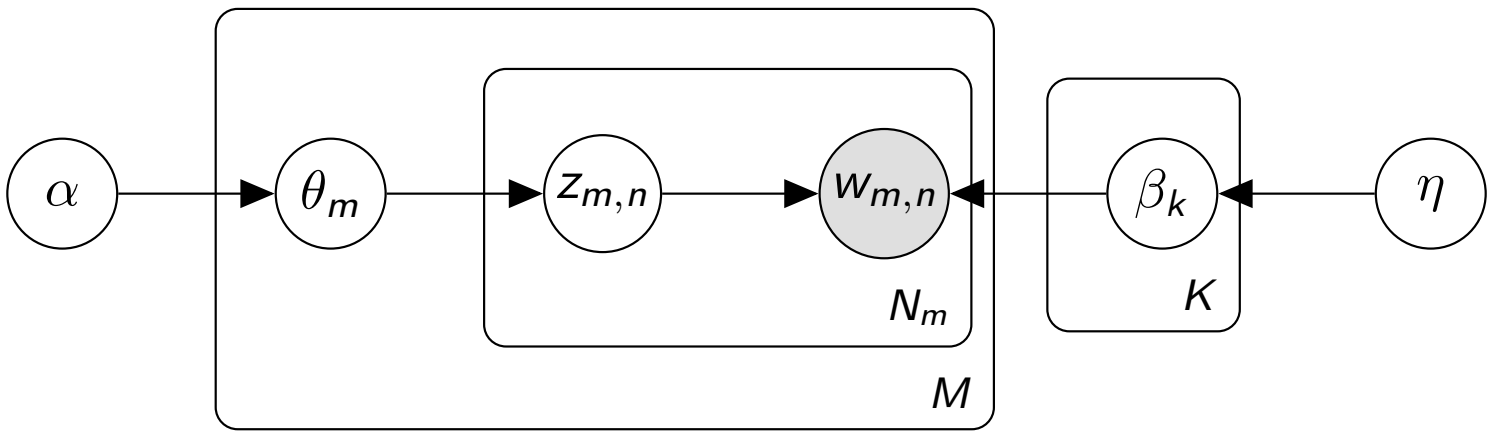
Also plotting the log-likelihood and the perplexity for several values of K can be useful, since a good trade-off between model precision and model length can be find by looking for "bends" in the graph, which are either the optimum or after which the improvements are not so large to justify a complication of the model.

Smoothed LDA I

One problem of our basic LDA model is that, if we've words in the vocabulary that don't appear in any of the training documents then, new documents with unseen vocabulary words will have zero probability of been generated by the model. **Observe** that this problem is similar to the one seen for **PLSA**, in fact the solution is similar.

Smoothed LDA II

In order to solve this problem, we extend the basic model, with a prior on the distributions of words over topics β, this prior will be an **exchangeable Dirichlet distribution**, namely a Dirichlet distribution where all the $\vec{\alpha}$ components α_i are set equal to a scalar η.



Smoothed LDA III

So in this setting, β is treated as a $N \times K$ matrix, where **each column is independently drawn from the exchangeable Dirichlet distribution** with parameter $\vec{\alpha} = (\alpha_1 = \eta, \ldots, \alpha_K = \eta)$.

Considerations I

The basic LDA model is not the usually used one, the **smoothed model is preferred**, **since** we can also specify a prior on the distributions of words over topics and **it can behave better on unseen documents**.

The hyperparameters α and η **can be, estimated** via an EM procedure, **but this is usually not done**, **if we have some domain knowledge** about the distributions of words over topics and topics over documents for our application case **we can specify the hyperparameters consistently**.

Considerations II

- (+): LDA is less prone to overfitting than PLSA since the number of parameters of LDA is strictly less than the number of parameters of PLSA.
- (+): LDA is a pure generative model for documents, while PLSA can not define properly how the topic proportions of documents are generated, LDA overcomes this problem with the dirichlet prior.

Considerations III

- (-): LDA has a number of restrictions due to its assumptions, namely:
  - (-): **BoW assumption**, the order of words in document it is not considered.
  - (-): **Document exchangeability**, if we have a temporal sequence of documents this information is ignored by LDA.
  - (-): **Fixed number of topics**, moreover, we have to specify the number of topics a-priori.

LDA's extensions

Since LDA is formulated as a probabilistic model it can be easily used as a module in more complicated models for more achieving more complicated goals. This can be done for example relaxing an assumption of the LDA generative model.

Relaxing the Bag-of-words assumption

If we want to use LDA for sophisticated goals as **language generation**, then the BoW assumption is not reasonable. *Wallach* presented in 2006 [18] a topic model where instead of a BoW assumption it is assumed that the **topics generate words conditional on the previous word**.

Relaxing the exchangeability of documents assumption

If we have a collection of documents over time (e.g. a collection of newspapers over years), then it is desirable to have that **topics change over time**. This is what **Dynamic topic models** [4], rather than a single distribution over words, a topic is a **sequence of distributions over words**. We can find an underlying theme of the collection and track how **it has changed over time**.

Relaxing the number of topics known and fixed assumption

If we have no idea about the number of topics contained in our collection of documents, then **Hierarchical Dirichlet Processes (HDP)** [19] provides a solution to this problem. In HDP the number of topics is determined by the collection during posterior inference, and furthermore, new documents can exhibit previously unseen topics.

Including metadata

Another way for extending LDA is to incorporate the metadata knowledge about documents, some examples are:

- **author-topic model**: it accounts for the authors of the documents.
- **relational topic model**: it accounts for relationships about the documents in the collection, as for example citation in the case of scientific papers.

Compute similarities on probability distributions

In the probabilistic topic modeling context, we have that the documents are represented as **probability distributions** over topics, and that words are represented as probability distributions over topics as well. Some similarity measures over probability distributions that can be used are:

- **Kullback-Leibler divergence**
- **Jensen-Shannon divergence**

Conclusions on probabilistic topic models

Conclusions

Word embeddings

Recap I

Dimensionality reduction and topic modeling:

- **Topic modeling task**: discover **latent topics** in a collection of documents.
- **Topics**: **latent variables** that capture some **common structure** among the given documents.
- **Topic modeling techniques**: family of **unsupervised methods** for discovering latent topics among a collection of documents.
- **Latent Semantic Analysis (LSA)**: the first topic modeling technique called LSA is based on the **matrix decomposition** of the **term-document matrix** followed by a **dimensionality reduction** step. We have seen how this simple technique helps to bring out similarities between words and documents.

Recap II

Probabilistic topic models:

- **Probabilistic topic models**: they assume a **generative model** for documents, **topics are probability distributions over words** and a **document is a mixture of topics** (hence a probability distribution over topics).
- The output of probabilistic topic models is **interpretable**.
- They are doing **dimensionality reduction** over the word-context probability matrix, the difference with **SVD/LSA** is in the objective function used for obtaining the reconstructed matrix.
- Their **limitations** are given by the **assumptions** made by the generative model considered, however it is possible to extend the generative model by relaxing some of the assumptions, or incorporating some additional information known about the documents that we want to handle.

Word embeddings I

The task of **representing words** is part of almost all the NLP tasks. The first idea for representing words as vectors is taken a vocabulary of words $\mathcal{W} = \{w_1, \ldots, w_N\}$ to encode each word w_n as a N-dimensional vector where all the components are set to zero expect the n-th component that is set to one, this kind of coding for words in $\mathcal{W}$ is called **one-hot representation**. However, this representation has at least two major problems:

1. The dimensionality of the representations linearly increases with the number of words in $\mathcal{W} \Rightarrow$ **curse of dimensionality**
2. Words with similar meanings are distant in the semantic space, moreover their representation are orthogonal $\Rightarrow$ **no similarity**.

Word embeddings II

We would expect that the vector representations for words are such that **words with similar meaning** have similar representations, or said in another way that their representations are **closed in the semantic representation space**.
The concept of similar meaning is defined by the **Distributional hypothesis**, namely, words with similar meanings appear in similar contexts [1].

Word embeddings III

Word embeddings are dense, fixed-length word vectors, built using word co-occurrence statistics as per the distributional hypothesis [1].

Word embeddings IV

We can see the representations for words obtained with topic models as **word embeddings**. They are **fixed-size vector** representations, they are dense and words with similar meanings tends to be close in the semantic space of the representations. For **LSA** the word representations are the word-topic representations, while for **probabilistic topic models** are the probability distributions of topics over words.

Word embeddings V

With the advent of **neural networks** another way of obtaining word embeddings emerged. In particular the first approaches in this sense emerged from the **language modeling task** where the task is to predict the next word in a sentence given the previous words (the context) [17].

Word2Vec - 2013

Word2Vec was created and published in 2013 by a team of researchers led by Tomas Mikolov at Google and patented [13] [14].

Intermediate representations and Deep Learning

We're not going to explain in detail what Deep Learning is, but it's useful to keep in mind that the different layers used in a deep neural network provide **intermediate learned representations** of the raw data fed into the network, that the network uses to do the desired task (prediction, classification or whatever).

Idea

Represent a word by it's meaning, a word's meaning is given by the words that frequently appear in the contexts surrounding it.

Word2Vec considers contexts within a **fixed-size window**, and use the many contexts of a word w_n to build up a K-dimensional representation of it.

Two similarities with LSA

In LSA the meaning of a word is given by the contexts in which appear, where the contexts are the words that co-occur in documents within the considered word.
What we want to do with Word2Vec it is not really different, the **main difference** is in considering contexts within a fixed-size window and not within documents.
Moreover, as in LSA also in Word2Vec we represent **words as vectors in a K-dimensional semantic space**.

Word analogies I

In addition to give a way to compare the meaning of words, the word vectors obtained with Word2Vec show also another interesting behavior: they can be used to find **word analogies** [13].

Word analogies II

Say we want to find the feminine analogous of *king*, we write it as the relation:

$$\text{man} : \text{king} = \text{woman} : x$$

and x is given by:

$$x = \arg\max_i \frac{\left(w_{king} - w_{man} + w_{woman}\right)^T w_i}{\left\| w_{king} - w_{man} + w_{woman} \right\|}$$

We can read it as:

> What is the most similar word to *king* with the *woman* component instead of the *man* component?

It turns out that the result is as expected: *queen*.

Word analogies III

Expression	Nearest token
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android

Description

- Each word is represented by a K-size vector.
- We process a large corpus of text, with T words.
- We have a fixed-size window of radius r.
- For each position $t = 1, \ldots, T$ we focus on w_t that we call **central word** and we consider $w_{t-r}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+r}$ as its context, we call them **context words**.

Training strategies

The idea is to iterate through each word of the corpus and adjust at each iteration the likelihood of having the observed contexts. To do that we can use two different strategies:

- The **Continuous Bag-of-Words** (CBOW) strategy, starting from the context words tries to predict the central word.
- The **Skip-gram** strategy, starting from the central word tries to predict the context words.

CBOW

This is probably the most intuitive strategy, as an example consider this task that corresponds to the **Continuous Bag-of-Words** strategy: given the following context, with window-radus equal to 2, what is the most likely missing word?

Tomorrow, I will ___ the guitar.

CBOW - Network Model

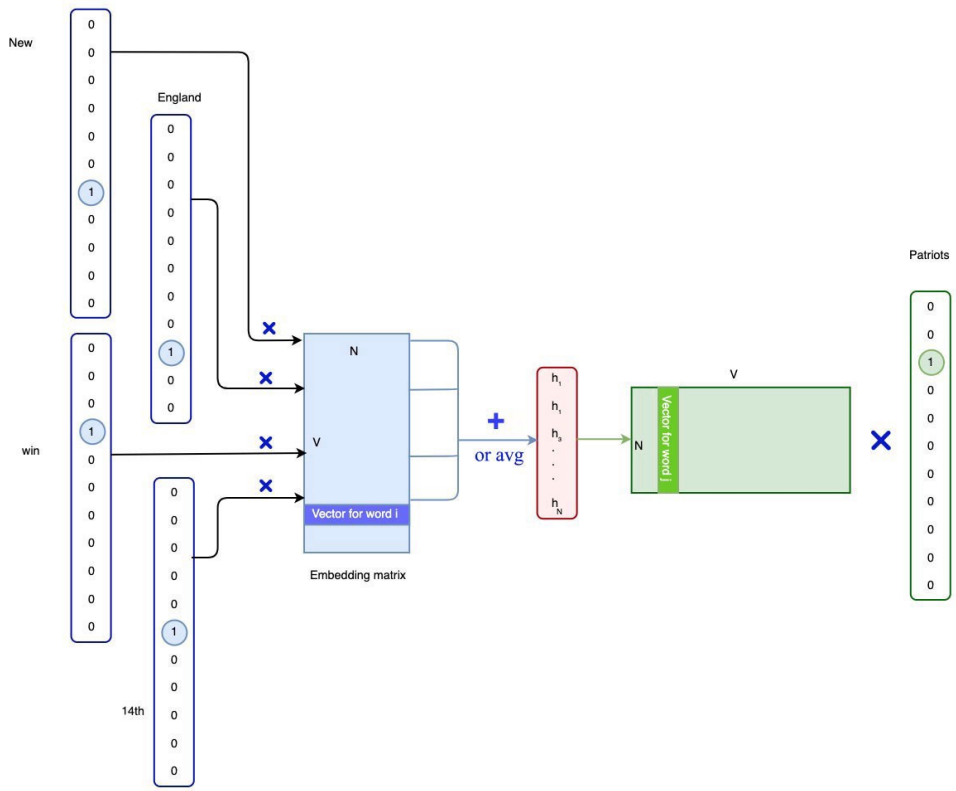


Figure: Graphical representation of the CBOW model

Source: `https://medium.com/@jonathan_hui/nlp-word-embedding-glove-5e7f523999f6`

Skip-gram - 1

What the **Skip-gram** strategy does is the opposite of what it's done by CBOW: we try to predict the surrounding word in a context-window given the central word. Starting from the previous example for CBOW, the Skip-gram version is:

> Given the word play what are the most likely words to surround it?

Skip-gram - 2

The **Skip-gram** strategy may appear counter-intuitive, but it tends to work better than CBOW on **rare words** and this is why this is the preferred strategy to obtain word vectors.

Skip-gram network model

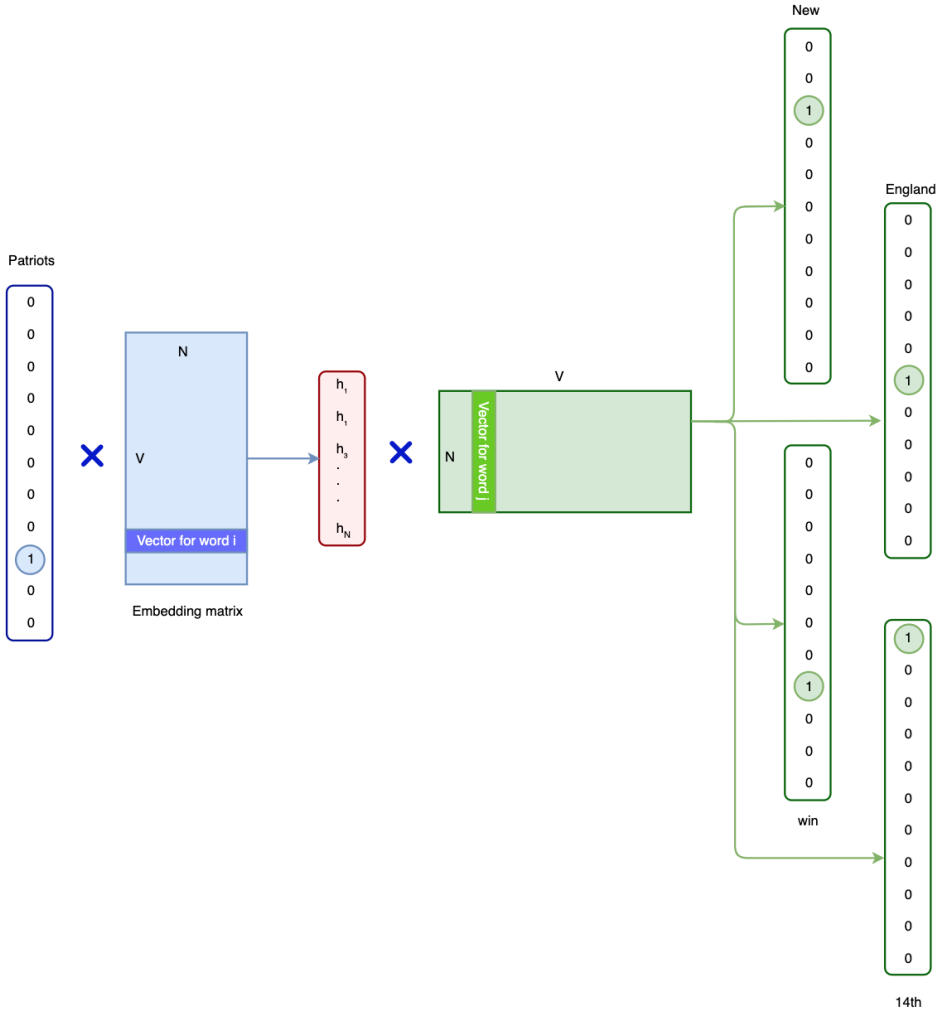


Figure: Graphical representation of the Skip-gram model

Source: https://medium.com/@jonathan_hui/nlp-word-embedding-glove-5e7f523999f6

Likelihood - Skip-gram model

For each position $t = 1, \ldots, T$ of the corpus we want to predict context words within a window of fixed size r, given the central word w_t. By this formulation of the problem, we have the following **likelihood** equation:

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-r \leq j \leq r \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta) \tag{9}$$

where θ is the set of parameters of the model, in particular it contains the word vectors.

Objective function

We consider as **objective function** $J(\theta)$ the average negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-r \leq j \leq r \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta) \tag{10}$$

Clearly minimizing the objective function is equal to maximizing the likelihood. The question is:

$$\text{How to compute } P(w_{t+j} \mid w_t; \theta)?$$

How to compute $P(w_{t+j} \mid w_t; \theta)$? I

We will use two vectors for each word w in the vocabulary:

- v_w when w is the central word, this will be our word vector associated to w.
- u_w when w is a context word.

How to compute $P(w_{t+j} \mid w_t; \theta)$? II

For a central word c and a context word o (o stands for outside) we compute $P(o|c)$ as the **soft-max function** of the dot-product between u_o and v_c:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in \mathcal{W}} \exp(u_w^T v_c)} \tag{11}$$

The dot-product computes the similarity between the two vectors, the assumption is then that similar words have high probability of co-occur within a context.

How to compute $P(w_{t+j} \mid w_t; \theta)$? III

However, this basic formulation of the skip-gram model is **impractical** due to the normalization factor $\sum_{w \in \mathcal{W}} \exp(u_w^T v_c)$, that is over all the words of the vocabulary $\mathcal{W}$ that could be very large. The authors addressed this problem proposing another way of defining $P(o \mid c)$ called **negative sampling** [14], that empirically results in better word embeddings.

Negative sampling I

We define the negative sampling objective for a couple (c, o) seen in the training corpus as:

$$\log P(o \mid c) = \log \sigma(u_o^T v_c) + \sum_{j=1}^{J} \langle \log \sigma(-u_{w_j}^T v_c) \rangle_{w_j \sim P_n(w)} \tag{12}$$

where $\sigma(\cdot)$ is the **standard logistic function**:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

and $P_n(w)$ is a **noise-distribution** over the words in $\mathcal{W}$.

Negative sampling II

So the skip-gram with negative sampling objective function task is to **distinguish** the real context-word o from J **draws from the noise distribution** $P_n(w)$ using **logistic-regression**. The authors state that good results can be obtained setting the number of negative samples J in the range $5 - 20$ for small training sets, and in the range $2 - 5$ for large training sets. A common choice for the noise distribution $P_n(w)$ is the **unigram distribution** $U(w)$ raised to the $3/4$ power.

Subsampling of frequent words I

In order to accelerate the training phase and at the same time improve the quality of the obtained word embeddings, another technique adopted by word2vec is the **subsampling of frequent words** [14].

Subsampling of frequent words II

The idea is that frequent words (eg. 'the', 'a', 'and,' ...) usually provide **less information** than the rare words, and moreover, their representations do **not change significantly after training on several examples**.

Subsampling of frequent words III

In order to address for this intuition, a **subsampling approach** is used: each word w_i in the training set **is discarded** with probability:

$$P(w_i) = 1 - \sqrt{\frac{t}{\#w_i}} \tag{14}$$

where t (es. 10^{-5}) is a chosen threshold.

SGNS as a matrix factorization I

We can see the **skip-gram with negative sampling** (SGNS) objective function as the factorization of an implicit matrix $M \in \mathbb{R}^{N \times N}$ each cell $M_{o,c}$ of M can be seen as representing a quantity $f(o, c)$ that represents the strength of the association between the context-word pair (o, c) [12]. But what can we say about $f(o, c)$? In other words, what matrix is implicitly being factorized by SGNS?

SGNS as a matrix factorization II

Levy and Goldberg [12] shown that an entry of the matrix M implictly factorized is:

$$M_{o,c} = u_o^T v_c = PMI(o, c) - \log J \tag{15}$$

Where $PMI(o, c)$ is called **point-wise mutual information** and is defined as:

Definition 4.1. Point-wise mutual information $PMI(o, c)$ Let o, c be two discrete outcomes from a probability distribution P, the **point-wise mutual information** between o and c is defined as:

$$PMI(o, c) = \log \frac{P(o, c)}{P(o)P(c)} \tag{16}$$

SGNS as a matrix factorization III

PMI can be estimated empirically observing the actual number of observations in the training corpus:

$$PMI(o, c) \approx \log \frac{\#(o, c) * N}{\#(o)\#(c)} \tag{17}$$

SGNS as a matrix factorization IV

The matrix M where each entry $M_{o,c} = PMI(o, c) - \log J$ is named **shifted PMI matrix** M^{PMI_J}. So, **SGNS's objective** can be casted as the problem of finding the best K-dimensional reconstruction of the matrix M^{PMI_J} under a metric which **pays more for deviation under frequent** (o, c) **pairs than deviations on infrequent pairs** [12]. SGNS does this using a **stochastic gradient descent** procedure.

Decomposing M^{PMI_J} with SVD I

Instead of using stochastic gradient descent we can think of applying **SVD decompositon** on the M^{PMI_J} matrix and keeping only the first K principal singular values/vectors in order to find a best K-dimensional reconstruction of M^{PMI_J}, however notice that, for a couple (o, c) that does not appear in the corpus we have $M^{PMI_J}_{o,c} = -\infty$

Decomposing M^{PMI_J} with SVD II

At this purpose, instead of using M^{PMI_J} we can use M^{SPPMI_J}, that uses as association metric the **shifted positive PMI** $SPPMI_k$:

$$SPPMI_J(o, c) = \max(PMI(o, c) - \log J, 0) \tag{18}$$

However, it is worth noting that we are **losing information about negative associations** for pairs (o, c) where o and c are frequent in the corpus and appear rarely together, since in such case we would have a negative value for $PMI_J(o, c)$.

Decomposing M^{PMI_J} with SVD III

Performing **SVD** on M^{SPPMI_J} and taking ony the K most relevant singular values/vectors we obtain:

$$M_K^{SPPMI_J} = U_K \Sigma_K V_K^T$$

that is the best possible rank K approximation of M^{SPPMI_J} wrt the 2-norm and the Frobenius-norm. We then could define:

$$\begin{aligned} V^{SVD} &= U_K \Sigma_K \\ U^{SVD} &= V_K \end{aligned} \tag{19}$$

where V^{SVD} is the word embeddings matrix and U^{SVD} is the context embedding matrix.

SVD vs. SGNS I

SVD advantages:

- it is **exact**, and does **not require learning rates** or hyperparameter tuning.
- while SGNS require each observation (o, c) to be presented indepently, SVD can be trained on **count-aggregated data**, for example triplets $(o, c, \#(o, c))$.

SVD vs. SGNS II

SGNS advantages:

- SGNS weights different (o, c) differently, it prefer to assign correct values to frequent (o, c) pairs while allowing more error in unfrequent (o, c) pair!
- since SGNS cares only about observed (and samples) (o, c) pairs, it can optimize the exact M^{PMI_J} matrix, that can not be done with SVD as we have seen.

It turns out that for this motivation SGNS obtains **better word embeddings** than using the SVD decomposition on the M^{SPPMI_J} matrix, in particular the SGNS embeddings perform better on the word analogy task.

Main difference between Word2Vec and LSA

We have seen a lot of similarities between Word2Vec and LSA, however, they have a main difference in the information that they use to build word vectors:

- Word2Vec uses only **local information** of the language: the semantics for a given word is affected only by its surrounding words, this lead to best word vectors for analogy task, but it poorly use the statistics of the corpus.
- LSA uses only **global statistics** of a corpus, namely the global co-occurrence counts. It uses efficiently the statistical information, but the obtained word vectors do relatively poorly on the word analogy task.

GloVe - 2014 I

In 2014 Jeffrey Pennington et al. at the University of Stanford proposed **GloVe** [15] a method that **combines** the best of the **global matrix factorization approach** (e.g. SVD) and of the **local context window approach** (e.g. Word2Vec).

GloVe - 2014 II

GloVe firstly build a **window-based co-occurrence matrix** of the corpus, and then it's trained only on the nonzero elements of that matrix, rather than on the entire sparse matrix (as done for LSA) or on an individual context windows in a large corpus (as done in Word2Vec). Moreover, GloVe seems to work better than Word2Vec on word analogy task.

How word vectors are evaluated?

Typically word vectors are used in a larger model where they serve as an intermediate representation of words. Then, we have **two types of evaluation** for word vectors:

How word vectors are evaluated?

Typically word vectors are used in a larger model where they serve as an intermediate representation of words. Then, we have **two types of evaluation** for word vectors:

- **Intrinsic evaluation** is the evaluation of a set of word vectors on **specific intermediate subtasks** (such as word analogies). This evaluation is typically simple and fast to compute.

How word vectors are evaluated?

Typically word vectors are used in a larger model where they serve as an intermediate representation of words. Then, we have **two types of evaluation** for word vectors:

- **Intrinsic evaluation** is the evaluation of a set of word vectors on **specific intermediate subtasks** (such as word analogies). This evaluation is typically simple and fast to compute.
- **Extrinsic evaluation** is the evaluation of a set of word vectors on the **real task at hand**. It's clearly slower than the intrinsic evaluation, morevoer if the system don't perform as expected nothing guarantees that the problem are the used word vectors.

Considerations - 1

Word2Vec and GloVe are **useful methods to build vector representations of words**. A strength of these representations if compared with **LSA** (or LDA) is that it's possible to find pre-trained on billion tokens word vectors that are ready to be used on personal complex systems. Another strength is that these representations allow us to find word analogies.

Considerations - 2

The weakness of these representations is that they have a unique representation of a word independently by the context in which appear, we recall that the word vector representation obtained with these models is given by the many context in which the word appears.

ELMo - 2018

ELMo stands for **"Embeddings from Language Models"**, it was presented in 2018 by Peters et al. from the Allen institute for AI [16].

Goal

ELMo borns from the needing of have **contextual word embeddings**, for instance consider the word *hang* in the following sentences:

"*Hang* your coat"
"Wanna *hang* out tomorrow?"

Clearly, *hang* has different meanings in these sentences.
ELMo's goal is to **give different embeddings to words depending on the context in which they appear**.

Bidirectional language modeling task I

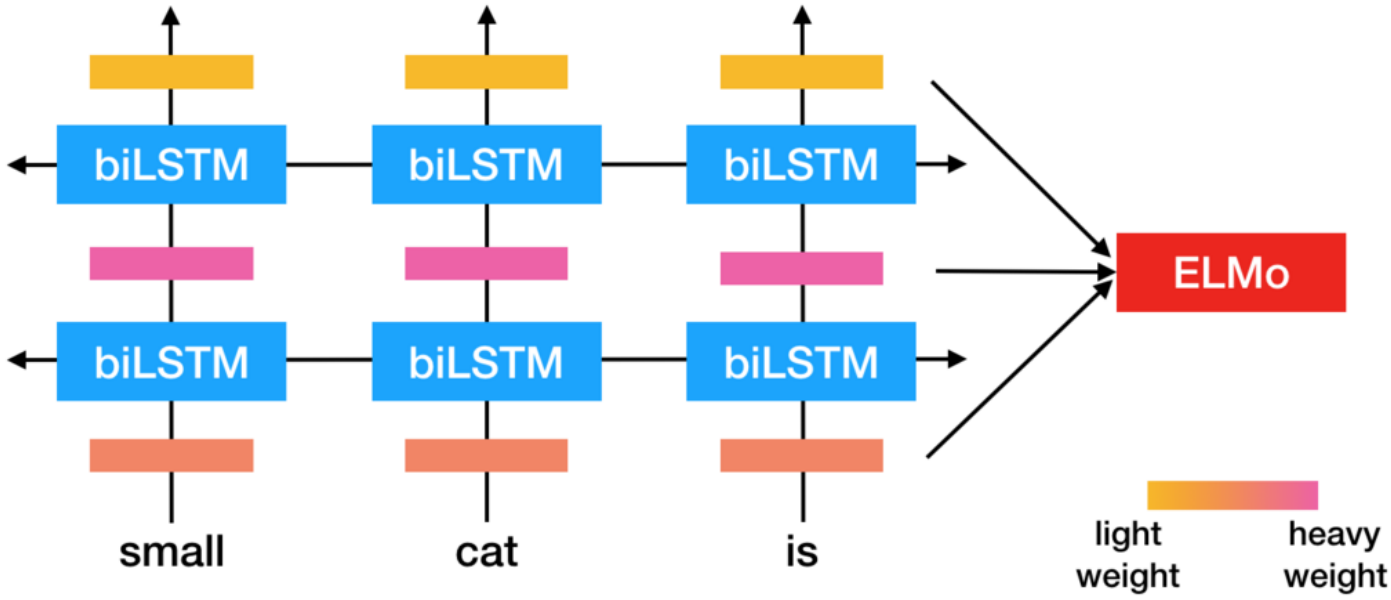
ELMo builds contextual representations of words by looking at the entire sentences in which they appear. This is done considering the **bidirectional language modeling task**, namely find a missing word w_t reading from left-to-right all the preceding words $w_1, \ldots, w_T$ and reading in the opposite direction all the following words $w_T, w_{T-1}, \ldots, x_{t+1}$. If we consider a sentence $x_1, \ldots, x_T$, the likelihood is then:

$$\begin{aligned} L(\theta) =& p(x_1, \ldots, x_T) \\ =& \prod_{t=1}^{T} p(x_t | x_1, x_2, \ldots, x_{t-1}) p(x_t | x_T, x_{T-1}, \ldots, x_{t+1}) \end{aligned} \tag{20}$$

Bidirectional language modeling task II

The bidirectional language modeling task is typically approached with **Recurrent Neural Networks**. RNNs have an hidden state for each word seen in the sentence; **the hidden state for a word is influenced by the states of the previously considered words in the sentence**. Considering how RNNs work, it seems natural to see **the associated hidden state for a word as a contextual representation** for it. ELMo uses a specific kind of RNNs called **biLSTM** (bidirectional Long-Short Term Memory).

ELMo network model



Source:
`https://kamujun.hatenablog.com/entry/2018/07/06/173931`

ELMo embeddings I

Looking at the ELMo network model we can see how contextual word embeddings are build:

1. The first layer of the network builds raw word vector representations of the sentence's words using a **character-level** convolutional neural network (CNN).
2. The second layer is a biLSTM which hidden states are **syntactic-level** representations of the sentence's words.
3. The last layer is another biLSTM which hidden states are **semantic-level** representations of the sentence's words.

How ELMo does it? - 4

Then what ELMo does is to take the **three different representations** of a specific word and combine them together as a **weighted sum** to form the context specific embedding for the considered word.

$$ELMO_k^{task} = \gamma^{task} \sum_{l=1}^{3} s_l^{task} \mathrm{h}_{k,l} \tag{21}$$

The entire ELMo vector is scaled by γ^{task}.

How ELMo representations are used?

Typically, **pre-trained biLM** are taken and integrated inside a supervised architecture for a NLP task (e.g. classification). Then we simply run the biLM to generate the three representations for each word and we let the end task model learn the scaling parameter γ^{task} and the weights $s_l^{task}, s = 1, 2, 3$. This process is called **fine-tuning**.

Why a weighted sum?

The weight given at each layer representation to form the contextual embedding depends on the task for which we want to use them. There may be tasks for which we want to give more focus on the semantical-aspects and other tasks for which we want to give more focus on the syntactic-aspects.

Considerations

ELMo representations overcome the main problem of classical word embedding methods as Word2Vec and GloVe: the lack of specific contextual representation of words. This solution can give good performance improving in many NLP tasks, and the price to pay if pre-trained solution on the language at hand are available is not too high since we have only to do a fine tuning on the specific task that we are considering. ELMo however is not the state-of-the-art for contextual word embeddings, more recent methods as **BERT** [6] are.

A Simple But Tough-to-Beat Baseline for Sentence Embeddings - 2017

We will give the idea behind an **unsupervised method for building sentence embeddings starting from word embeddings** proposed in 2017 by Sanjeev Arora et al. [2] from Princeton University.

Idea

The proposed method is another point of contact between embeddings and SVD/PCA. The main idea is to assume a **probabilistic generative model** for sentences, where each sentence has an underlying **common discourse** that is related to the grammar and that we can estimate as the **first principal component** of the considered sentences. Thus, to obtain more meaningful representations for the considered sentences we will remove the common discourse for each represented sentence.

Considerations - 1

It turns out that this simple unsupervised method works very well on **text similarity task**, moreover it can be applied to any types of word embeddings and it should be used when we lack of training data.

Considerations - 2

The sentence embeddings obtained by the presented method can be used as features in supervised tasks, where it turns out that for some tasks (e.g. similarity task, entailment task) they perform as well as sentence embeddings obtained with supervised methods (e.g. RNN/LSTMs). However there are tasks for which this unsupervised method doesn't perform as well as RNNs/LSTMs, this is the case for example of the **sentiment analysis task** where the authors assume that this is due to the "antonym problem" hence as an example, the downweighting of words like "not" because of their frequency can be not a good idea.

Summary

Summary: Dimensionality reduction and topic modeling I

We've started seeing two **algebraic methods**, **PCA** and **SVD**, which are most commonly used to do **dimensionality reduction** on data. The process of dimensionality reduction can help to **bring out significant relationships** between data while **removing menaningless information** (noise). We've also see how this two methods are tied between them.

Summary: Dimensionality reduction and topic modeling II

From the idea of exploiting SVD to do dimensionality reduction on **term-document matrices** in 1997 borns by Landauer et al. **LSA**, the first method that we've seen with some ideas of **topics/latent variables**. LSA apply **dimensionality reduction** on the term-document matrix associated with a corpus to bring out meaningful relationships and similarities between words and documents, fundamentally we set a K-dimensional semantic space where each dimension represent a "topic" and in which we project our data. The reconstructed term-document matrix on the K-dimensional space is a **least-squares best fit** for the original term-document matrix.

Summary: Dimensionality reduction and topic modeling III

- (+): easy to apply.
- (+): it's possible to exploit the natural notion of **cosine similarity** for computing term-term and document-document similarities.
- (-): the obtained document-topic and term-topic matrices are **not interpretable**.

Probabilistic topic modeling I

We have given an overview on probabilistic topic modeling, showing the main idea of representing **topics as probability distributions over words** and **documents as mixtures of topics**; and the idea of defining a **generative model for documents**. We have seen the advantages of this approach, mainly the **interpretability** of the results and how this approach can handle **polysemy**. Moreover, we have seen how to train this kind of models giving an overview on the **EM algorithm** and the **MCMC sampling approach**.

Probabilistic topic modeling II

Starting from the main idea of LSA of representing documents by topics, giving to this idea a more formal definition using **probability distributions** and a **probabilistic generative model** for documents, Thomas Hofmann in 1999 proposes **PLSA**.

Probabilistic topic modeling III

PLSA makes two assumptions on the data under hand. The first is the **bag-of-words** assumption, namely, the order in which words occur in a document is unrelevant with respect to the document-meaning. The second is the **conditional independence** assumption namely the probability of seeing a word in a document given a topic depends only by the topic.

Probabilistic topic modeling IV

We have seen the **connection** between PLSA and **LSA**, in fact, PLSA objective can be seen as finding a **matrix decomposition in a K-dimensional space that best reconstructs the probability term-document matrix** estimated by the training corpus.

Probabilistic topic modeling V

- (+): PLSA offers **interpretable results**, overcoming the major problem of LSA.
- (+): It offers a more **solid statistical basis** than LSA, defining a generative model for documents.

Probabilistic topic modeling VI

- (-): it is **prone to overfitting**, since the number of parameters of the model increase linearly with the size of the training corpus.
- (-): PLSA does not make any assumptions about how the topic proportions are generated, making it **difficult to test the generalizability** of the model to new documents.
- (-): there is **no natural way of assigning topic proportions to unseen documents**.

Probabilistic topic modeling VII

Assuming PLSA as a good starting point for topic modeling and finding how to overcome the problem of assigning topic distributions for unseen documents, David Blei et al. propose in 2003 **Latent Dirichlet Allocation**.

Probabilistic topic modeling VIII

To overcome the major problem of PLSA, LDA assumes that documents are represented as **random mixtures over latent topics**, where topics as for PLSA are distributions of probability over words. The random mixtures of topics for each document are drawn from a **Dirichlet distribution** that is a distribution over distributions of probability which can be influenced by a K-dimensional vector parameter α which specifies what the expected value of the distribution is, and how much sparse it is.

Probabilistic topic modeling IX

The commonly used version of LDA assumes that also topics are drawn from a dirichlet distribution, this is done to **prevent problems due to unseen words** in the training data, clearly also this dirichlet distribution can be influenced by its own parameter η.

Probabilistic topic modeling X

LDA inherits all the assumptions and the good aspects of PLSA, in particular its interpretability, it adds also some interesting features as the **possibility of specifying some domain-knowledge** (e.g. how likely to be the distributions of topics in documents are) to the model that can be result in more coherent topics.

Probabilistic topic modeling XI

- (+): LDA is **less prone to overfitting** than PLSA since the number of parameters of LDA is strictly less than the number of parameters of PLSA.
- (+): LDA is a **pure generative model for documents**, while PLSA can not define properly how the topic proportions of documents are generated, LDA overcomes this problem with the dirichlet prior.

Probabilistic topic modeling XII

- (-): LDA has a number of **restrictions due to its assumptions**, namely:
  - (-): **BoW assumption**, the order of words in document it is not considered.
  - (-): **Document exchangeability**, if we have a temporal sequence of documents this information is ignored by LDA.
  - (-): **Fixed number of topics**, moreover, we have to specify the number of topics a-priori.

Probabilistic topic modeling XIII

Finally we have seen how it's possible to obtain other topic models for addressing different kind of problems by **relaxing some of the assumptions** of the LDA generative model, or by **extending** it in order to take in account some additional information that we have about the documents we are trying to model.

Summary: Word Embeddings I

In the last part of this cycle of seminars we have given an overview on the task of **representing words** as fixed-size vectors and a definiton of what **word embeddings** are. We have noticed that the representations obtained with **topic models** for words can be seen as word embeddings as well.

Summary: Word Embeddings II

We have given an overview of **Word2Vec**, a neural method for obtaining word embeddings based on the **distributional hypothesis** as the topic models seen in precedence.

Summary: Word Embeddings III

The main idea is to project words in a K-dimensional space, but differently from LSA we do not consider the co-occurrence of words within entire documents, but we consider the **co-occurrence of words within a fixed-size window.** This turns out to give the possibility of compute **word analogies** as well as **word similarities**.

Summary: Word Embeddings IV

Moreover we have seen how it is possible to train a word2vec model, and how the **Skip-gram with negative sampling** training method is related to an implicit matrix factorization done with the stochastic gradient descent algorithm.

Summary: Word Embeddings V

Giving a comparison between LSA and Word2Vec, we have seen how **GloVe** combines the best of the two methods in order to obtain better vector representations of words with respect to Word2Vec. Finally, we have highlighted the main problem of this **word embeddings**, the **lack of contextual-dependent representation for words**.

Summary: Word Embeddings VI

- (+): possibility of computing **word analogies** and similarities.
- (+): possibility of using **pre-trained** word embeddings.
- (-): lack of contextual-dependent representation for words.

Summary: Word Embeddings VII

In 2018 by the idea if Peters et al. of build embeddings from language models to overcome the main problem of classical word embeddings borns **ELMo**. The important thing to know about ELMo is that it is build over the idea that the hidden state associated with a word in a **biLSTM** is a contextual-representation for the considered word. Moreover, ELMo builds **three different representations** for each word that captures three different aspects of it like the syntactic and the semantic, then the final ELMo representation is a **weighted sum of the three different representation** for it.

Summary: Word Embeddings VIII

If the available computing power allow the training of the bidirectional language-model, ELMo it is surely a better choice than the classical word embeddings technique. Moreover, if the available computing power is not amazing, if a pre-trained biLM is available for the desired language it is possible to do only an operation of **fine-tuning**, to find the weights to give at each representation by running the system in which we embedd the ELMo model. However, ELMo is **not the state-of-the-art** for obtaining contextual word embeddings, more recent methods as **BERT** obtain better performances on almost all NLP tasks.

Summary: Word Embeddings IX

- (+): contextual-dependent representation for words.
- (+): possibility of using **pre-trained** biLM and do only an operation of **fine-tuning**.
- (-): not the state-of-the-art.

Summary: Word Embeddings X

Finally, we've seen a simple unsupervised method proposed in 2017 by Sanjeev Arora et al. to build sentence embeddings starting from word embeddings. The method builds the embedding for a sentence as a weighted sum of the embeddings of the words that make up the sentence and then remove from them the first principal component that is assumed to be a **common discourse part** in which we can find the syntactic rule of the grammar and is related to the high frequent words.

Summary: Word Embeddings XI

- (+): **unsupervised** method, when we lack of training data it could be a good choice.
- (-): there are **tasks** as sentiment analysis for which using embeddings obtained with this method give **poor performances**.

References I


Felipe Almeida and Geraldo Xexéo.
Word embeddings: A survey.
CoRR, abs/1901.09069, 2019.


Sanjeev Arora, Yingyu Liang, and Tengyu Ma.
A simple but tough-to-beat baseline for sentence embeddings.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.


David M. Blei.
Probabilistic topic models.
Commun. ACM, 55(4):77–84, 2012.

References II


David M. Blei and John D. Lafferty.
Dynamic topic models.
In William W. Cohen and Andrew W. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 113–120. ACM, 2006.


David M. Blei, Andrew Y. Ng, and Michael I. Jordan.
Latent dirichlet allocation.
J. Mach. Learn. Res., 3:993–1022, 2003.


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
BERT: pre-training of deep bidirectional transformers for language understanding.
CoRR, abs/1810.04805, 2018.

References III


Karl Pearson F.R.S.
Liii. on lines and planes of closest fit to systems of points in space.
The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11):559–572, 1901.


Thomas Hofmann.
Probabilistic latent semantic analysis.
In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 289–296. Morgan Kaufmann, 1999.


Harold Hotelling.
Relations between two sets of variates.
Biometrika, 28(3/4):321–377, 1936.


Thomas K Landauer, Peter W. Foltz, and Darrell Laham.
An introduction to latent semantic analysis.
Discourse Processes, 25(2-3):259–284, 1998.

gap
IN RELATION WE TRUST

References IV

Dumais S. T. Landauer T. K.
A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.
Psychological Review, 104(2):211–240, 1997.

Omer Levy and Yoav Goldberg.
Neural word embedding as implicit matrix factorization.
In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185, 2014.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

gap
IN RELATION WE TRUST

References V

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean.
Distributed representations of words and phrases and their compositionality.
In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning.
Glove: Global vectors for word representation.
In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014.

References VI


Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer.
Deep contextualized word representations.
CoRR, abs/1802.05365, 2018.


Erhan Sezerer and Selma Tekir.
A survey on neural word embeddings.
CoRR, abs/2110.01804, 2021.


Hanna M. Wallach.
Topic modeling: beyond bag-of-words.
In William W. Cohen and Andrew W. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 977–984. ACM, 2006.

References VII

Matthew J. Beal Yee Whye Teh, Michael I. Jordan and David M. Blei.
Hierarchical dirichlet processes.
Journal of the American Statistical Association, 101(476):1566–1581, 2006.