Master Degree in Computer Science

Information Retrieval

Topic Modeling

Prof. Alfio Ferrara

Department of Computer Science, Università degli Studi di Milano
Room 7012 via Celoria 18, 20133 Milano, Italia alfio.ferrara@unimi.it

sed noli modo

Motivations for latent semantics discovery

Two problems are typical of Vector Space Models:

High dimensionality of vector representation of documents (that are also sparse).

Synonymy. Many terms have the same meaning but corresponds to different dimensions in the vector space representation. This typically leads to poor recall.

Ambiguity. The same term (same vector dimension) may have different meanings.

Example:

Napoleon dominated European and global affairs for more than a decade while leading France against a series of coalitions in the Napoleonic Wars.	Napoleon was born in Corsica to a relatively modest family from minor Italian nobility.	Napoleon is located in the northwestern corner of Lafayette County and bordered on the north side by the Missouri River and on the south side by U.S. Route 24 and on the west by Jackson County.	He won most of these wars and the vast majority of his battles, building a large empire that ruled over continental Europe before its final collapse in 1815.	It is approximately 40 miles east of the Kansas/ Missouri line on the south side of the Missouri River and about 6 miles east of the Independence city limits as of this writing.

Latent Semantic Indexing

The intuition of LSI (sometimes called Latent Semantic Analysis, LSA) is to discover topics or latent concepts that motivate data even when data use a different terminology for expressing a topic by exploiting matrix factorization techniques.

Quick recap on eigenvalues and eigenvectors

Let *A* be an $n \times n$ matrix with elements being real numbers. If $\overrightarrow{x}$ is an n-dimensional vector, then the matrix-vector product $A\overrightarrow{x}$ is well-defined, and the result is again an n-dimensional vector.

In general, multiplication by a matrix changes the direction of a non-zero vector $\overrightarrow{x}$, unless the vector is special and we have that

$$A\overrightarrow{x} = \lambda\overrightarrow{x}$$

λ is an **eigenvalue**

$\overrightarrow{x}$ is an **eigenvector**

Latent Semantic Indexing

Assuming $A\vec{x} = \lambda\vec{x}$, then $(A - \lambda I)\vec{x} = 0$ where *I* is the identity matrix

Since $\vec{x}$ is non zero, then $|A - \lambda I| = 0$

In general, for a $n \times n$ matrix, $|A - \lambda I| = 0$ has *n* solutions leading to *n* eigenvalues

According to this result, we can create a matrix using the eigenvalues as follows:

$$AX = A[\vec{x}_1, \ldots, \vec{x}_n] = A\vec{x}_1 + \ldots + A\vec{x}_n$$

$$= \lambda_1\vec{x}_1 + \ldots + \lambda_n\vec{x}_n = [\vec{x}_1, \ldots, \vec{x}_n]\begin{bmatrix} \lambda_1 & & \\ & \cdots & \\ & & \lambda_n \end{bmatrix} = [\vec{x}_1, \ldots, \vec{x}_n]\Lambda = X\Lambda$$

where Λ is the diagonal matrix of the eigenvalues and $X = [\vec{x}_1, \ldots, \vec{x}_n]$

Latent Semantic Indexing

If the *n* eigenvalues are distinct values, then the *n* eigenvectors are linearly independent

Recall that a square matrix *M* is invertible if there exists a square matrix *B* such that $MB = BM = I$. *B* is then the inverse of *M*, denoted M^{-1}

$$AX = X\Lambda \rightarrow AXX^{-1} = X\Lambda X^{-1} \rightarrow A = X\Lambda X^{-1}$$

This representation of *A* is called **diagonalization** of *A*

If *A* is symmetric ($A = A^T$), we have

$$A = X\Lambda X^T$$

Moreover, $n \times n$ symmetric matrices always have real eigenvalues and their eigenvectors are perpendicular

Singular Value Decomposition

Consider a term/document matrix $A_{m\times n}$ with $m > n$. Take $B_{n\times n} = A^T A$.

$$B^T = (A^T A)^T = A^T (A^T)^T = A^T A = B$$

We sort the eigenvalues of *B* and take $\sigma_i^2 = \lambda_i$ for each of the first *k* non-zero ones, so that $V = [\vec{v}_1, \ldots, \vec{v}_k]$. We also define $U = [\vec{u}_1 = \frac{1}{\sigma_1} A\vec{v}_1, \ldots, \vec{u}_k = \frac{1}{\sigma_k} A\vec{v}_k]$ that are perpendicular *m*-dimensional vectors of length 1 ($\vec{u}_i$ are normalized to 1). We have then:

$$\vec{u}_j^T A\vec{v}_i = \vec{u}_j^T(\sigma_i \vec{u}_i) = \sigma_i \vec{u}_j^T \vec{u}_i \quad \text{That is} \quad U^T AV = \Sigma \rightarrow A = U\Sigma V^T$$

$\Sigma_{k\times k}$ is the diagonal matrix having $\sigma_1, \ldots, \sigma_k$ along the diagonal. U^T is $k \times m$, *A* is $m \times n$, *V* is $n \times k$

Singular Value Decomposition

In practice, the matrices *U*, *Σ*, and *V* can be found by transforming *A* in a square matrix and by computing the eigenvectors of this square matrix.

The square matrix is obtained by multiplying the matrix *A* by its transpose. In particular:

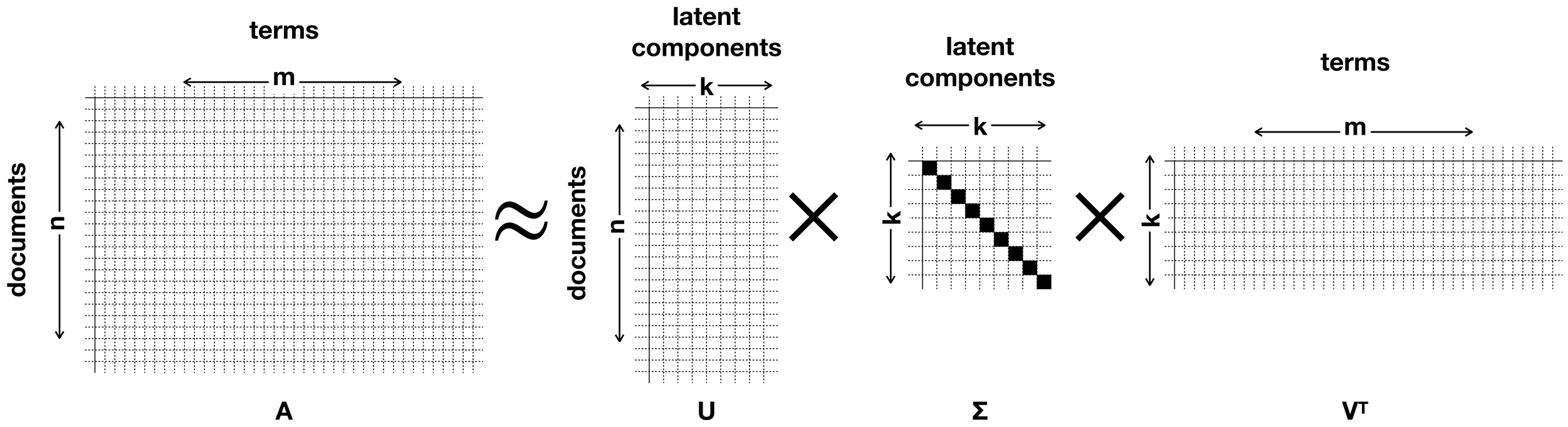
- U is defined by the **eigenvectors** of AA^T
- V is defined by the **eigenvectors** of A^TA
- Σ is the **diagonal matrix composed starting from the eigenvalues** of AA^T and A^TA (which are the same values)

Note that the eigenvalues are involved in two matrix products, that's why we define **the singular values** as $\sigma_i = \sqrt{\lambda_i}$

If the take a number of singular values lower than *n*, SVD becomes a way of reducing the dimensionality of the original matrix.

Singular Value Decomposition

Interpretation of SVD for latent semantic indexing



Latent Semantic Indexing

Recalling that the documents in the new space are represented by the row vectors of *U* whereas the terms by the column vectors of V^T, having $U\Sigma V^T$, we obtain the matrix $\mathscr{D}^{n\times k}$ of documents with respect to latent topics and the matrix $\mathscr{T}^{k\times m}$ of terms as follows:

$$\mathscr{D}^{n\times k} = U\Sigma \qquad\qquad \mathscr{T}^{k\times m} = \Sigma V^T$$

Query: Get the columns $R = r_i, \ldots, r_j$ of $\mathscr{T}$ corresponding to the terms $i, \ldots, j$ in the query; Get the query vector $\overrightarrow{q}$ as the mean of R; Calculate cosine similarity of $\overrightarrow{q}$ agains $\mathscr{D}$

Soft clustering: The absolute value of $\mathscr{T}_{ij}$ provides the relevance of term *j* for the topic *i*. The same for $\mathscr{D}$.

Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is a generative probabilistic model for collections of discrete data such as text corpora.

LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics.

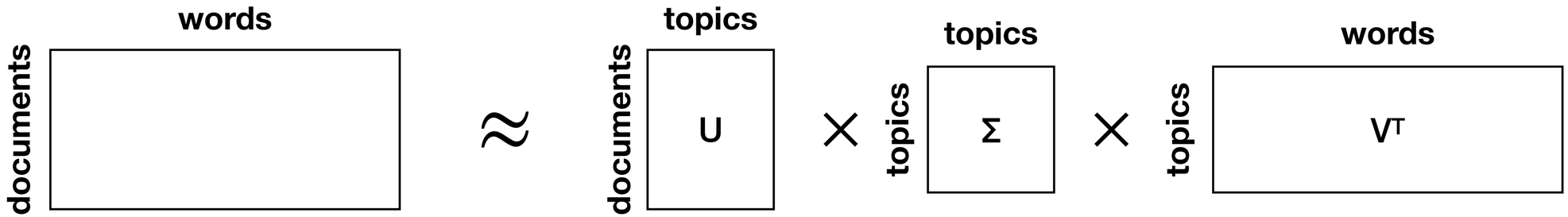
Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. “Latent dirichlet allocation." the Journal of machine Learning research 3 (2003): 993-1022.

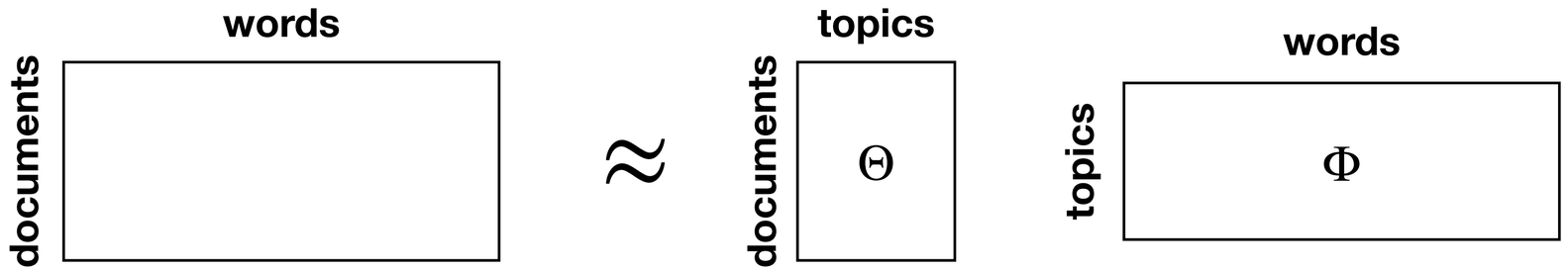
Latent Dirichlet allocation

Comparison with LSI

LSI



LDA



Latent Dirichlet allocation

Underlying generative model

$\phi^{(k)}$ Is a discrete **probability distribution over the vocabulary** for the *k*th topic

θ_d Is a **document distribution** over the topics

z_i Is the **topic index** for the word w_i

α and β are the **hyperparameters** for the generation of the **Dirichlet distributions**

For $k = 1, \ldots, K$ $\quad \phi^{(k)} \sim Dir(\beta)$

For $d \in D$ $\quad \theta_d \sim Dir(\alpha)$

Generate each word $w_i \in d$

$$z_i \sim Discrete(\theta_d)$$

$$w_i \sim Discrete(\phi^{(z_i)})$$

$$Dir(x, \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)} \qquad \Gamma(\alpha) = (\alpha - 1)!$$

Latent Dirichlet allocation

Derivation of the joint distribution

$$p(w, z, \theta, \phi \mid \alpha, \beta) = p(\phi \mid \beta)p(\theta \mid \alpha)p(z \mid \theta)p(w \mid \phi_z)$$

Where, starting from data, we need to estimate z, ϕ, θ.

θ_d is a representation of *d* in the topic space

z_i represents which topic generated the word instance w_i

Each $\phi^{(k)}$ is a matrix $K \times W$ where $\phi_{ij} = p(w_i \mid z_j)$

Posterior inference

In order to find the latent variables, we need to solve

$$p(\theta, \phi, z \mid w, \alpha, \beta) = \frac{p(\theta, \phi, z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)}$$

Latent Dirichlet allocation

Gibbs sampling

Gibbs Sampling is one member of a family of algorithms from the Markov Chain Monte Carlo (MCMC) framework

MCMC algorithms aim to construct a Markov chain that has the target posterior distribution as its stationary distribution. In other words, after a number of iterations of stepping through the chain, sampling from the distribution should converge to be close to sampling from the desired posterior.

Example: to sample from $p(X) = p(x_1, \ldots, x_n)$ we can:

- Randomly initialize *X*
- For each iterative step *t*:
  - $x_1^{t+1} \sim p(x_1 \mid x_2^t, x_3^t, \ldots x_n^t)$
  - $x_2^{t+1} \sim p(x_2 \mid x_1^{t+1}, x_3^t, \ldots x_n^t)$
  - ...
  - $x_n^{t+1} \sim p(x_n \mid x_1^{t+1}, x_2^{t+1}, \ldots x_{n-1}^{t+1})$

Latent Dirichlet allocation

Gibbs sampling for LDA

We need to estimate $\phi^{(k)}, \theta_d, z_i$. But note that:

where *n(d, z)* is the number of words in *d* assigned to *z*

$$\theta_{d,z_i} = \frac{n(d, z_i) + \alpha}{\sum_{j \in Z} n(d, z_j) + \alpha}$$

$$\phi_{z,w_i} = \frac{n(z, w_i) + \beta}{\sum_{j \in W} n(z, w_j) + \beta}$$

where *n(z, w)* is the number of times word *w* is assigned to topic *z*

Thus, we just need to estimate z_i. This simpler version is called **collapsed Gibbs sampling**

We estimate then:

$$p(z_i \mid z_{\neg i}, \alpha, \beta, w),$$

where $z_{\neg i}$ represents all topic allocations except for z_i

Latent Dirichlet allocation

Collapsed Gibbs sampling

We have

$$p(z_i \mid z_{\neg i}, \alpha, \beta, w) = \frac{p(z_i, z_{\neg i}, w \mid \alpha, \beta)}{p(z_{\neg i}, w \mid \alpha, \beta)} \propto p(z_i, z_{\neg i}, w \mid \alpha, \beta) = p(z, w \mid \alpha, \beta)$$

$$p(z, w \mid \alpha, \beta) = \iint p(z, w, \phi, \theta \mid \alpha, \beta) d\phi d\theta$$

Latent Dirichlet allocation

Collapsed Gibbs sampling

$$p(z, w \mid \alpha, \beta) = \iint p(z, w, \phi, \theta \mid \alpha, \beta) d\phi d\theta$$

According to the LDA model, we can

$$p(z, w \mid \alpha, \beta) = \iint p(\phi \mid \beta) p(\theta \mid \alpha) p(z \mid \theta) p(w \mid \phi_z) d\phi d\theta$$

and group dependent variables

$$p(z, w \mid \alpha, \beta) = \int p(z \mid \theta) p(\theta \mid \alpha) d\theta \int p(w \mid \phi_z) p(\phi \mid \beta) d\phi$$

that are two multinomial distributions with Dirichlet priors.

Latent Dirichlet allocation

In LDA the Dirichlet distribution is chosen as the conjugate prior for the multinomial distributions of docs/words given the topics.

Recall that, given $p_1, \ldots, p_n$ the probabilities determining the multinomial distribution are:

$$(p_1, \ldots, p_n) \sim Dir(\alpha_1, \ldots, \alpha_n)$$

as the priors. Then, given the topics $z_1, \ldots, z_n$

$$(p_1, \ldots, p_n) \mid (z_1, \ldots, z_n) \sim Dir(\alpha_1 + z_1, \ldots, \alpha_n + z_n)$$

Latent Dirichlet allocation

As a further tool for solving the integrals, we use the **gamma** and **beta functions** that are used in the definition of the Dirichlet distribution PDF

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \quad ; \quad B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1} dt$$

With:

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

We can extend this definition as:

$$B(\alpha_1, \alpha_2, \ldots, \alpha_n) = \frac{\Gamma(\alpha_1) \cdot \Gamma(\alpha_2) \cdot \ldots \cdot \Gamma(\alpha_n)}{\Gamma(\alpha_1 + \alpha_2 + \ldots + \alpha_n)}$$

$$\text{In } \mathbb{N}: \Gamma(n) = (n-1)! \; ; \; B(n,m) = \frac{(n-1)!(m-1)!}{(n+m-1)!}$$

Latent Dirichlet allocation

The probability density function of the Dirichlet distribution is:

$$f(x_1, \ldots, x_n; \alpha_1, \ldots, \alpha_n) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1},$$

With:

$$B(\alpha) = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{n} \alpha_i\right)}, \alpha = (\alpha_1, \ldots \alpha_n)$$

Latent Dirichlet allocation

Back to Collapsed Gibbs sampling:

$$p(z, w \mid \alpha, \beta) = \int p(z \mid \theta) p(\theta \mid \alpha) d\theta \int p(w \mid \phi_z) p(\phi \mid \beta) d\phi$$

Left part

$$\int p(z \mid \theta) p(\theta \mid \alpha) d\theta = \int \prod_i \theta_{d,z_i} \frac{1}{B(\alpha)} \prod_k \theta_{d,k}^{\alpha_k} d\theta_d$$

$$= \frac{1}{B(\alpha)} \int \prod_k \theta_{d,k}^{n_{d,k}+\alpha_k} d\theta_d = \frac{B(n_{d,.} + \alpha)}{B(\alpha)}$$

Right part

$$\int p(w \mid \phi_z) p(\phi \mid \beta) d\phi = \int \prod_d \prod_i \phi_{z_{d,i},w_{d,i}} \prod_k \frac{1}{B(\beta)} \prod_w \phi_{k,w}^{\beta_w} d\phi_k$$

$$= \prod_k \frac{1}{B(\beta)} \int \prod_w \phi_{k,w}^{\beta_w + n_{k,w}} d\phi_k = \prod_k \frac{B(n_{k,.} + \beta)}{B(\beta)}$$

where $n_{d,k}$ is the number of times words of *d* are assigned to *k* ($n_{d,.}$ is the sum over *k*)

Latent Dirichlet allocation

Back to Collapsed Gibbs sampling:

$$p(w, z \mid \alpha, \beta) = \prod_d \frac{B(n_{d,.} + \alpha)}{B(\alpha)} \prod_k \frac{B(n_{k,.} + \beta)}{B(\beta)}$$

Now we can estimate assignment z_i using the chain rule (we leave out α and β for clarity):

$$p(z_i \mid z^{\neg i}, w) = \frac{p(w, z)}{p(w, z^{\neg i})} = \frac{p(z)}{p(z^{\neg i})} \cdot \frac{p(w \mid z)}{p(w^{\neg i} \mid z^{\neg i}) p(w_i)}$$

$$\propto \prod_d \frac{B(n_{d,.} + \alpha)}{B(n_{d,.}^{\neg i} + \alpha)} \prod_k \frac{B(n_{k,.} + \beta)}{B(n_{k,.}^{\neg i} + \beta)} \qquad \propto (n_{d,k}^{\neg i} + \alpha_k) \frac{n_{k,w}^{\neg i} + \beta_w}{\sum_{w'} n_{k,w'}^{\neg i} + \beta_{w'}}$$