Text Mining — Assignment #2

Roger Cuscó, Matthew-Sudmann-Day and Miquel Torrens

Exercise 1

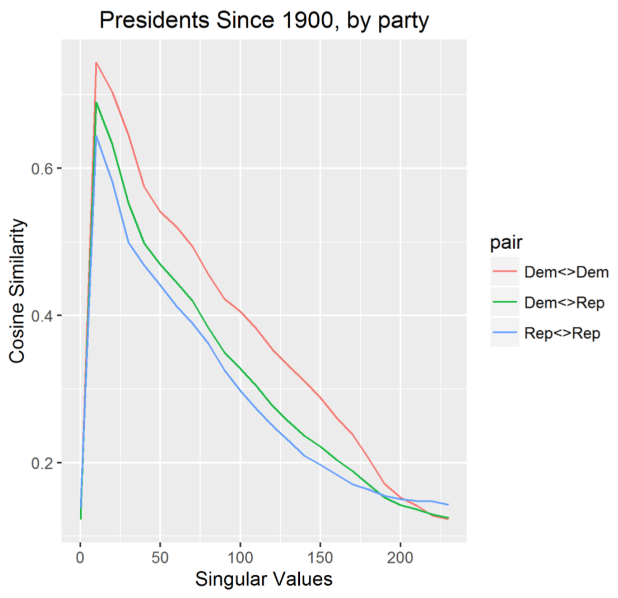
The python code for this exercise can be found in the `week2_hw.py` file.

We constructed a TF-IDF matrix for the full corpus of the State of the Union speeches dataset. We extracted subsets of those documents based on characteristics that we believe could affect the content. Then we performed a cosine similarity comparison between documents within each subset and between the two subsets. This will tell us whether or not the characteristics we have chosen for categorizing the documents are reflected in term frequencies.

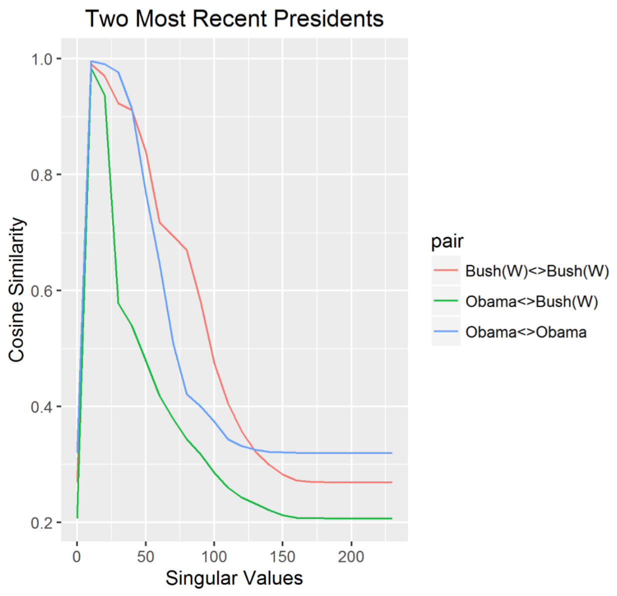
We can perform the above analysis directly as described or we can reduce noise in the data by performing a singular value decomposition (SVD), replacing the smaller singular values with zeros, and then recomposing. Selecting a "reasonable"number of singular values to retain is not necessarily easy so we plotted results against a variety of singular values, in increments of 10.

On the plots below, the x-axis shows the number of singular values retained. Zero was preserved as a special value meaning that no SVD was performed so the analysis was explicitly done against the original TF-IDF matrix in that one case. One can easily observe, unsurprisingly, that when we retrain **all** singular values, the result is the same as if the SVD was not performed at all. This explains why the left edge and the right edge of the plots look identical.

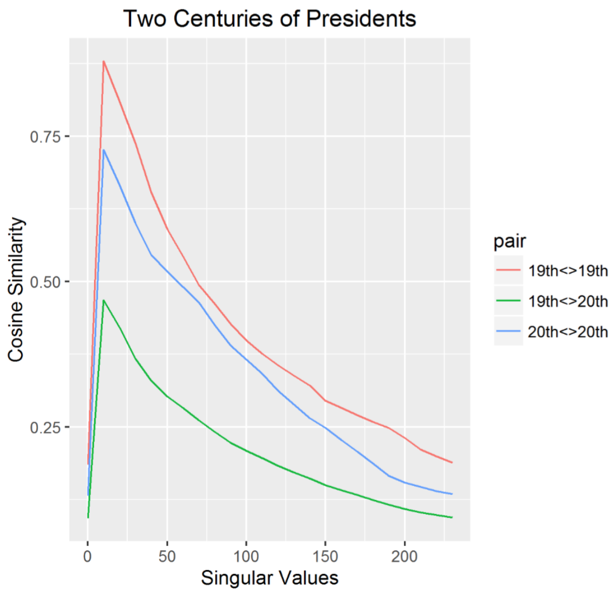
First, observe a breakdown by party. As you can see, the number of retained singular values changes the conclusions but only at the extreme. As long as we eliminate a few singular values, we eliminate enough noise. And we conclude that SoU speeches of Republicans are more similar to each other than those of Democrats. But they are both more similar to each other than they are to speeches from presidents of the other party.



For those who have been alive in the last 16 years, the follow is a very intuitive result is below: speeches of Bush (W) have a character that is easily identified as different from those of Obama. However, retaining too many or too few singular values will tell us that Obama's speeches more resemble each other than Bush's do. Retaining a medium or "reasonable" number will tell us the opposite.



Finally, the language and politics evolves enough over the centuries to easily separate 19th century and 20th century speeches, regardless of the party of the president.



Exercise 2

Part (a)

The components of the model are the following:

- Parameters:
  - $\boldsymbol{\rho} = (\rho_1, \dots, \rho_k, \dots, \rho_K) \in \mathbb{R}^K$
  - $\mathbf{B}^{(1)} = \left(\beta^{(1)}_{z_1}, \dots, \beta^{(1)}_{z_i}, \dots, \beta^{(1)}_{z_N}\right)$, where $\beta^{(1)}_{z_i} \in \Delta^{V_1 - 1}, \forall i$
  - $\mathbf{B}^{(2)} = \left(\beta^{(2)}_{z_1}, \dots, \beta^{(2)}_{z_i}, \dots, \beta^{(2)}_{z_N}\right)$, where $\beta^{(2)}_{z_i} \in \Delta^{V_2 - 1}, \forall i$

  For notational purposes, let $\mathbf{B} = \{\mathbf{B}^{(1)}, \mathbf{B}^{(2)}\}$.

- Latent variables:
  - $\mathbf{z} = (z_1, \dots, z_i, \dots, z_N) \in \mathbb{R}^N$, for $i \in \{1, \dots, N\}$ observations

- Observed data:
  - $\mathbf{x}^{(1)}_i = \left(x^{(1)}_{i1}, \dots, x^{(1)}_{iv_1}, \dots, x^{(1)}_{iV_1}\right) \in \mathbb{R}^{V_1}, \forall i \in \{1, \dots, N\}$
  - $\mathbf{x}^{(2)}_i = \left(x^{(2)}_{i1}, \dots, x^{(2)}_{iv_2}, \dots, x^{(2)}_{iV_2}\right) \in \mathbb{R}^{V_2}, \forall i \in \{1, \dots, N\}$

  For notational purposes, let $\mathbf{x}_i = \left\{\mathbf{x}^{(1)}_i, \mathbf{x}^{(2)}_i\right\} \forall i$ and $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$.

Part (b)

The density function is:

$$\begin{aligned}
\mathbb{P}(\mathbf{x}_i|\boldsymbol{\rho}, \mathbf{B}) &= \mathbb{P}\left(\mathbf{x}^{(1)}_i, \mathbf{x}^{(2)}_i|\boldsymbol{\rho}, \mathbf{B}\right) \\
&= \mathbb{P}\left(\mathbf{x}^{(1)}_i|\boldsymbol{\rho}, \mathbf{B}\right)\mathbb{P}\left(\mathbf{x}^{(2)}_i|\boldsymbol{\rho}, \mathbf{B}\right) \qquad \text{(By independence of the two vectors)} \\
&= \sum_{k=1}^{K} \mathbb{P}(z_k|\boldsymbol{\rho}, \mathbf{B})\,\mathbb{P}\left(\mathbf{x}^{(1)}_i|z_k, \boldsymbol{\rho}, \mathbf{B}\right)\mathbb{P}\left(\mathbf{x}^{(2)}_i|z_k, \boldsymbol{\rho}, \mathbf{B}\right) \\
&= \sum_{k=1}^{K} \rho_k \mathbb{P}\left(\mathbf{x}^{(1)}_i|z_k, \boldsymbol{\rho}, \mathbf{B}\right)\mathbb{P}\left(\mathbf{x}^{(2)}_i|z_k, \boldsymbol{\rho}, \mathbf{B}\right) \\
&= \sum_{k=1}^{K} \rho_k \prod_{v_1=1}^{V_1}\left(\beta^{(1)}_{z_i v_1}\right)^{x^{(1)}_{iv_1}} \prod_{v_2=1}^{V_2}\left(\beta^{(2)}_{z_i v_2}\right)^{x^{(2)}_{iv_2}}.
\end{aligned}$$

The likelihood function is the joint probability of all $\mathbf{x}_i$'s:

$$\ell(\mathbf{x}|\boldsymbol{\rho}, \mathbf{B}) = \prod_{i=1}^{N}\sum_{k=1}^{K} \rho_k \prod_{v_1=1}^{V_1}\left(\beta^{(1)}_{z_i v_1}\right)^{x^{(1)}_{iv_1}} \prod_{v_2=1}^{V_2}\left(\beta^{(2)}_{z_i v_2}\right)^{x^{(2)}_{iv_2}}.$$

The complete likelihood incorporates the latent variable and writes as follows:

$$\ell_{\text{comp}}(\mathbf{x}, \mathbf{z}|\boldsymbol{\rho}, \mathbf{B}) = \prod_{i=1}^{N}\prod_{k=1}^{K}\left[\rho_k \prod_{v_1=1}^{V_1}\left(\beta^{(1)}_{z_i v_1}\right)^{x^{(1)}_{iv_1}} \prod_{v_2=1}^{V_2}\left(\beta^{(2)}_{z_i v_2}\right)^{x^{(2)}_{iv_2}}\right]^{\mathbb{1}_{\{z_i = k\}}}.$$

Finally, the complete log-likelihood gives:

$$\log \ell_{\text{comp}}(\mathbf{x}, \mathbf{z}|\boldsymbol{\rho}, \mathbf{B}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}_{\{z_i=k\}} \left[\log \rho_k + \sum_{v_1=1}^{V_1} x_{iv_1}^{(1)} \log \beta_{z_i v_1}^{(1)} + \sum_{v_2=1}^{V_2} x_{iv_2}^{(2)} \log \beta_{z_i v_2}^{(2)} \right].$$

Part (c)

Denote the function Q as:

$$Q(\boldsymbol{\rho}, \mathbf{B}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \hat{z}_{i,k} \left[\log \rho_k + \sum_{v_1=1}^{V_1} x_{iv_1}^{(1)} \log \beta_{z_i v_1}^{(1)} + \sum_{v_2=1}^{V_2} x_{iv_2}^{(2)} \log \beta_{z_i v_2}^{(2)} \right], \tag{1}$$

where using the Bayes' rule:

$$\begin{aligned} \hat{z}_{i,k} \equiv \mathbb{E}\left[\mathbb{1}_{\{z_i=k\}} | \boldsymbol{\rho}, \mathbf{B}, \mathbf{x}_i\right] &= \mathbb{P}\left(z_i = k | \boldsymbol{\rho}, \mathbf{B}, \mathbf{x}_i\right) \\ &\propto \mathbb{P}\left(\mathbf{x}_i | \boldsymbol{\rho}, \mathbf{B}, z_i = k\right) \mathbb{P}\left(z_i = k | \boldsymbol{\rho}, \mathbf{B}\right) \\ &= \rho_k \prod_{v_1=1}^{V_1} \left(\beta_{z_i v_1}^{(1)}\right)^{x_{iv_1}^{(1)}} \prod_{v_2=1}^{V_2} \left(\beta_{z_i v_2}^{(2)}\right)^{x_{iv_2}^{(2)}}. \end{aligned} \tag{2}$$

Part (d)

The Lagrangian is defined as:

$$\mathcal{L}(\boldsymbol{\rho}, \mathbf{B}, \boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}, \nu) := Q(\boldsymbol{\rho}, \mathbf{B}) + \nu \left(1 - \sum_{k=1}^{K} \rho_k\right) + \sum_{k=1}^{K} \lambda_k^{(1)} \left(1 - \sum_{v_1=1}^{V_1} \beta_{z_i v_1}^{(1)}\right) + \sum_{k=1}^{K} \lambda_k^{(2)} \left(1 - \sum_{v_2=1}^{V_2} \beta_{z_i v_2}^{(2)}\right).$$

We take derivatives with respect to the parameters:

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \rho_k} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{N} \frac{\hat{z}_{i,k}}{\rho_k} - \nu = 0 \tag{3}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_{k,v_1}} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{N} \hat{z}_{i,k} x_{iv_1}^{(1)} \frac{1}{\beta_{z_i v_1}^{(1)}} - \lambda_k^{(1)} = 0 \tag{4}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_{k,v_1}} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{N} \hat{z}_{i,k} x_{iv_2}^{(2)} \frac{1}{\beta_{z_i v_2}^{(2)}} - \lambda_k^{(2)} = 0 \tag{5}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \nu} = 0 \quad \Leftrightarrow \quad \sum_{k=1}^{K} \rho_k = 1 \tag{6}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \lambda_k^{(1)}} = 0 \quad \Leftrightarrow \quad \sum_{v_1=1}^{V_1} \beta_{z_i v_1}^{(1)} = 1 \tag{7}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \lambda_k^{(2)}} = 0 \quad \Leftrightarrow \quad \sum_{v_2=1}^{V_2} \beta_{z_i v_2}^{(2)} = 1. \tag{8}$$

Combining (3) and (6) we see:

$$\nu = \frac{1}{\rho_k} \sum_{i=1}^{N} \hat{z}_{i,k} \Leftrightarrow \nu \underbrace{\sum_{k=1}^{K} \rho_k}_{=1} = \sum_{k=1}^{K} \sum_{i=1}^{N} \hat{z}_{i,k} \underbrace{\Leftrightarrow}_{\text{Plug } \nu \text{ into (3)}} \rho_k^* = \frac{\sum_{i=1}^{N} \hat{z}_{i,k}}{\sum_{k=1}^{K} \sum_{i=1}^{N} \hat{z}_{i,k}}, \tag{9}$$

Analogously, combining (4) and (7) we get:

$$\beta^{(1)}_{z_i v_1}\lambda^{(1)}_k = \sum_{i=1}^{N}\hat{z}_{i,k}x^{(1)}_{iv_1} \Leftrightarrow \underbrace{\sum_{v_1=1}^{V_1}\beta^{(1)}_{z_i v_1}}_{=1}\lambda^{(1)}_k = \sum_{i=1}^{N}\hat{z}_{i,k}\sum_{v_1=1}^{V_1}x^{(1)}_{iv_1} \underbrace{\Leftrightarrow}_{\text{Plug } \lambda^{(1)}_k \text{ into (4)}} \left(\beta^{(1)}_{z_i v_1}\right)^* = \frac{\sum_{i=1}^{N}\hat{z}_{i,k}x^{(1)}_{iv_1}}{\sum_{i=1}^{N}\hat{z}_{i,k}\sum_{v_1=1}^{V_1}x^{(1)}_{iv_1}}. \quad (10)$$

The exact same procedure can be applied with (5) and (8) (simply changing subscripts) to obtain:

$$\beta^{(2)}_{z_i v_1}\lambda^{(2)}_k = \sum_{i=1}^{N}\hat{z}_{i,k}x^{(2)}_{iv_2} \Leftrightarrow \underbrace{\sum_{v_2=1}^{V_2}\beta^{(2)}_{z_i v_2}}_{=1}\lambda^{(2)}_k = \sum_{i=1}^{N}\hat{z}_{i,k}\sum_{v_2=1}^{V_2}x^{(2)}_{iv_2} \underbrace{\Leftrightarrow}_{\text{Plug } \lambda^{(2)}_k \text{ into (5)}} \left(\beta^{(2)}_{z_i v_2}\right)^* = \frac{\sum_{i=1}^{N}\hat{z}_{i,k}x^{(2)}_{iv_2}}{\sum_{i=1}^{N}\hat{z}_{i,k}\sum_{v_2=1}^{V_2}x^{(2)}_{iv_2}}. \quad (11)$$

The parameters found in this step, i.e. the set of ρ^*_k's, $\left(\beta^{(1)}_{z_i v_1}\right)^*$'s and $\left(\beta^{(2)}_{z_i v_2}\right)^*$'s will be the ones used in the next iteration.

Part (e)

Pseudo-code for the EM algorithm in this context:

```
#################################################################################
em = function(X, K, eps = 1e-6, maxiter = 1e5) {
# X: input data, with word counts for the N observations
# K: number of clusters of the latent variable
# eps: value that defines the distance at which the algorithm stabilizes
# maxiter: maximum number of iterations allowed before convergence
#################################################################################
  # Number of observations
  N = length(X)

  # Initialize parameters
  for (i in 1:N) {
    beta1[i] = [1 / v1, ..., 1 / V1]  # length V1
    beta2[i] = [1 / v2, ..., 1 / V2]  # length V2
  }
  B1 = [beta1[1], ..., beta1[N]]  # matrix with betas1
  B2 = [beta2[1], ..., beta2[N]]  # matrix with betas2
  rho = [1 / K, ..., 1 / K]  # length K

  # Perform iteration until convergence
  iter = 1
  repeat {
    # Expectation step
    zhat = compute[Equation (2); X, rho, B1, B2]
    Q = compute[Equation (1); X, rho, B1, B2]

    # Maximization step
    for (k in 1:K) {
      rho[k] = sum(zhat[1:N, k]) / sum(zhat[1:N, 1:K])  # Equation (9)
    }
    for (i in 1:N) {
      # Equation (10)
      for (v1 in 1:V1) {
        beta1[i, v1] = sum(zhat[1:N, k] * X[1:N, v1]) /
                       sum(zhat[1:N, k] * X[1:N, 1:V1])
      }

      # Equation (11)
      for (v2 in 1:V2) {
        beta1[i, v2] = sum(zhat[1:N, k] * X[1:N, v2]) /
                       sum(zhat[1:N, k] * X[1:N, 1:V2])
      }
    }

    # Update all parameters
    B1 = [beta1[1], ..., beta1[N]]  # matrix with betas1
    B2 = [beta2[1], ..., beta2[N]]  # matrix with betas2
    rho = [1 / K, ..., 1 / K]  # length K

    # If the parameters have stabilized, we stop
    if (norm(B1) < eps & norm(B2) < eps & norm(rho) < eps) {
      converged = TRUE
      break
    }

    # If the algorithm has not stabilized after lots of iterations, stop
    if (iter > maxiter) {
      converged = FALSE
      break
    }

    # Update iterator
    iter = iter + 1
  }

  # End
  return(zhat, B1, B2, rho, converged)
}
```