# Clustering bilingual text corpora using mixtures of von Mises-Fisher distributions

**With an application to the corpus of abstracts of the Austrian Journal of Statistics**

Master Thesis

**Dominik Ernst**

February 21, 2013

Supervision:
Prof. Dr. Thomas Augustin
Dr. Bettina Grün
Dr. Manuel Eugster

Department of Statistics
Ludwig–Maximilians–University
Munich

# Contents

## Abstract

With the increase of availability of text corpora on the internet and specifically bilingual text corpora, the objective of this thesis is to develop an extension to the movMF model by Banerjee et al. (2005). This model was implemented using two algorithms, the EM algorithm and an annealing variant called the DAEM algorithm, which often yielded better results.

With the aim of analyzing the corpus of abstracts of the Austrian Journal of Statistics (AJS), the automated retrieval, extracting and processing the abstracts from the journal website was developed and explained. Also simulation studies were conducted that showed that the DAEM algorithm usually outperforms the EM algorithm except for simple cases where both algorithms showed similar performance. Further they showed that problems with the same number of observations and amount of dimensions as the AJS corpus were generally very difficult for both algorithms. Simulation also showed that the model estimates deteriorate if less than half the documents are available in both languages.

For the analysis of the AJS corpus two different schemes to reduce the dimensions were employed as well as a reweighting of the data-sets. However in neither case the models produced a descriptive result, in contrast to the equivalent unilingual model.

# Chapter 1

# Introduction

Document clustering describes the process of grouping a set of documents, commonly referred to as a text corpus, into clusters such that similar documents are assigned to the same cluster and dissimilar documents are assigned to different clusters. One type of models addressing this task are "bag of words" models, where each document is described by the number of occurrences of each word alone while disregarding the order in which they occured. A notable example is the *spherical k-means* algorithm (Dhillon and Modha, 2001). In this model documents are encoded as $L_2$-normalized vectors of their term-frequencies and thus each document is represented as a point on the $d$-dimensional unit hypersphere, where $d$ is the number of different terms in a corpus. Term frequencies being positive in nature, these documents then lie in the upper orthant of $\mathbb{R}^d$. The spherical k-means algorithm is then a variation of the "euclidean" k-means algorithm (Duda and Hart, 1973) using the cosine similarity as similarity measure.

Another general approach to clustering are finite mixture models where each observation is said to be generated from any of the mixture components. Banerjee et al. (2005) propose a mixture model where each component is said to follow a von Mises-Fisher (vMF) distribution. This distribution is often parameterized by a mean direction $\mu$ and a concentration parameter $\kappa$, the latter corresponding to the inverse of the variance. This mixture model (movMF) is related to spherical k-means in such a way that if $\kappa$ is assumed to approach infinity in each component, the movMF model can be shown to maximise the cosine similarity of each observation to a cluster representative (Banerjee et al., 2005, section 5.1) and thus being in essence equivalent to the spherical k-means algorithm (for $\kappa_h \to \infty$).

Both the spherical k-means algorithm and the movMF model assume that the documents in a corpus stem from the same language. Corpora containing documents in different translations are not uncommon however.

Bilingual corpora considered in the following consist of documents in two different languages and are parallel at the document-level, meaning that one pair of documents in different languages correspond to one another. Examples for such corpora could be constructed from the online encyclopedia Wikipedia, where articles are often translated into different languages. Other bilingual corpora can be constructed from the proceedings of the European Parliament, which are published on the Internet (Koehn, 2005). The OPUS project (Tiedemann, 2012) lists more examples of parallel corpora including the website and documentation of the European Central Bank and the documentation of the PHP programming language (Tiedemann, 2009).

The aim of this thesis is to develop an extension of the movMF model called the bilingual mixture of von Mises-Fisher [distributions] model (bimovMF) to accomodate bilingual text corpora. Chapter 2 gives a brief introduction to von Mises-Fisher distributions and mixtures thereof as described by Banerjee et al. (2005). Chapter 3 goes on to describe the bilingual mixture model detailing parameter estimation and the estimation algorithms as well as some implementation details. As an application of this model chapter 6 examines the corpus of abstracts of the Austrian Journal of Statistics (AJS) including a description of all data acquisition and pre-processing steps involved for better reproducibility. Chapter 5 then shows several simulation studies trying to experimentally derive properties of the bimovMF model where artificial data sets of similar size and dimensions are used as in the application data set. Chapter 7 then contains concluding remarks and pointers for further research.

# Chapter 2

# Preliminaries

## 2.1 von Mises–Fisher distribution (vMF)

A $d$–dimensional random unit vector $x$ is said to follow a von Mises-Fisher (vMF) distribution if its density can be written as (Dhillon and Sra, 2003):

$$f(x) = c_d(\kappa)\exp(\kappa\mu^T x) \tag{2.1}$$

with $||x|| = 1$, $||\mu|| = 1$, $\kappa \geq 0$ and the normalizing constant $c_d(\kappa)$ being

$$c_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{d/2-1}(\kappa)} \tag{2.2}$$

with $I_{d/2-1}(\cdot)$ being the modified Bessel function of the first kind of order $d/2 - 1$.

Any obversation $x$ is a vector of fixed length ($||x|| = 1$), so any two random vectors from a vMF distribution only differ in their directions. As such the vMF distribution is commonly referred to as a directional distribution.

The parameter $\mu$ denotes the mean direction, while $\kappa$ is called the concentration parameter. The latter is roughly equivalent to the inverse of the variance of a normal distribution. The vMF distribution can even be motivated by deriving the conditional distribution of a normal distribution given all points lie on the unit hypersphere: let

$$z \sim N(\mu, \kappa^{-1} I_p), \quad \text{with } ||\mu|| = 1 \tag{2.3}$$

where $I_p$ denotes the identity matrix, then

$$z \big| ||z|| = 1 \sim \text{vMF}(\mu, \kappa) \tag{2.4}$$

## 2.2 Mixtures of vMF distributions

For the purpose of clustering text documents Banerjee et al. (2005) propose a mixture of vMF distributions. Generally, a mixture model has the form

$$f(x|\Theta) = \sum_{h=1}^{k} \alpha_h f_h(x|\theta_h) \tag{2.5}$$

with $\Theta = \{\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k\}$ being the parameter vector. $\alpha_h$ are a priori weights for each component and $f_h(x|\theta_h)$ is the density of the $h$-th component with corresponding parameters $\theta_h = (\mu_h, \kappa_h)$. An observation from such a mixture is usually generated by first drawing from a multinomial distribution with probabilities $(\alpha_1, \dots, \alpha_k)$, thus determining class membership. The final observation is then drawn from the distribution of this component.

The likelihood of a mixture model is, in general, rather difficult to optimize unless the true class assignments of each observation is known. Banerjee et al. (2005) use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to circumvent this problem. The EM algorithm iteratively improves the current likelihood estimate and can be shown to converge to a stationary fixed point if the likelihood is bounded (Dempster et al., 1977, Theorem 2). Each iteration consists of two steps called E-step and M-step respectively. During the E-step the expected log-likelihood is computed which is then maximised at the M-step with respect to the parameters $\Theta$ (see also Fraley and Raftery (2000, p. 10)). These two steps are repeated until a convergence criterion is met, such as that the absolute log-likelihood difference of two succeeding iterations falls below a pre-defined threshold.

Let $x = (x_1, \dots, x_n) \in \mathbb{R}^{n \times d}$ be the observed data with $n$ observations and let $z = (z_1, \dots, z_n) \in \{1, \dots, k\}^n$ be a hidden random variable determining class membership, such that observation $i$ is drawn from $f_{z_i}(\cdot|\theta_{z_i})$, assuming a total of $k$ classes. Then $(x, z) \in \mathbb{R}^{n \times (d+1)}$ form the complete data and assuming $z$ is known the complete data log-likelihood is written as

$$\ln P(x, z|\Theta) = \sum_{i=1}^{n} \ln\left(\alpha_{z_i} f_{z_i}(x_i|\theta_{z_i})\right) \tag{2.6}$$

Computing the expected value of (2.6) mostly entails estimating the conditional distribution of the class labels given the observed data and current parameter estimates, which Banerjee et al. (2005) derive as follows:

$$p(h|x_i, \Theta) = \frac{\alpha_h f_h(x_i|\theta_h)}{\sum_{l=1}^{k} \alpha_l f_l(x_i|\theta_l)} \tag{2.7}$$

Maximizing the expected value of (2.6) under the constraints $\mu_h^T \mu_h = 1$ and $\kappa_h \geq 0$ Banerjee et al. (2005) then determine the following update equations:

$$\alpha_h = \frac{1}{n} \sum_{i=1}^{n} p(h|x_i, \Theta) \tag{2.8}$$

$$r_h = \sum_{i=1}^{n} x_i p(h|x_i, \Theta) \tag{2.9}$$

$$\hat{\mu}_h = \frac{r_h}{||r_h||} \tag{2.10}$$

$$\frac{I_{d/2}(\hat{\kappa}_h)}{I_{d/2-1}(\hat{\kappa}_h)} = \frac{||r_h||}{\sum_{i=1}^{n} p(h|x_i, \Theta)} \tag{2.11}$$

Note also that the estimates for $\kappa_h$ can only be expressed as fraction of two Bessel-functions and in practice the value of $\kappa_h$ will have to be determined by a numerically suitable method.

# Chapter 3

# Model

## 3.1 Specification

In the following an adaptation of the movMF model of Banerjee et al. (2005) described in section 2.2 is presented in order to model mixtures on bilingual text corpora. Here, a bilingual corpus is considered as a set of documents in two languages. For a subset of these documents the respective translation is contained in the corpus as well.

Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d_x}$ and $y = (y_1, \ldots, y_n) \in \mathbb{R}^{n \times d_y}$ be two row–normalised ($||x_i|| = ||y_i|| = 1$ for all $i$) document-term-matrices, each containing documents of one language. Note that row $x_i$ and $y_i$ correspond to the same document in different languages. For documents which are only available in one language $x_i$ or $y_i$ is assumed to be $(0, \ldots, 0)$. Then let $c_x = (c_{x1}, \ldots, c_{xn}) \in \{0,1\}^n$ and $c_y = (c_{y1}, \ldots, c_{yn}) \in \{0,1\}^n$ be indicators whether document $i$ is available in language $x$ or $y$ respectively. For document $i$ the mixture density with $k$ components is written as

$$H(x_i, y_i|\Theta) = \sum_{h=1}^{k} \alpha_h \cdot f_h(x_i|\theta_h)^{c_{xi}} \cdot f_h(y_i|\psi_h)^{c_{yi}} \tag{3.1}$$

with $f_h(\cdot|\cdot)$ being the density of the von Mises-Fisher distribution, $\theta_h = (\mu_{xh}, \kappa_{xh})$ and $\psi_h = (\mu_{yh}, \kappa_{yh})$ the respective parameters in component $h$ and $\Theta = (\alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k, \psi_1, \ldots, \psi_k)$ being the vector of all parameters. When there are documents of both languages available equation (3.1) can also be viewed as a mixture density where each component follows the joint distribution of two vMF distributions[1].

[1] This is equivalent to the "combined approach" in (Andrews and Currim, 2003).

## 3.2 Estimation

Let $z = (z_1, \ldots, z_n) \in \{1, \ldots, k\}^n$ be the hidden random variable that determines the class membership of observation $i$. Let further $z_{ih}$ be the corresponding indicator variable:

$$z_{ih} := \begin{cases} 1 & z_i = h \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Assuming $z$ is known, the complete data log likelihood of this model is written as

$$\ln P(x, y, z|\Theta) = \sum_{i=1}^{n} \ln \left[\alpha_{z_i} \cdot f_{z_i}(x_i|\theta_{z_i})^{c_{xi}} \cdot f_{z_i}(y_i|\psi_{z_i})^{c_{yi}}\right] \tag{3.3}$$

and its expectation over $p$ is

$$E_p \left[\ln P(x, y|\Theta)\right] = \sum_{i=1}^{n} \sum_{h=1}^{k} \Big[\big(\ln \alpha_h + c_{xi} \ln f_h(x_i|\theta_h) + c_{yi} \ln f_h(y_i|\psi_h)\big) \cdot p(h|x_i, y_i, \Theta)\Big] \tag{3.4}$$

The update equations for the EM algorithm are obtained analogously to (2.7) and (2.8)-(2.11) in much the same way as outlined in Banerjee et al. (2005, A.2). First off, for the posterior class probability the joint density is substituted:

$$p(h|x_i, y_i, \Theta) = \frac{\alpha_h \cdot f_h(x_i|\theta_h)^{c_{xi}} \cdot f_h(y_i|\psi_h)^{c_{yi}}}{\sum_{\ell=1}^{k} \alpha_\ell \cdot f_\ell(x_i|\theta_\ell)^{c_{xi}} \cdot f_\ell(x_i|\psi_\ell)^{c_{yi}}} \tag{3.5}$$

From (3.3) it is apparent that $\alpha$, $\theta$ and $\psi$ can be optimized independently from one another and one subsequently obtains (see appendix **??**)

$$\alpha_h = \frac{1}{n} \sum_{i=1}^{n} p(h|x_i, y_i, \Theta) \tag{3.6}$$

$$r_{xh} = \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \tag{3.7}$$

$$\hat{\mu}_{xh} = \frac{r_{xh}}{||r_{xh}||} \tag{3.8}$$

$$\frac{I_{d/2}(\hat{\kappa}_{xh})}{I_{d/2-1}(\hat{\kappa}_{xh})} = \frac{||r_{xh}||}{\sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta)} \tag{3.9}$$

Parameter estimates for $\psi$ are analog to the $\theta$ estimators above but are omitted for brevity. These estimates are the same as for the unilingual case except that the posterior class probability depends on the joint density over both data sets (where available). Also another addition to the estimators are the indicators $c_{xi}$ and $c_{yi}$ so that the estimates are only influenced where there was data observed.

Having all the parts in place, algorithm 1 on page 10 shows the EM algorithm derived from the softmax algorithm in (Banerjee et al., 2005). The document-term-matrices $x$ and $y$ are considered the input to the algorithm as well as the indicators $c_x$ and $c_y$. In step 2 initial values for the parameter estimates are chosen: firstly, let $P$ be the matrix of posterior class probabilities for each component $h$ and each observation $i$

$$P = \Big(p(h|x_i, y_i, \Theta)\Big)_{i,h} = \begin{pmatrix} p(1|x_1, y_1, \Theta) & \cdots & p(k|x_1, y_1, \Theta) \\ \vdots & \ddots & \vdots \\ p(1|x_n, y_n, \Theta) & \cdots & p(k|x_n, y_n, \Theta) \end{pmatrix}. \tag{3.10}$$

Since $P$ can be estimated via $(\theta, \psi, \alpha)$ and vice-versa, in practice it is sufficient to initialise the algorithm with either an initial $P$ matrix or a parameter triple $(\theta, \psi, \alpha)$. The algorithm then alternates between the E and the M-step: first the conditional expectation $E_p[\ln P(x, y|\Theta)]$ is computed using the initial parameter values (step 3). This expectation is then maximised with respect to the parameters $(\theta, \psi, \alpha)$ (step 4) to form new current parameter estimates. Those are in turn used to re-compute the conditional expectation in the next iteration and so on. The algorithm is then considered converged if the likelihood improvement between two consecutive iterations falls under a pre-defined threshold.

As noted earlier the EM algorithm converges towards a local maximum or stationary fixed-point if the likelihood is bounded, meaning that the estimation usually depends on the choice of the initial parameter values. In some cases, such as some outlined in section 5, the estimates rely heavily on the initialisation step and are often of poor quality. To alleviate this problem, a modification to the EM algorithm (Ueda and Nakano, 1998) is shown in algorithm 2: the Deterministic Annealing EM (DAEM) algorithm further introduces parameters $\beta \in (0, 1)$ and $\delta\beta$. Instead of computing and maximising the conditional expectation $E_p[\ln P(x, y|\Theta)]$ an objective function depending on $\beta$ is computed and maximised:

$$\begin{aligned} U_\beta(x, y|\Theta) = \sum_{i=1}^{n} \sum_{h=1}^{k} \quad & \Big[ \big( \ln \alpha_h + c_{xi} \ln f_h(x_i|\theta_h) + c_{yi} \ln f_h(y_i|\psi_h) \big) \\ & \cdot \tfrac{p(h|x_i, y_i, \Theta)^\beta}{\sum_{\ell=1}^{k} p(\ell|x_i, y_i, \Theta)^\beta} \Big] \end{aligned} \tag{3.11}$$

which is mostly the same as $E_p[\ln P(x, y|\Theta)]$ with the exception that the term $p(h|x_i, y_i, \Theta)$ is raised to the power of $\beta$ and divided by a normalizing term. $\beta$ can be seen here as a smoothing parameter applied to $U_\beta(x, y|\Theta)$ while $\delta\beta$ represents a constant proportional to the rate of change of $\beta$. The concrete choice of $\beta$, $\delta\beta$ and a mapping $(\beta_t, \delta\beta) \mapsto \beta_{t+1}$ is somewhat arbitrary; $\beta_0$ should be small enough so that only one global maximum is visible. According to Ueda and Nakano (1998) $\beta_0$ of 0.1 is usually good enough[2]. They also suggest a multiplicative increase of $\beta$ per iteration, thus the value of $\beta$ in the $t+1$-th iteration is computed as $\beta_{t+1} = \beta_0(\delta\beta)^t$.

The DAEM algorithm can informally be motivated as follows: at first a small value of $\beta$ is chosen resulting in a high degree of smoothing of the objective function. Being highly smoothed the local maxima would almost disappear making the parameter estimates move close to the global maximum. Then at each iteration step one reduces the smoothing (increase $\beta$) by a small amount and determines new parameter estimates in order to refine the estimates until finally the parameters are estimated using the original objective function.

[2]For the datasets considered here $\beta_0 = 0.01$ seemed to produce better results.

**Algorithm 1** (EM)**.**

1. **Input** data matrix $(x, y)$ and indicators $(c_x, c_y)$
2. **Initialize** Choose initial $P$ or $(\theta, \psi, \alpha)$
3. **E-step** Compute the conditional expectation $E_p[\ln P(x, y|\Theta)]$ using (3.5)
4. **M-step** maximise $E_p[\ln P(x, y|\Theta)]$:

   $$\Theta = (\theta, \psi, \alpha) \leftarrow \arg\max_{\theta,\psi,\alpha} E_p[\ln P(x, y|\Theta)]$$

   using (3.6)–(3.9).
5. **Repeat** steps 3 and 4 until convergence

**Algorithm 2** (DAEM)**.**

1. **Input** data matrix $(x, y)$ and indicators $(c_x, c_y)$
2. **Initialize** Choose initial $P$ or $(\theta, \psi, \alpha)$ and initial $\beta_0$, $\delta\beta$
3. **E-step** Compute $U_\beta(x, y|\Theta)$ using (3.11)
4. **M-step** Maximise $U_\beta(x, y|\Theta)$:

   $$(\theta, \psi, \alpha) \leftarrow \arg\max_{\theta,\psi,\alpha} U_\beta(x, y|\Theta)$$

   using (3.6)–(3.9).
5. **Increase** $\beta$
6. **Repeat** steps 3 and 5 while $\beta < 1$

# Chapter 4

# Implementation

## 4.1 $\kappa$ estimation

Recall that the ML-estimator for $\kappa_{xh}$ in the bilingual model is written as (equation 3.9)

$$\frac{I_{d/2}(\hat{\kappa}_{xh})}{I_{d/2-1}(\hat{\kappa}_{xh})} = \frac{||r_{xh}||}{\sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta)}$$

A small user-supplied correction factor $\nu \geq 0$ was added to avoid numerical overflows (Hornik and Grün, 2012):

$$\frac{I_{d/2}(\hat{\kappa}_{xh})}{I_{d/2-1}(\hat{\kappa}_{xh})} = \frac{||r_{xh}||}{\nu + \sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta)} \tag{4.1}$$

Throughout this thesis a value of $\nu = 0.01$ was used. Instead, an absolute upper bound as in Banerjee et al. (2005) could have been used. Defining the left-hand side of (4.1) as $A_d(\kappa)$ and the right-hand side as $\bar{r}$, solving for $\kappa_{xh}$ can be written as a one dimensional root finding problem which is not specific to this particular model[1]:

$$A_d(\kappa) = \bar{r} \tag{4.2}$$

Since $A_d(\kappa)$ is the fraction of two bessel functions, an expression for $\kappa$ can not be found in closed form and numeric methods have to be employed. Banerjee et al. (2005) present the following estimator, derived by the continued fraction representation of $A_d(\kappa)$ and with an added correction term "determined empirically":

[1]For brevity $\kappa_{xh}$ is instead written as $\kappa$

$$\kappa \approx \frac{\bar{r}d - \bar{r}^3}{1 - \bar{r}^2} \tag{4.3}$$

Hornik and Grün (2012) show that $\kappa$ can be estimated using standard root finding algorithms, for example `uniroot` in R, given that good starting values are supplied. They further show that the approxmiation used by Banerjee et al. (2005) has the same maximum estimation error but is biased. The default estimator in Hornik and Grün (2012), which is also used in this thesis' implementation, however uses the Newton algorithm to find the root of $A_d(\kappa) - \bar{r}$.

## 4.2 Empty components

A common problem in mixture models is that single components might become empty. This could happen if the estimated prior class probability $\alpha_h$ is estimated to be very small for one component. This can be problematic for variance estimates for example. For very few observations in this component the variance would be estimated to be extremely high, which can cause numeric difficulties later on. More so, if only a single observation is assigned to a cluster, the variance would often be estimated as infinite. To avoid this, such components are often dropped when they fall below a minimal allowed value for $\alpha_h$. The choice of this lower threshold is often arbitrary and while this was also implemented during this thesis, there is an alternative. When a component gets empty and with no assigned observations, then $\kappa_h$ is estimated as 0, and therefore the distribution in this component would be equal to the uniform distribution on the unit hypersphere. This is facilitated by the fact that the $\kappa$ estimator in Hornik and Grün (2012) allows the precision parameter to become 0.

# Chapter 5

# Simulation

This chapter shows the result of a number of simulation studies performed. These simulations were run with similar number of observations and dimensionality as the AJS data set from section 6. Generally, in each simulation run a dataset $(x, y)$ was drawn from the bilingual mixture model with $n = 220$ observations and $d_x = 2000$ and $d_y = 3000$ dimensions. Per simulation there were 100 data sets simulated, with 5 components each. Here, the concentration parameters $\kappa_h$ were kept equal for all components, while the estimation algorithm however did not assume these parameters to be the same. The mean direction $\mu_h$ was drawn from a uniform distribution and then normalised such that $||\mu_h|| = 1$ holds. In simulations where two or more models were to be compared, the models were estimated with the same initial values; as initialization the matrix of posterior probabilities $P$ (3.10) was randomly drawn from a uniform distribution and rescaled for all row-sums to be equal to 1 ($\sum_{j=1}^{k} P_{ij} = 1,\ \forall i$). Also, since the model estimates depend on the starting values, in each simulation run, every model was estimated with 25 different initial $P$ matrices, for both the EM and DAEM algorithms. From these, the model exhibiting the highest log-likelihood value was chosen. The DAEM algorithm still depends on the starting values and can benefit from choosing different initializations, however the effect is less pronounced as with the EM algorithm. The EM algorithm was considered converged, when the absolute difference of the log-likelihood in subsequent iterations fell below $10^{-6}$; a maximum number of iterations was not set. The DAEM algorithm was computed with $\beta_0 = 0.01$ and $\delta\beta = 1.1$, the convergence criterion for this algorithm is $\beta \geq 1$ and hence all DAEM estimates went through 49 iterations until this criterion is met.

In some instances bilingual and unilingual models were compared. The unilingual model can be regarded as a special case of the bilingual model with $c_y = 0$. This way, unilingual estimates using the DAEM algorithm were obtained.

## 5.1 Excursus: Mutual Information

As a quality criterion determining how well a model performed in a simulation given that the true cluster memberships are known the *normalized mutual information (NMI)* is used. This quantity however deserves a more detailed explanation. As a prerequisite let $X$ be a discrete[1] random variable which assumes values from a set (alphabet) $\mathcal{A}_X = \{x_1, \ldots, x_{|\mathcal{A}_X|}\}$ with probabilities $\mathcal{P}_X = \{P(X = x)|x \in \mathcal{A}_X\}$.

**Definition 1** (Shannon information content of $x$)**.**
The Shannon information content of a realisation $x$ of a random variable $X$ is defined as (MacKay, 2005, eq. 2.34):

$$h(x) = \log_2 \left( \frac{1}{P(X = x)} \right)$$

**Definition 2** (Entropy)**.**
Let $X$ be a (discrete) random variable. The entropy of $X$ is the expected Shannon information content (MacKay, 2005, eq. 2.35):

$$H(X) = E\left[h(x)\right] = \sum_{x \in \mathcal{A}_X} P(X = x) \ln \left( \frac{1}{P(X = x)} \right)$$

with $0 \times \ln 1/0 \equiv 0$ for $P(X = x) = 0$ since $\lim_{\theta \to 0^+} \theta \ln(1/\theta) = 0$.

The entropy is non-negative and $H(X) \leq \log(|\mathcal{A}_X|)$.

The mutual information (MI) can be used to measure the agreement between estimated and true cluster labels and is defined as follows (Ghosh and Banerjee, 2006):

**Definition 3** (Mutual Information)**.**
Let $X$ and $Y$ be two discrete random variables; the mutual information is then defined by (MacKay, 2005, eq. 8.11)

$$I(X;Y) = E\left[ \frac{P(X,Y)}{P(X)P(Y)} \right] = D_{KL}\Big(X, Y \Big|\Big| XY\Big)$$

[1]In Information Theory one is usually concerned with the encoding of a message from an information source into a digital message. The latter is encoded as a stream of consecutive bits, which are either 0 or 1. For such a digital message the information content is computed, hence the information content is defined for discrete random variables.

The mutual information is the expectation of the fraction of the joint distribution by the joint distribution under the assumption of independence. Or more intuitively the mutual information can be written as the Kullback-Leibler divergence between the joint distribution and the product of the marginal distributions. This divergence assumes larger values when these two distributions are very dissimilar, that is, when $X$ and $Y$ depend on one another. While the Kullback-Leibler divergence is not generally symmetric, in this case the symmetry holds for the mutual information: $I(X;Y) = I(Y;X)$ (MacKay, 2005, exercise 8.4). Moreover the mutual information satisfies *Gibbs' inequality*, that is $I(X;Y) \geq 0$ with equality iff $X$ and $Y$ are independent (MacKay, 2005, eq. 2.46).
The normalized mutual information is then defined as the mutual information divided by its maximum value in order to lie in the interval $[0;1]$ (Ghosh and Banerjee, 2006):

**Definition 4** (Normalized Mutual Information)**.**

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$

with $H(X)$ and $H(Y)$ being the entropy of $X$ and $Y$ respectively.

In subsequent sections $X$ and $Y$ correspond to the distributions of the true and estimated cluster labels respectively. The NMI will usually be computed from the empirical distributions, for example see the contingency tables in table 5.1 The two contingency tables show the result of a fictional

(a)

| X/Y | 1 | 2 | |
|---|---|---|---|
| 1 | 8 | 16 | 24 |
| 2 | 16 | 10 | 26 |
| | 24 | 26 | 50 |

(b)

| X/Y | 1 | 2 | |
|---|---|---|---|
| 1 | 1 | 23 | 24 |
| 2 | 24 | 2 | 26 |
| | 25 | 25 | 50 |

Table 5.1: Example contingency tables

2-dimensional classification problem. In table (a) many observations were incorrectly classified while in table (b) most of othe observations are correctly classified. Also the latter shows an example of label switching where most observations from the first class are assigned to the second. The NMI is invariant towards changes in cluster labels which is a useful property for assessing the quality of clusterings in a mixture model.

The NMI values are computed by first computing the expectation of the mutual information using

$$I(X;Y) = \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} P(X = x, Y = y) \ln \left( \frac{P(X = x, Y = y)}{P(X = x)P(Y = y)} \right) \tag{5.1}$$

The joint and marginal density are derived from the contingency table (by dividing by the total number of observations: here 50). For example (a) equation 5.1 expands to

$$\begin{aligned} I(X_{(a)}; Y_{(a)}) &= \tfrac{8}{50} \ln\left(\frac{\frac{8}{50}}{\frac{24}{50}\cdot\frac{24}{50}}\right) + \tfrac{16}{50} \ln\left(\frac{\frac{16}{50}}{\frac{24}{50}\cdot\frac{26}{50}}\right) + \tfrac{16}{50} \ln\left(\frac{\frac{16}{50}}{\frac{24}{50}\cdot\frac{26}{50}}\right) + \tfrac{10}{50} \ln\left(\frac{\frac{10}{50}}{\frac{26}{50}\cdot\frac{26}{50}}\right) \\ &\approx 0.040 \end{aligned}$$

For the normalizing constant the entropies of the margins are to be computed:

$$\begin{aligned} H(X_{(a)}) &= \sum_{x \in \mathcal{A}_X} P(X = x) \ln\left(\frac{1}{P(X = x)}\right) \\ &= \tfrac{24}{50} \ln\left(1/\tfrac{24}{50}\right) + \tfrac{26}{50} \ln\left(1/\tfrac{26}{50}\right) \approx 0.692 \\ H(X_{(b)}) &\approx 0.692 \end{aligned}$$

Thus the NMI of example (a) evaluates to approximately 0.058 and analogously the NMI of example (b) to about 0.678.

## 5.2 Comparison by concentration parameter

In the following the effect of the concentration parameter is investigated. The data sets were simulated with the same value of $\kappa_h$ in each component. With increasing $\kappa_h$ (decreasing variance) one would expect the classification problem to become easier since the data points would be better separated. For each simulated data set the NMI was computed given the respective model estimates and the true cluster assignments. Figure 5.1 shows the resulting median NMI values. The DAEM algorithm (black) achieved a fairly small NMI value of around 0.35 for high variance but then quickly increases to obtain near perfect estimates at $\kappa_h \geq 800$. The unilingual DAEM estimates (black dashed) show the same tendency but increase in quality a bit slower than the bilingual model. The EM algorithm (red) results in poor classifications at first and slowly improves with smaller variance and also obtains near-perfect estimates at very small variances ($\kappa_h \geq 1200$). The unilingual EM estimates perform somewhat worse and also seem to improve more slowly than the bilingual variant. As comparison the EM algorithm initialised with true parameters is shown in green. The model seems to be able to properly separate the data for good parameter estimates. Arriving at good estimates however, seems difficult.

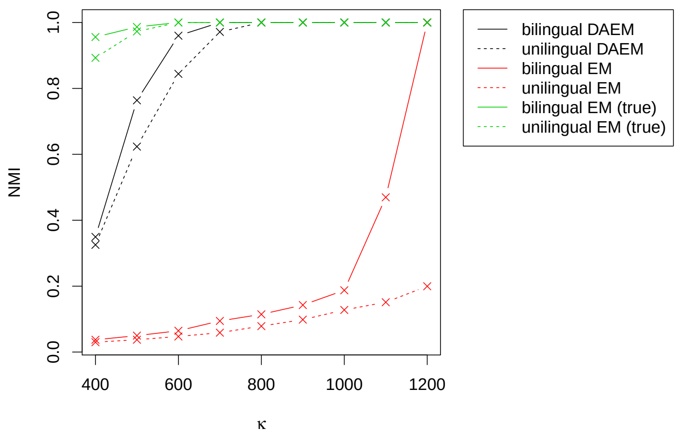

Figure 5.1: Medians of simulated NMI values for different values of $\kappa$ comparing bi- and unilingual models estimated using the DAEM and EM algorithms and EM initialised with true parameters.

## 5.3 Incomplete datasets

Following the effect of having an incomplete $y$ dataset is investigated. The simulated data sets are of size $x \in \mathbb{R}^{n \times 2000}$ and $y \in \mathbb{R}^{n_y \times 3000}$, meaning that in the $y$ data set (corresponding to the german corpus from the AJS example) only $n_y$ observations were made. $\kappa_h$ was set to 600 for all components, $n$ is fixed at 220 and the ratio $n_y/n$ is then varied in simulations and the result shown in figure 5.2. Ratio $n_y/n = 0$ would be equivalent to the unilingual movMF model on the $x$ data set and $n_y/n = 1$ equivalent to the full bilingual model. For each of the 100 data sets models were computed using different ratios $n_y/n \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}$. On the left hand side of figure 5.2 boxplots of the NMI values obtained using the DAEM algorithm are displayed. The bilingual model (ratio = 1) achieves the best classification. For small ratios the NMI values tend to decline compared to the unilingual model while for ratios $\geq 0.5$ the "partial" bilingual model achieves better results than the unilingual model. It would appear that having only a small number of extra observations diverts the model from performing a good classification. On the other hand, a ratio of 0.1 would mean that $y$ would be a dataset of around 22 observations and 3000 dimensions, which would be very surprising if a good estimate could be performed on such a data set.

Figure 5.2 on the right side shows boxplots of NMI values obtained by the EM algorithm initialised with true parameters, resulting in very high NMI values. The EM algorithm itself performed badly for all ratios (not shown).

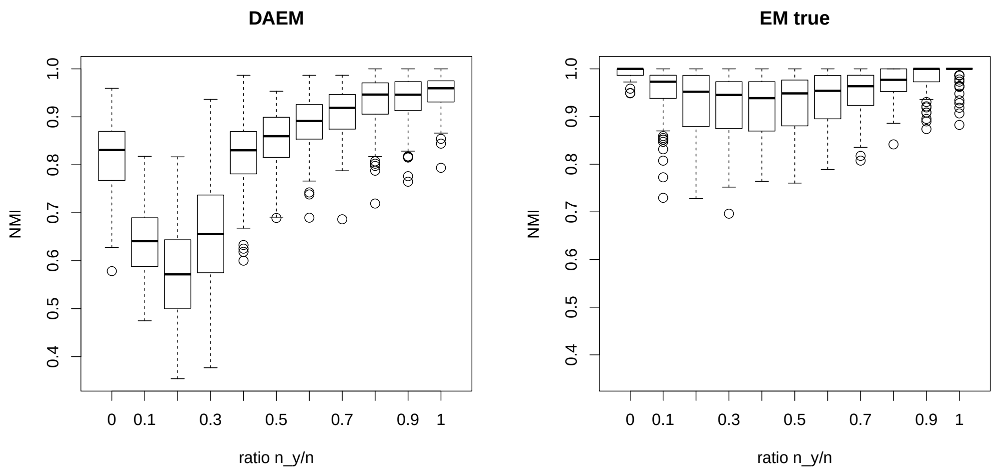

Figure 5.2: Boxplots of simulated NMI values for different ratios $n_y/n$ estimated via DAEM (left) and EM initialised with true parameter values (right).

## 5.4 Comparison by dataset size

Following the effect of the dataset size on the classification accuracy is investigated. Simulated datasets were of size $x \in \mathbb{R}^{n \times 2000}$ and $y \in \mathbb{R}^{n \times 3000}$ with $\kappa_h = 500$ for all five components. This simulation was intentionally designed to be harder so the effect of increasing sample size would be visible. Figure 5.3 shows the result of this simulation. On the left hand side boxplots of the NMI values for the DAEM algorithim are displayed. For small datasets the NMI values tend to be very small ($\leq 0.2$) showing that the classifications were of little quality. At 500 observations the quality of the estimates start to increase while achieving almost perfect results at $n \geq 800$. The right hand side shows the medians of the NMI values for the different algorithms. For the DAEM algorithm the same trend is visible as in the boxplot, the EM algorithm seems to perform badly for any $n$ in the range of 100–1000. The EM algorithm initialised with true parameters always achieves near-perfect classification, hence the model itself can differentiate between classes very well but it appears to be very hard to find good parameter estimates. These models also exhibited a considerably higher log-likelihood value than the DAEM models with random initializations.

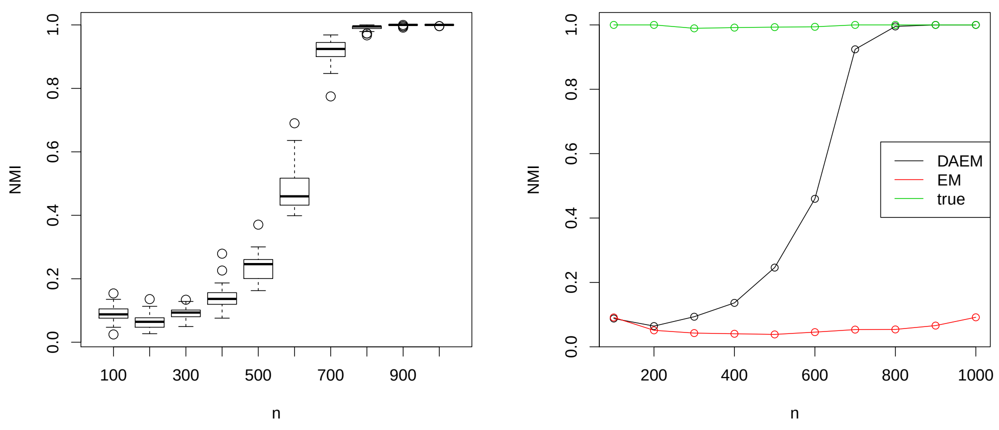

Figure 5.3: Boxplots for DAEM estimates (left) and Median NMI (right) of simulations with varying dataset sizes.

## 5.5 Comparison by number of dimensions

Here, datasets were simulated with varying dimensions, i.e. $x, y \in \mathbb{R}^{n\times d}$ with $d \in \{500, 1000, \ldots, 4000\}$. Figure 5.4 shows the resulting median NMI values per number of dimensions $d$ and per algorithm. The DAEM algorithm (black) seems to achieve near-perfect classifications for $d$ up to 1500; for larger dimensions the clustering quality seems to slowly decrease. The EM algorithm (red) shows near-perfect estimates as well for $d \leq 1000$ and then steeply declines towards very poor estimates. The EM algorithm initialised with true parameters (green) always achieves near-perfect accuracy indicating that the clusters are separable and can be correctly classfied by the bilingual movMF model. For larger dimensions it seems to be increasingly difficult however to obtain parameter estimates near the global maximum. These models also exhibited a considerably higher log-likelihood value than the DAEM models with random initializations.

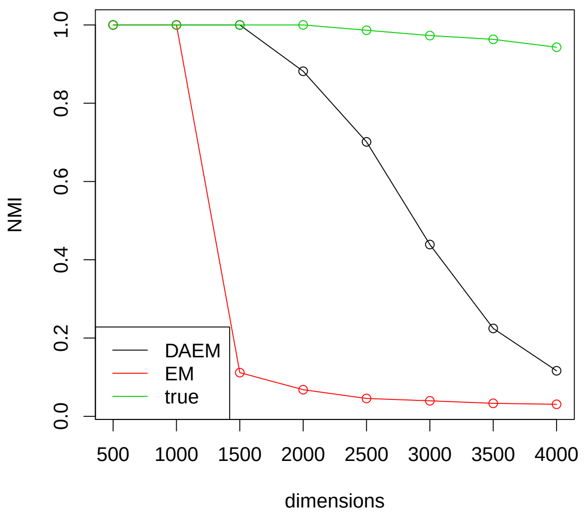

Figure 5.4: Median NMI values plotted against the number of dimensions for DAEM and EM algorithms and EM initialised with *true* parameters.

# Chapter 6

# Application

As an application to the bilingual movMF model the corpus of abstracts of the Austrian Journal of Statistics (AJS) is analyzed in this section. The Austrian Journal of Statistics is published by the Austrian Statistical Society. It is currently edited by Herwig Friedl of the TU Graz. The editions are usually published online free of charge or may be purchased in printed form. The remarkable property of this journal however, which gave rise to the following analyses, is the fact that for many articles the abstract is published both in english and in a german translation.

This section will also describe the steps involved in pre-processing the text documents. Many articles contain a german abstract alongside an english one. Assuming the german abstracts are translations of the english abstract one can treat the set of english and german abstracts as a parallel corpus. In some cases only an english abstract is present. The bilingual movMF model is then used to model all uni- and bilingual abstracts simultaneously.

The articles published via the AJS are accessible as PDF[1] files from the official website `http://www.stat.tugraz.at/AJS`. This task can easily be accomplished by a website download utility, such as `HTTrack`[2] or `GNU wget`[3]. The AJS website also includes editorials, obituaries and Festschriften which are however excluded from the following analysis and only "regular" articles will be considered.

[1]Portable Document Format

[2]HTTrack Website Copier, `http://www.httrack.com`

[3]`http://www.gnu.org/software/wget/`

## 6.1 Pre-processing

The first step in pre-processing is to convert the PDF files into the HTML format, which is easier to parse than PDF. Most programming languages contain libraries for parsing HTML, such as the `XML` library in R (Lang, 2012). In this case, the text from the PDF files was extracted using `Apache PDFBox`[4]. `PDFBox` outputs the result in an HTML format, for which the structure is most conveniently explained using an example. Listing 6.1 shows the output of `PDFBox` for one article mildly adjusted for readability and some uninteresting parts left out. Firstly the whole document is enclosed in `<html>` and `<body>` tags. Each page of the article is contained in a `<div>` tag, which then contains a number of paragraphs (`<p>`). In this example the first paragraph contains a header line denoting the name of the journal, the volume and page number (omitted) (line 4). The following paragraphs then contain the title and the authors of this article (lines 5–7). The numbers behind the authors' names used to be footnotes and are displayed without formatting now. The next paragraph in lines 8–16 contains information about the institutions of the authors as well as the abstracts and the keywords. One thing visible already is that the umlaut ü was encoded as `&#776;` and the beginnings of the abstracts and key words are indicated by strings in the form of "Abstract:". Line 17 denotes the end of the first page; the second page (lines 18–22) first contains a header line again (line 19) and then continues with the normal text (lines 20–21).

Listing 6.1: `PDFBox` example output

```
1  <html>
2  <body>
3  <div>
4  <p>AUSTRIAN JOURNAL OF STATISTICS Volume [...]</p>
5  <p>Domain-Based Benchmark Experiments [...]</p>
6  <p>Manuel J. A. Eugster1, Torsten Hothorn1,
7     and Friedrich Leisch2</p>
8  <p>1Institut fu&#776;r Statistik, LMU Mu&#776;nchen,
9    Germany 2Institut fu&#776;r Angewandte Statistik
10   und EDV, BOKU Wien, Austria
11   Abstract: Benchmark experiments are the method of
12   choice to compare [...]
13   Zusammenfassung: Benchmark-Experimente sind die
14   Methodik der Wahl [...]
15   Keywords: Benchmark Experiment [...].
16 </p>
```

[4] `http://pdfbox.apache.org`

```
</div>
<div>
<p>6 Austrian Journal of Statistics [...]</p>
<p>1 Introduction</p>
<p>In statistical learning [...]</p>
</div>
</body>
</html>
```

As a precursor to subsequent processing steps certain characters were either omitted or replaced in order to facilitate electronic processing:

- Umlaute: German umlaute can be represented in three different ways in the obtained HTML output. Many articles were apparently created by LaTeX, where umlaute are often written as the composite of the letter and a pair of dots above it, e.g. ä is written as `"a`. This structure is retained in the PDF file where a pair of dots is placed above the character by use of absolute or relative positioning. The other ways to encode umlaute are either special glyphs contained in the PDF file saying for example "this character is an a umlaut" or directly encoding the umlaut as unicode character[5]. Here the last method is chosen so that there is only one way in which umlaute can be represented.

- Punctuation is removed since the model described earlier in section 3 models word frequencies only and punctuation adds no additional information in this regard.

- Ligatures: certain pairs of characters are often joined as one glyph, most notably, ff and fl are written as ﬀ and ﬂ. These ligatures are removed so that the resulting word consists only of regular letters.

- Hyphens: some words are divided at the line-ending and separated by hyphens. In such instances the hyphens were removed and the word parts joined back together.

- Numbers are removed as well.

The next important step is to identify which parts of the article consitute the abstracts in either language. The PDF format contains little to no information about the logical structure of a document but mostly consists of layout primitives used to position text on paper. Common problems are

[5]Unicode also allows encoding umlaute as composite character, this case was however not encountered in this corpus.

that abstracts may be interspersed with header lines or footnotes, which are not considered to be part of the abstract and are sought to be omitted. As such a set of heuristics were developed by trial-and-error:

- Abstracts are usually contained in the first two pages of an article (the HTML output of PDFBox encloses each page in a `<div>` tag).
- The first paragraph in the output is usually a header line and is thus omitted.
- Footnotes start with a superscript digit. In the HTML output however footnotes begin with a digit and thus the last paragraphs of a page beginning with digits are omitted.
- Abstracts start with the string “Abstract:” or “Zusammenfassung:”. The end of an abstract is denoted by either the beginning of the other language’s abstract or by the string “Keywords:” or “Schlüsselwörter:”. Should the article contain no keywords, another criterion is needed to determine how many paragraphs should be counted towards the abstracts. Assuming that the first text after the abstracts is the first section headline which is usually written as “1 Section Title”, the abstract is considered to range up until this headline (footnotes and headers are already removed at this point).

Next, all words were stemmed using Porter’s english or german stemmer so that the same words in different flections are reduced to a common word stem (usually not equal to the grammatical word stem). For example the german occurences “Parametern”, “Parameters” and “Parameter” are reduced to “paramet”.

From the set of stemmed words two document-term-matrices for either language are constructed containing the frequencies of stemmed words on a per-document basis. Let there be $n$ articles, $d_x$ english stemmed words and $d_y$ german stemmed words. Let further $h_{xij}$ and $h_{yij'}$ be the absolute word frequencies of term $j$ (or $j'$ respectively) in document $i$. The document-term-matrices are then defined as

$$\mathrm{DTM}_x := \begin{pmatrix} h_{x11} & \dots & h_{x1d_x} \\ \vdots & \ddots & \vdots \\ h_{xn1} & \dots & h_{xnd_x} \end{pmatrix}, \quad \mathrm{DTM}_y := \begin{pmatrix} h_{y11} & \dots & h_{y1d_y} \\ \vdots & \ddots & \vdots \\ h_{yn1} & \dots & h_{ynd_y} \end{pmatrix} \tag{6.1}$$

## 6.2 Excursus: tf-idf

Following a weighting scheme is used which is called by the shorthand of tf-idf and was previously used in other text mining applications like Blei and Lafferty (2009) and Grün and Hornik (2011). tf-idf stems originally from the field of information retrieval (IR)[6]. A common problem in IR is to retrieve a (small) set of relevant documents from a large collection[7] of documents by the use of a search query. Consider an internet search engine as an example: a user inputs a set of keywords (the search query) expecting to obtain a list of relevant web sites. The search engine has a collection of text documents (i.e. web sites) from which an index was constructed counting for each document the number of occurrences of each term.

**Definition 5** (term frequency)**.** Let $t$ be a term and $d$ refer to a document in a collection, then

$$\mathrm{tf}_{t,d}$$

is the number of occurrences of term $t$ in document $d$ and is called the *term frequency.*

Naively one might try to answer the query above by retrieving documents which exhibit a high term frequency for the search terms. This would not account however for terms that occur very frequently but have little discriminative power. Most notably stop words are examples of such terms. Also, for a collection of documents about a specific domain there might be very frequent domain-specific words (e.g. "model" for a corpus about statistics). Therefore, in the tf-idf weighting scheme words that are contained in many documents are assigned a lower weight.

**Definition 6** (inverse document frequency)**.** Let $t$ be a term, then

$$\mathrm{idf}_t = \log_2 \left( \frac{N}{\mathrm{df}_t} \right)$$

is the *inverse document frequency* of that term in a collection, with $N$ being the total number of documents in the collection and $df_t$ the number of documents containing term $t$.

The *tf-idf* score is simply the product of both the term frequency and the inverse document frequency (Manning et al., 2009) or the product of term frequency and inverse document frequency with the term frequencies normalised by document length (Feinerer et al., 2008). The latter definition is used here.

[6]This section is largely based on Manning et al. (2009)

[7]A collection is a set of text documents and is used interchangably with "corpus" here.

**Definition 7** (tf-idf)**.** For a term $t$ and a document $d$ tf-idf is defined as

$$\text{tf-idf}_{t,d} = \log_2\left(\frac{N}{\text{df}_t}\right) \cdot \frac{\text{tf}_{t,d}}{\sum_{t\in\mathcal{T}} \text{tf}_{t,d}}$$

with $\mathcal{T}$ being the set of all terms in the collection. The tf-idf score for a term $t$ is defined as the mean tf-idf$_{t,d}$ score over the documents that contain $t$:

$$\text{tf-idf}_t = \log_2\left(\frac{N}{\text{df}_t}\right) \cdot \frac{\sum_{d\in D}\left(\text{tf}_{t,d}/\sum_{t\in\mathcal{T}}\text{tf}_{t,d}\right)}{\text{df}_t}$$

Consider the term "model" from our application data set for an example. It occurs 284 times in 95 documents. The data set consists of 220 documents in total, hence the tf-idf score for model is computed as

$$\text{tf-idf}_{\text{model}} = \frac{4.7}{95}\log_2\left(\frac{220}{95}\right) \approx 0.059 \qquad (6.2)$$

Whilst being the term with the highest term frequency, the tf-idf score puts it only into the 0.026 quantile, meaning only 60 of 2326 terms are ranked lower than "model" according to tf-idf.

In text mining applications the tf-idf score can be used to remove terms in order to reduce complexity. As Manning et al. (2009) put it, tf-idf weighs terms lower that have little "discriminative power". In clustering applications one would be most interested in keeping terms that discriminate well between clusters, therefore terms with low tf-idf scores are prime candidates for removal.

Document-term-matrices described in chapter 6 contain the term frequencies (note that the super-script $T$ is a transposed sign, since tf$_{t,d}$ indexes by terms first):

$$\text{DTM} = \left(\text{tf}_{t,d}\right)^T_{t\in\mathcal{T},\ d\in D}$$

By default, one would run the models on the term-frequencies. Another approach would be to rescale the document-term-matrix by tf-idf scores and use this as data matrix, as was done in Hornik and Grün (2012). The document-term-matrix would then read as follows

$$\text{DTM} = \left(\text{tf-idf}_{t,d}\right)^T_{t\in\mathcal{T},\ d\in D}.$$

## 6.3 Descriptives

The dataset considered here consists of 220 abstracts taken from the articles published in the Austrian Journal of Statistics over a timespan of eight years (2004–2012). From these articles, in 134 a german abstract could be automatically extracted as well.

Table 6.1 shows the 10 most frequently used stemmed words for english and german abstracts respectively.

| word | count | word | count |
|---|---|---|---|
| model | 284 | modell | 111 |
| estim | 274 | dat | 79 |
| data | 183 | statist | 70 |
| distribut | 159 | schätzer | 63 |
| statist | 136 | method | 54 |
| method | 111 | verteil | 48 |
| paper | 110 | verwendet | 37 |
| test | 100 | vorgeschlag | 33 |
| paramet | 97 | neu | 31 |
| sampl | 97 | schätzung | 31 |

Table 6.1: Word frequencies

### 6.3.1 Abstract lengths

Figure 6.1 shows the density of the length of abstracts. Unit of measurement is the number of stemmed words per abstract. Note also, that stop-words have been removed. The length of english and german abstracts seem to be similar with the germans being a bit longer. Mean abstract length is 55 (english) and 58.1 (german) words. The shortest english abstract on record amounted to 13 different terms while the shortest german abstract counts 17 terms. Not from a loss of words suffered the longest abstracts with 192 and 191 terms.

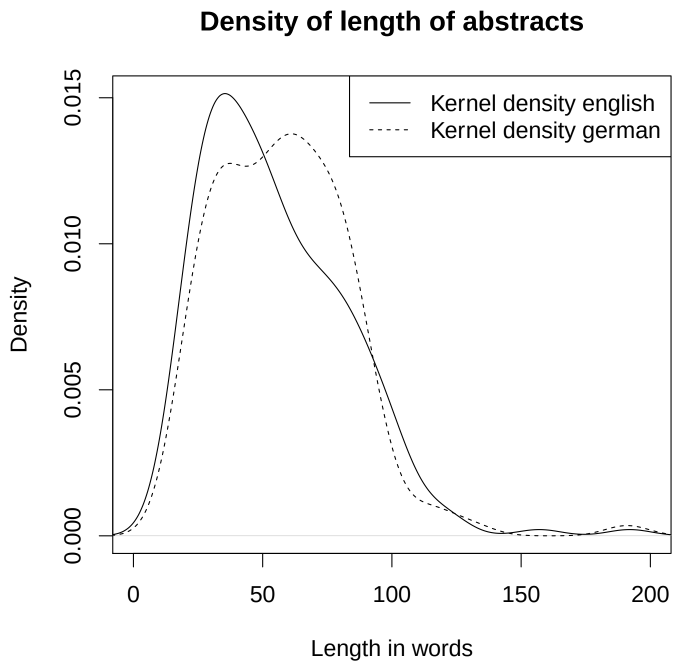

Figure 6.1: Kernel density estimates of the length of abstracts (in words) for english (solid) and german (dashed) abstracts.

### 6.3.2 Year of publication

Figure 6.2 shows the number of articles per year. With a total of 44 papers the year 2006 counts the most articles. In 2006 the AJS featured a normal issue as well as a double-issue containing proceedings of the third international workshop on *Perspectives on Modern Statistical Inference.* In 2010 the least amount (10) of papers were published, one of the three editions "only" consisting of a Festschrift. On average the AJS publishes 24.4 papers per year.

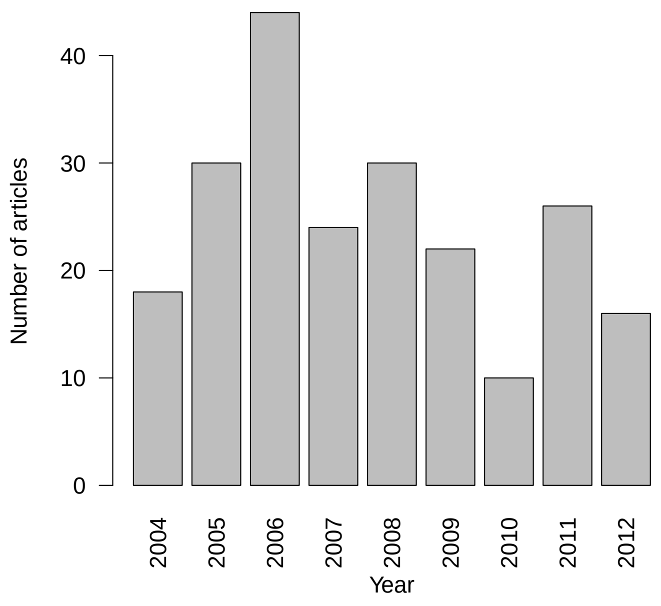

Figure 6.2: Number of articles per year

### 6.3.3 Articles per author

Figure 6.3 shows a barplot of how many authors published how many articles in the AJS. The author data was extracted from the web pages and were assembled assuming they were always spelt identically for multiple occurrences of author names. One likely typo was corrected by hand.

A total of 355 distinct authors' names were retrieved, of which the vast majority (300) published exactly one article in the AJS. 44 authors published two articles each, 5 authors wrote 3 or 4 papers respectively. Only one person did publish a total of 6 papers through the AJS.

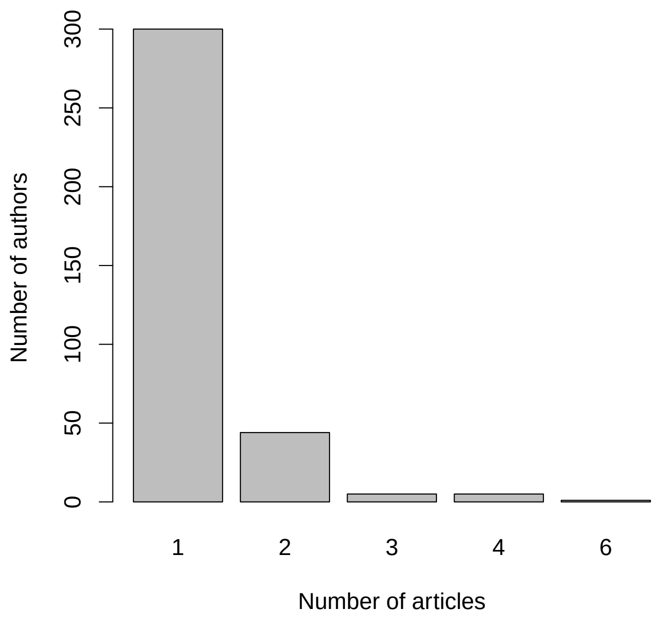

Figure 6.3: Number of articles per authors

## 6.4 Bilingual clustering

In the following the results of applying the bilingual mixture model are discussed. A few different kinds of models were computed. As the simulations in section 5.5 indicate it might be beneficial to reduce to complexity of the clustering problem. In this case the complexity was reduced by removing terms from the document-term-matrices. One approach is to remove terms with very small tf-idf values. Alternatively one could simply remove terms whose absolute term frequencies were either very small or very big. After the removal of these terms by either method one could still choose to weight the document-term-frequencies by tf-idf as outlined in section 6.2. Table 6.2 gives an overview of the models computed and the choices made as to the selection criteria for terms and shows the resulting number of dimensions in columns $d_x$ and $d_y$. Models (a) and (b) only retained terms with tf-idf of at least 0.12 which is approximately the median of all tf-idf values, thus keeping about half the terms. Models (c) and (d) only selected terms that appeared at least 3 times in the complete corpus and at most 100 times.

| model | selection | weighting | $d_x$ | $d_y$ |
|---|---|---|---|---|
| (a) | tf-idf $\geq 0.12$ | tf | 1173 | 1096 |
| (b) | tf-idf $\geq 0.12$ | tf-idf | 1173 | 1096 |
| (c) | tf $\geq 3$ & tf $\leq 100$ | tf | 902 | 714 |
| (d) | tf $\geq 3$ & tf $\leq 100$ | tf-idf | 902 | 714 |

Table 6.2: Overview of different types of models computed.

Firstly however one needs to choose the number of clusters $k$. A popular approach with mixture models is to calculate models with different values of $k$ chosen from a predefined range, e.g. $k \in \{1, \ldots, 30\}$, and then choose the model which exhibits the lowest BIC value (Fraley and Raftery, 2000). This seemed not feasible in this application however, because the number of dimensions is much larger than the number of observations. The BIC penalizes extra parameters and for an extra component over 1000 extra parameters have to be estimated, thus the BIC criterion did always favor the model with exactly 1 component even on simulated data where the true number of components was known to be larger than 1.

An alternative is to regard $k$ as a hyper-parameter to be chosen by cross-validation (Grün and Hornik, 2011). In 10-fold cross-validation the dataset is divided into 10 parts, where in each cross-validation iteration a test data set is chosen as one of those 10 parts and a training set is comprised of the remaining 9 parts. Then one estimates the model on the training data and with the resulting parameter estimates a function assessing the model fit is evaluated on the test data. A model with hyper-parameter $k$ would generalise well over the training data if the model also "fits well" to the test data. How good a model describes the test data is in this case determined by the perplexity (Newman et al., 2009) on one hand and on the other hand by the median of the individual log-likelihood values for each observation in the test data set.

**Definition 8** (Perplexity)**.** Let $\ell(x, y|\theta, \psi, \alpha)$ be the log-likelihood function of a model evaluated for a data set $(x, y)$ and parameters $(\theta, \psi, \alpha)$ then the perplexity is defined as:

$$\text{Perplexity} := \exp\left(-\frac{1}{n}\ell(x, y|\theta, \psi, \alpha)\right)$$

The perplexity is a monotonous transformation of the log-likelihood. Then in essence the perplexity assumes lower values when the log-likelihood is large. Low values of the perplexity are considered "good".

The results of the cross-validation for model (c) are shown in figure 6.4. For each $k \in \{1, \dots, 30\}$ there were 10 perplexity values and 10 median log-likelihood values obtained. These values are plotted as one solid line for each fold. One fold refers to one combination of training and test data sets. The prediction on a test data set usually depends on the fold, that is, certain combinations of training and test data sets perform better than others, therefore the perplexity and median log-likelihood values were centered around 0 to correct for this "fold-effect" (the 0 is indicated by a dashed horizontal line). The second row of graphics shows the same values in form of boxplots. A LOESS curve was added to these plots to visualize the trend of the perplexity and median log-likelihood values. For small values of $k$ the perplexity tends to decline and then starts to increase again for $k \geq 10$ thus indicating that models with around 8–10 components perform the best. The median log-likelihood values seem to mirror the trend of the perplexities, however reaching the maximum for smaller values of $k$ around 5 components. While perplexities and median log-likelihoods do not seem to be in perfect agreement in the following a model with $k = 9$ components is described. The third row of figure 6.4 shows AIC and BIC values for models with $k$ components. The AIC values reach their minimum at $k = 7$ while the BIC values only tend to increase with increasing number of components.

Tables 6.3 and 6.4 list the (stemmed) terms with the highest rankings for model (c) and 9 components. The weighting shown next to the words is determined by summing over all row-normalised documents in the respective clusters. From the tabulation of highest ranked word one would expect to see

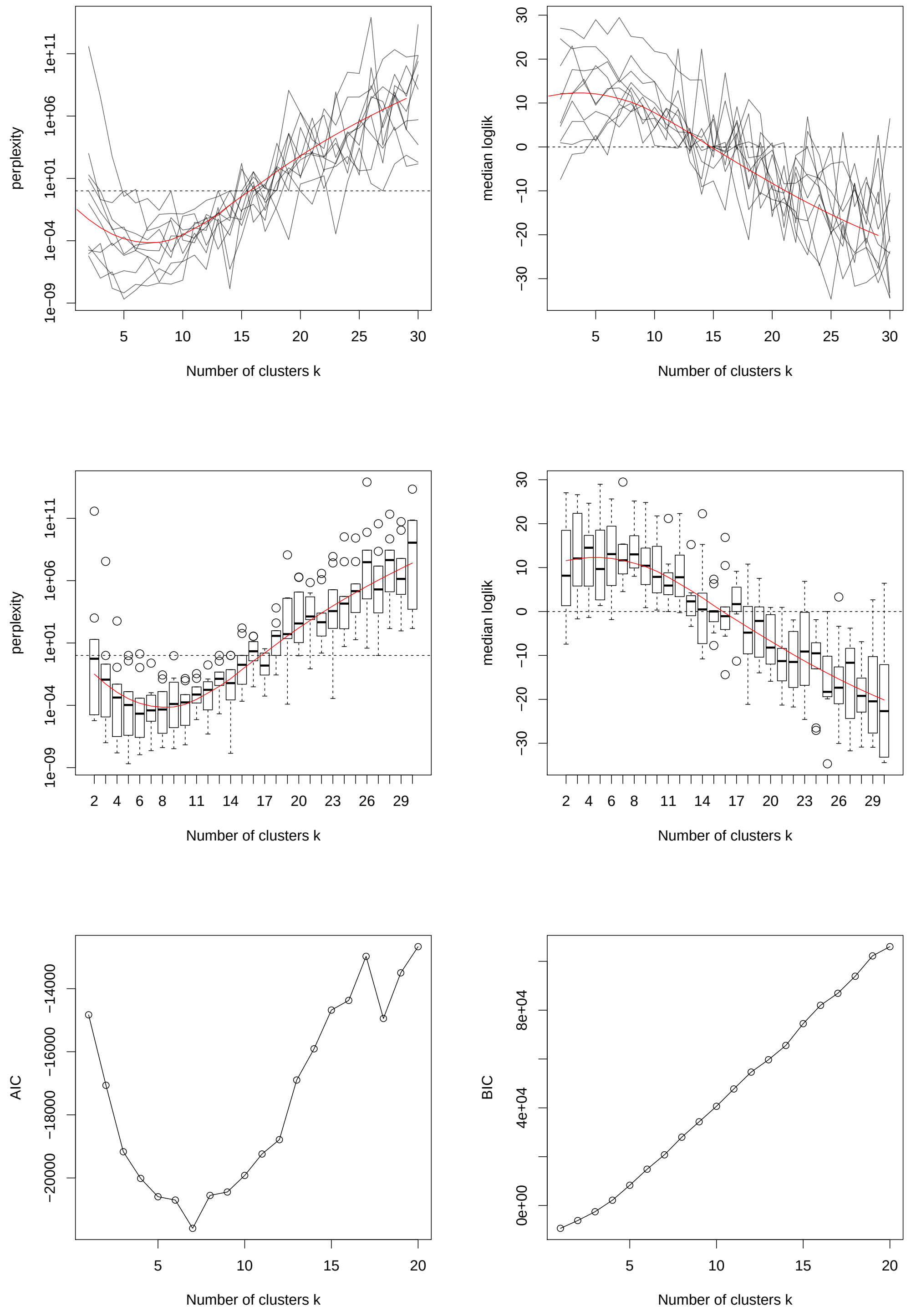

Figure 6.4: Perplexity, median log-likelihood and AIC/BIC for model (c)

a clear division of certain topics among the clusters. So that for example one might conclude that a certain cluster contains documents primarily concerned with spatial statistics. This is not the case however. Even though a lot of the terms were removed there are still terms ranked high such as “test”, “sampl[e]” and “stud[y]”, which occur in multiple clusters and thus do not seem to discriminate well between clusters. Also the english and german clusters should “match”, that is, from the highest ranking english and german terms it should be discernible that the respective documents share the same topics. Again, this does not seem to be the case.

Table 6.5 shows a few statistics for each component of the model under consideration. Each component was roughly assigned the same number of documents as seen from the estimated prior class probabilities $\hat{\alpha}_h$ and the number of assigned docments to cluster $n_h$. Only cluster 7 seems to deviate from this and contains a lot fewer documents. Consequently cluster 7 exhibits the lowest estimated variance (highest estimated concentration parameters) in both english and german documents. Apart from that, most $\kappa$ values lie in the range of 200–400 which is rather low.

The EM and DAEM algorithms are technically soft classification methods, from which the cluster assignment was determined by assigning the cluster for each observation which exhibited the highest estimated posterior class probability $p(h|x, y, \Theta)$. Then one should examine how ambiguous this assignment was. In this case, the highest posterior probabilities per observation were very close to 1, in fact, the lowest such observation was estimated to be 0.997. This means the observations were separable, which is not surprising, given the high-dimensional nature of the parameter space.

Another point worth of investigation would be, if document assignment to clusters depends on the availability of said documents in both languages. A possible scenario would have been, that all documents only available in english would have been assigned to certain clusters while documents available in both languages would have been assigned to another set of clusters.

The (contingency) table 6.6 shows for each cluster the amount of documents either available in only one language or in both languages. Intuitively documents are distributed independent from language availability over clusters, This suspicion is confirmed by the $\chi^2$ test for indepence, which yields a p-value of 0.63.

As for the models (b) and (d) in tf-idf weighting, figures B.2 and B.4 in appendix B show that the models only deteriorate with higher number of components, and are thus discarded since no reasonable choice on the amount of clusters could be made. In other terms, these plots indicate that there were no “good” clustering solutions found for those models. For model (a) the perplexity and median log-likelihood values obtained from the cross-validation as shown in figure B.1 are not in agreement, the median log-likelihood values are non-increasing and are thus not favoring any number of components. The results for this model are not shown as well for being equally undeclarative as in the other models.

| Cluster 1 | | Cluster 2 | | Cluster 3 | | Cluster 4 | | Cluster 5 | |
|---|---|---|---|---|---|---|---|---|---|
| test | (3.5) | qualiti | (2.1) | paramet | (2.3) | develop | (1.5) | base | (1.4) |
| studi | (2.4) | base | (1.9) | process | (1.9) | studi | (1.5) | sampl | (1.3) |
| consid | (2.2) | sampl | (1.6) | base | (1.6) | test | (1.2) | random | (1.1) |
| paramet | (2.0) | applic | (1.5) | robust | (1.3) | process | (1.2) | studi | (1.0) |
| approach | (2.0) | paramet | (1.5) | random | (1.2) | result | (1.1) | bay | (1.0) |
| base | (1.9) | test | (1.4) | observ | (1.2) | popul | (1.1) | use | (1.0) |
| propos | (1.8) | use | (1.3) | consid | (1.2) | paramet | (1.0) | function | (1.0) |
| function | (1.8) | result | (1.1) | function | (1.1) | use | (0.9) | provid | (1.0) |
| procedur | (1.8) | assess | (1.1) | propos | (1.1) | analysi | (0.9) | design | (1.0) |
| independ | (1.7) | studi | (1.1) | variabl | (1.0) | base | (0.9) | margin | (1.0) |
| gleichzeit | (1.8) | charakterist | (1.2) | emh | (1.0) | gestalt | (1.4) | geschloss | (1.0) |
| univariat | (1.7) | datenintegration | (1.1) | steu | (0.8) | anfang | (1.2) | kerndichteschätz | (1.0) |
| spezialfäll | (1.6) | lernalgorithm | (0.9) | model | (0.7) | hypothes | (1.0) | hauptsächlich | (1.0) |
| erzielt | (1.1) | shrinkag | (0.9) | approximativ | (0.7) | version | (1.0) | schachspiel | (0.9) |
| null | (1.0) | abgleich | (0.8) | dispersion | (0.7) | lernalgorithmus | (0.9) | interaktiv | (0.9) |
| amtlich | (1.0) | excel | (0.8) | folgend | (0.7) | moran | (0.9) | konstrui | (0.9) |
| selektiv | (0.9) | leicht | (0.8) | geograph | (0.7) | republ | (0.9) | zählung | (0.8) |
| beeinfluss | (0.9) | bangladesh | (0.8) | schrittweis | (0.7) | balakrishnan | (0.9) | kumulativ | (0.8) |
| inklusionswahrschein | (0.8) | geschloss | (0.7) | schwierig | (0.7) | erlaub | (0.8) | median | (0.8) |
| modellbasier | (0.8) | reflektiert | (0.7) | shrinkag | (0.7) | dispersion | (0.8) | beeinfluss | (0.7) |

Table 6.3: Top words for model (c) for clusters 1–5

| Cluster 6 | | Cluster 7 | | Cluster 8 | | Cluster 9 | |
|---|---|---|---|---|---|---|---|
| sampl | (2.2) | sampl | (1.1) | propos | (1.9) | test | (1.8) |
| propos | (1.7) | regist | (0.5) | test | (1.9) | paramet | (1.7) |
| time | (1.4) | cell | (0.5) | function | (1.1) | studi | (1.7) |
| develop | (1.3) | base | (0.5) | observ | (1.1) | sampl | (1.5) |
| studi | (1.3) | test | (0.5) | studi | (1.1) | measur | (1.5) |
| random | (1.2) | popul | (0.5) | consid | (1.1) | probabl | (1.4) |
| survey | (1.2) | type | (0.4) | use | (1.0) | result | (1.3) |
| result | (1.1) | studi | (0.4) | sampl | (1.0) | error | (1.3) |
| popul | (1.1) | fuzzi | (0.4) | approach | (1.0) | mean | (1.3) |
| function | (1.1) | minimum | (0.4) | process | (1.0) | class | (1.2) |
| darstell | (1.3) | weibull | (0.9) | entwickeln | (1.2) | design | (1.5) |
| kind | (1.0) | einschätzung | (0.8) | jänner | (1.2) | aufbau | (1.0) |
| likelihood | (1.0) | entwickeln | (0.4) | fixiert | (1.0) | weist | (1.0) |
| veränder | (0.9) | erfass | (0.4) | erkenn | (0.8) | umfangreich | (0.9) |
| zeil | (0.9) | erweiter | (0.4) | existenz | (0.7) | endlich | (0.8) |
| loglinear | (0.9) | hazardfunktion | (0.4) | populationsmittel | (0.7) | caussinus | (0.8) |
| metadat | (0.8) | liegt | (0.4) | changepoint | (0.7) | variiert | (0.8) |
| wien | (0.8) | zeitpunkt | (0.4) | gaussverteil | (0.7) | coxaal | (0.8) |
| anpass | (0.7) | führt | (0.3) | ausreiß | (0.6) | gewicht | (0.7) |
| bleib | (0.7) | maßnahm | (0.3) | beding | (0.6) | doubl | (0.7) |

Table 6.4: Top words for model (c) for clusters 6–9

| Cluster | $\hat{\alpha}_h$ | $n_h$ | $\hat{\kappa}_{xh}$ | $\hat{\kappa}_{yh}$ |
|---|---|---|---|---|
| 1 | 0.16 | 36 | 254.6 | 135.1 |
| 2 | 0.14 | 31 | 372.3 | 142.5 |
| 3 | 0.12 | 26 | 341.2 | 130.5 |
| 4 | 0.10 | 23 | 412.2 | 333.2 |
| 5 | 0.10 | 21 | 681.5 | 122.2 |
| 6 | 0.11 | 25 | 466.5 | 121.5 |
| 7 | 0.03 | 6 | 1102.1 | 789.9 |
| 8 | 0.12 | 26 | 332.2 | 238.2 |
| 9 | 0.12 | 26 | 478.7 | 336.3 |

Table 6.5: Estimated prior class probabilities $\alpha_h$, number of documents assigned to each cluster $n_h$, and estimated concentration parameters $\kappa_{xh}$ and $\kappa_{yh}$ for each component in model (c).

| Cluster | only english | english and german |
|---|---|---|
| 1 | 14 | 22 |
| 2 | 13 | 18 |
| 3 | 12 | 14 |
| 4 | 11 | 12 |
| 5 | 7 | 14 |
| 6 | 5 | 20 |
| 7 | 2 | 4 |
| 8 | 10 | 16 |
| 9 | 12 | 14 |

Table 6.6: Contingency table of documents assigned to clusters versus the availability of german abstracts.

## 6.5 Unilingual clustering

For a comparison with the bilingual model (c), a unilingual model with 9 components was estimated. This model was computed using the bilingual framework and the DAEM algorithm by marking the german corpus as missing ($c_y = 0$). Thus on the german corpus all components are modeled as uniform on the unit hypersphere, making it equivalent to a unilingual model[8].

The cluster assignments between the uni- and bilingual models do not seem to be in agreement of one another (NMI=0.072). Tables 6.7 and 6.8 show the highest ranked words by cluster in the unilingual model. These results appear to be a bit more interpretable, and the words displayed from one cluster seem to focus more on a common topic. Example terms that seem to fit together are "compon[ent]" and "mixture" (from mixture models) in cluster 1, "effect", "regress[ion]" and "predict" in cluster 5 and "test" and "hypothes[is]" in cluster 7. On the other hand, cluster 6 e.g. still seems hard to interpret. Overall the unilingual model appears to be performing better than the bilingual model.

Certain statistics of this model are shown in table 6.9. In this model the documents are fairly well distributed over all clusters as the estimated prior class probabilities $\alpha_h$ and number of documents per cluster $n_h$ show. Notably there is no single class which is very sparsely populated in contrast to the bilingual model. The estimated concentration parameters $\kappa_h$ are higher on average, for clusters 5 and 9 assuming values of close under 300 and over 400 for the reminaing clusters. Lower variance means also that observations are more homogenous within each cluster. Also, the highest estimated posterior class probabilities per observation was at least 1, hence the classifications were unambiguous in this case as well.

The result may be a bit surprising that the unilingual model achieved better results than the bilingual one, even though the latter was able to utilize more data. As the simulation in section 5.3 indicate a low ratio of german vs english documents might deteriorate the estimations. In this application for about 61% of documents there was a german abstract available, which might be too low, or at least on the low end of what is helpful for the model. As all simulations also indicate is, that there was always a near-optimal partition of the data, which could not always be approached by the estimation procedure, due to getting trapped in local maxima or stationary fixed points. Such effects might also get emphasized by the fact that the bilingual model has to find a solution in a much larger parameter space induced by the existence

[8]This was also empirically verified against the movMF package (Hornik and Grün, 2012) using the EM ("softmax") algorithm and same initialization values.

of the german docment-term-matrix. Another possibility might be that the assumption of conditional independence between english and german texts is violated. However at this point no conclusive evidence either supporting or refuting this hypothesis was found.

| Cluster 1 | | Cluster 2 | | Cluster 3 | | Cluster 4 | | Cluster 5 | |
|---|---|---|---|---|---|---|---|---|---|
| random | (23.0) | sampl | (43.0) | propos | (27.0) | function | (44.0) | effect | (22.0) |
| independ | (17.0) | probabl | (21.0) | margin | (16.0) | base | (19.0) | use | (20.0) |
| process | (17.0) | varianc | (17.0) | tabl | (14.0) | densiti | (18.0) | incom | (17.0) |
| compon | (10.0) | studi | (14.0) | mean | (12.0) | general | (14.0) | regress | (17.0) |
| mixtur | (9.0) | properti | (10.0) | base | (11.0) | skew | (14.0) | variabl | (17.0) |
| function | (8.0) | simul | (10.0) | consid | (10.0) | applic | (13.0) | approach | (15.0) |
| price | (8.0) | design | (9.0) | measur | (10.0) | compar | (13.0) | design | (15.0) |
| prove | (8.0) | random | (9.0) | edit | (9.0) | observ | (13.0) | predict | (15.0) |
| variabl | (8.0) | compar | (8.0) | equal | (9.0) | paramet | (12.0) | base | (14.0) |
| deriv | (7.0) | rate | (8.0) | categori | (8.0) | propos | (12.0) | seri | (14.0) |

Table 6.7: Top words for the unilingual movMF model clusters 1–5

| Cluster 6 | | Cluster 7 | | Cluster 8 | | Cluster 9 | |
|---|---|---|---|---|---|---|---|
| regist | (30.0) | test | (78.0) | paramet | (56.0) | algorithm | (21.0) |
| develop | (23.0) | studi | (14.0) | error | (26.0) | develop | (17.0) |
| popul | (22.0) | base | (12.0) | base | (18.0) | learn | (13.0) |
| qualiti | (22.0) | hypothes | (12.0) | sampl | (18.0) | spatial | (10.0) |
| match | (19.0) | consid | (11.0) | time | (17.0) | forecast | (9.0) |
| busi | (18.0) | result | (11.0) | propos | (16.0) | process | (9.0) |
| econom | (14.0) | approach | (10.0) | compar | (13.0) | tax | (9.0) |
| link | (14.0) | propos | (10.0) | consid | (13.0) | analysi | (8.0) |
| process | (13.0) | time | (10.0) | measur | (13.0) | domain | (8.0) |
| record | (13.0) | error | (9.0) | procedur | (12.0) | use | (8.0) |

Table 6.8: Top words for the unilingual movMF model clusters 6–9

| Cluster | $\hat{\alpha}_h$ | $n_h$ | $\hat{\kappa}_{xh}$ |
|---|---|---|---|
| 1 | 0.09 | 20 | 403.4 |
| 2 | 0.08 | 17 | 557.0 |
| 3 | 0.08 | 17 | 463.4 |
| 4 | 0.13 | 28 | 403.8 |
| 5 | 0.18 | 39 | 296.5 |
| 6 | 0.11 | 25 | 395.1 |
| 7 | 0.10 | 23 | 667.7 |
| 8 | 0.11 | 24 | 523.7 |
| 9 | 0.12 | 27 | 274.9 |

Table 6.9: Estimated prior class probabilities $\alpha_h$, number of documents assigned to each cluster $n_h$, and estimated concentration parameters $\kappa_{xh}$ and $\kappa_{yh}$ for each component in the unilingual model.

# Chapter 7

# Conclusions

In this thesis an extension to the movMF model by Banerjee et al. (2005) was presented, with the goal of being able to model parallel text corpora in two different languages at the same time. This was accomplished by using the joint distribution of two von Mises-Fisher distribution as the component distributions. Thus it is assumed the dependence between the two languages can be explained by the cluster membership alone, thus the vMF distributions for one document are conditionally independent given the cluster assignments. Furthermore a small addition was made so that documents available only in one language can be modeled simultaneously as well, these documents' distribution is assumed to be uniform where they were unobserved. The assumption of conditional independence also greatly simplifies the model and parameter estimators. The latter can be reduced to the unilingual case which is particularly helpful for the numerically difficult estimation of the concentration parameters.

An implementation was developed for this model using two different estimation algorithms, the EM and DAEM algorithms. The EM algorithm might get stuck in stationary fixed points, which is attempted to be alleviated by the DAEM algorithm which adds an annealing step to the estimation which can be envisioned to be a smoothing of the objective function which is successively reduced until the objective function is equal in the final iteration to that of the EM algorithm.

Testing this implementation a number of simulation studies were conducted, which were designed to be of similar dimensionality and number of observations as the application data set so that information about the performance of this model, relevant to the application might be revealed. The simulations showed that the DAEM algorithm consistently outperformed the EM algorithm except in relatively easy problems, such as large datasets or small variances, where both algorithms produced results of about equal quality. Also for the dataset size of the application the simulations showed a rather poor performance of both algorithms. Further the simulations indicated that if less than half of the articles were available in both languages the performance was below that of the unilingual model, meaning that only a small amount of extra data in a second language is detrimental to the model estimates.

As an application the corpus of abstracts of the Austrian Journal of Statistics (AJS) was used, which were in many cases published in two languages. These abstracts first had to be retrieved and extracted from the PDF files from the journal homepage, for which the processing steps were also explained. The application was then analyzed using the bilingual movMF model, where the data set was reduced in dimensionality using two different schemes, by selecting terms by term frequency or by the tf-idf scores. Additionally it was reweighted using the tf-idf score as well as left unweighted to produce a total of four models. For these models a reasonable number of components was determined via cross-validation where the tf-idf weighted models however did not appear to be appropriate. The clustering of one of the remaining models was then tabulated by highest term scores by clusters, which did not produce a descriptive result. In contrast, the equivalent unilingual model seemed to produce better results than the bilingual one.

Possible reasons for the nonsuccess of the bilingual model might be that overall sample size was too low to find reasonable estimates, which was emphasized by the bilingual parameter space being larger than the unilingual one. Possibly the assumption of conditional independence did not hold in this application. For the future it might be worth an investigation to explicitly model the dependence between two languages, for which copulas are a very generic tool. Deriving analytical derivatives of copulas in high dimensions might prove difficult however, for example the `copula` package in `R` (Yan, 2007) only provides derivatives of up to 10 dimensions. Another avenue of investigation might be to use variants of the von Mises-Fisher distribution which allow the specification of a covariance structure, which would most likely mean even more parameters would have to be estimated, which will be a challenging problem given the already small sample size. Last but not least the model should be examined on a larger data set in order to find out whether in this case better results are obtained and if they measure up to, or even improve upon, the results of unilingual modeling.

# List of Figures

# List of Tables

# Bibliography

Rick L. Andrews and Imran S. Currim. Recovering and profiling the true segmentation structure in markets: an empirical investigation. *International Journal of Research in Marketing*, 20(2):177–192, 2003.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, December 2005.

David M. Blei and John D. Lafferty. Topic models. *Text Mining: Classification, Clustering, and Applications.*, 2009.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2), 2001.

Inderjit S. Dhillon and Suvrit Sra. Modeling data using directional distributions. Technical report, Department of Computer Sciences, The University of Texas at Austin, 2003.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

Ingo Feinerer, Kurt Hornik, and David Meyer. Text mining infrastructure in R. *Journal of Statistical Software*, 25(5):1–54, 2 2008. ISSN 1548-7660. URL http://www.jstatsoft.org/v25/i05.

Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. Technical report, Department of Statistics, University of Washington, 2000.

Joydeep Ghosh and Arindam Banerjee. Scalable clustering algorithms with balancing constraints. *Data Mining and Knowledge Discovery*, 13, 2006.

Bettina Grün and Kurt Hornik. topicmodels: An R package for fitting Topic Models. *Journal of Statistical Software*, 40(13):1–30, May 2011.

Kurt Hornik and Bettina Grün. On maximum likelihood estimation of the concentration parameter of von Mises-Fisher distributions. Report 120, Institute for Statistics and Mathematics, WU Wirtschaftsuniversität Wien, Research Report Series, October 2012. URL http://epub.wu.ac.at/3669/.

Kurt Hornik and Bettina Grün. *movMF: Mixtures of von Mises-Fisher Distributions*, 2012. URL http://CRAN.R-project.org/package=movMF. R package version 0.1-0.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit 05*, 2005.

Duncan Temple Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2012. URL http://CRAN.R-project.org/package=XML. R package version 3.95-0.1.

David J.C. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, 2005.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2009.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. *Journal of Machine Learning Research*, pages 1801–1828, 2009.

R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2011. URL http://www.R-project.org/. ISBN 3-900051-07-0.

Jörg Tiedemann. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria, 2009. ISBN 978 90 272 4825 1.

Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Naonori Ueda and Ryohei Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

Jun Yan. Enjoy the joy of copulas: with a package copula. *Journal of Statistical Software*, 21(4):1–21, 2007.

# Appendix A

# Bilingual parameter estimates

This chapter features the deductions of the parameter estimates in the bilingual model (equation 3.6–3.9). The deduction follows analogously to Banerjee et al. (2005, A.2).

## A.1 Prior class probabilities $\alpha_h$

The expected complete data log-likelihood (3.4) is to be optimized with respect to $\alpha_h$ under the constraint that $\sum_{h=1}^{k} \alpha_h = 1$. First, the lagrangian is formed

$$\Lambda = E_p \left[\ln P(x, y|\Theta)\right] + \lambda \left( \sum_{h=1}^{k} \alpha_h - 1 \right)$$

Taking the partial derivative with respect to $\alpha_h$ and solving for zero:

$$\begin{aligned} \frac{\partial \Lambda}{\partial \alpha_h} &= \sum_{i=1}^{n} \frac{1}{\alpha_h} p(h|x_i, y_i, \Theta) + \lambda \overset{!}{=} 0 \\ \Leftrightarrow \alpha_h &= -\frac{1}{\lambda} \sum_{i=h}^{n} p(h|x_i, y_i, \Theta) \end{aligned} \tag{A.1}$$

Taking the partial derivative with respect to $\lambda$:

$$\begin{aligned} \frac{\partial \Lambda}{\partial \lambda} &= \sum_{h=1}^{k} \alpha_h - 1 \overset{!}{=} 0 \\ \sum_{h=1}^{k} \alpha_h &= 1 \end{aligned} \tag{A.2}$$

Summing over equation A.1 for $h = \{1, \dots, k\}$ and using eq. A.2:

$$\begin{aligned}
\sum_{h=1}^{k} \alpha_h &= -\frac{1}{\lambda} \sum_{h=1}^{k} \sum_{i=1}^{n} p(h|x_i, y_i, \Theta) \\
1 &= -\frac{1}{\lambda} \sum_{i=1}^{n} \underbrace{\sum_{h=1}^{k} p(h|x_i, y_i, \Theta)}_{=1} \\
1 &= -\frac{1}{\lambda} \sum_{i=1}^{n} 1 \\
\lambda &= -n \qquad \text{(A.3)}
\end{aligned}$$

Substituting A.3 in A.1 one obtains:

$$\hat{\alpha}_h = \frac{1}{n} \sum_{h=1}^{k} p(h|x_i, y_i, \Theta)$$

## A.2 Mean direction $\mu_{xh}$

In the following the expected complete data log-likelihood (3.4) is optimized for $\mu_{xh}$ under the constraints $||\mu_{xh}|| = 1$ for all $h = 1, \dots, k$ (analogously for $\mu_{yh}$). The lagrangian is formed:

$$\Lambda = E_p\left[\ln P(x, y|\Theta)\right] + \lambda_h \left(1 - \sum_{h=1}^{k} \mu_{xh}^T \mu_{xh}\right)$$

Taking the partial derivate thereof with respect to $\mu_{xh}$ (note that the density $f_h(y_i|\psi_h)$ vanishes in the derivation step since it is independent from $\mu_{xh}$ and is thus omitted) and solving for 0:

$$\begin{aligned}
\frac{\partial \Lambda}{\partial \mu_{xh}} &= \frac{\partial}{\partial \mu_{xh}} \left[\sum_{i=1}^{n} c_{xi} \ln\Big(c_d(\kappa_{xh}) \exp\left(\kappa_{xh} \mu_{xh}^T x_i\right)\Big) p(h|x_i, y_i, \Theta) - \lambda_h ||\mu_{xh}|| + \lambda_h\right] \\
&= \sum_{i=1}^{n} c_{xi} \cdot \kappa_{xh} \cdot x_i \cdot p(h|x_i, y_i, \Theta) - 2\lambda_h \mu_{xh} \overset{!}{=} 0 \\
\Leftrightarrow \mu_{xh} &= \frac{\kappa_{xh}}{2\lambda_h} \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \qquad \text{(A.4)}
\end{aligned}$$

Taking the partial derivative with respect to $\lambda$ and solving for 0:

$$\frac{\partial \Lambda}{\partial \lambda} = 1 - \mu_{xh}^T \mu_{xh} \overset{!}{=} 0 \quad \Leftrightarrow \quad \mu_{xh}^T \mu_{xh} = 1 \tag{A.5}$$

Substituting (A.4) into (A.5):

$$\begin{aligned} 1 &= \Big|\Big| \frac{\kappa_{xh}}{2\lambda_h} \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \Big|\Big| \\ \lambda_h &= \frac{\kappa_{xh}}{2} \Big|\Big| \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \Big|\Big| \end{aligned} \tag{A.6}$$

And substituting (A.6) into (A.4) one obtains:

$$\hat{\mu}_{xh} = \frac{\sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta)}{\Big|\Big| \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \Big|\Big|} =: \frac{r_{xh}}{||r_{xh}||} \tag{A.7}$$

## A.3 Concentration parameter $\kappa_{xh}$

Firstly, the langrangian is formed. While the constraint $\kappa_{xh} \geq 0$ should be considered, Banerjee et al. (2005, footnote, page 1374) argue that the multiplier for the KKT conditions would be zero for $\kappa_{xh} > 0$ and for $\kappa_{xh} = 0$ the distribution would be the uniform distribution on the sphere.

$$\Lambda = E_p \left[ \ln P(x, y|\Theta) \right]$$

Deriving $\Lambda$ with respect to $\kappa_{xh}$ and setting it zero:

$$\begin{aligned} \frac{\partial \Lambda}{\partial \kappa_{xh}} &= \frac{\partial}{\partial \kappa_{xh}} \sum_{i=1}^{n} \sum_{h=1}^{k} c_{xi} \cdot \ln f_h(x_i|\theta_h) \cdot p(h|x_i, y_i, \Theta) \\ &= \frac{\partial}{\partial \kappa_{xh}} \sum_{i=1}^{n} \sum_{h=1}^{k} c_{xi} \Big( \ln c_d(\kappa_{xh}) + \kappa_{xh} \mu_{xh}^T x_i \Big) p(h|x_i, y_i, \Theta) \\ &= \sum_{i=1}^{n} c_{xi} \Big( \frac{c_d^{'}(\kappa_{xh})}{c_d(\kappa_{xh})} + \mu_{xh}^T x_i \Big) p(h|x_i, y_i, \Theta) \overset{!}{=} 0 \end{aligned}$$

A few more transformations and substituting (A.7) into (A.8)

$$\begin{aligned} \frac{c_d^{'}(\kappa_{xh})}{c_d(\kappa_{xh})} \sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta) &= - \sum_{i=1}^{n} c_{xi} \cdot \mu_{xh}^T \cdot x_i \cdot p(h|x_i, y_i, \Theta) \\ \frac{c_d^{'}(\kappa_{xh})}{c_d(\kappa_{xh})} &= - \frac{\sum_{i=1}^{n} c_{xi} \cdot \mu_{xh}^T \cdot x_i \cdot p(h|x_i, y_i, \Theta)}{\sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta)} \\ &= - \frac{\left|\left| \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \right|\right|}{\sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta)} \end{aligned} \tag{A.8}$$

With (Banerjee et al., 2005, page 1374)

$$-\frac{c_d^{'}(\kappa_{xh})}{c_d(\kappa_{xh})} = \frac{I_{d/2}(\kappa_{xh})}{I_{d/2-1}(\kappa_{xh})}$$

the ML-estimator for $\kappa_{xh}$ is written

$$\frac{I_{d/2}(\hat{\kappa}_{xh})}{I_{d/2-1}(\hat{\kappa}_{xh})} = \frac{\left|\left| \sum_{i=1}^{n} c_{xi} \cdot x_i \cdot p(h|x_i, y_i, \Theta) \right|\right|}{\sum_{i=1}^{n} c_{xi} \cdot p(h|x_i, y_i, \Theta)} \tag{A.9}$$

# Appendix B

# Bilingual model plots

This chapter shows several plots for each type of model from section 6.4. As a remainder the model types were:

| model | selection | weighting | $d_x$ | $d_y$ |
|---|---|---|---|---|
| (a) | tf-idf $\geq 0.12$ | tf | 1173 | 1096 |
| (b) | tf-idf $\geq 0.12$ | tf-idf | 1173 | 1096 |
| (c) | tf $\geq 3$ & tf $\leq 100$ | tf | 902 | 714 |
| (d) | tf $\geq 3$ & tf $\leq 100$ | tf-idf | 902 | 714 |

On each page plots are shown for one type of model. The first two rows show the perplexity and median log-likelihood values obtained by cross-validation. Perplexity and median log-likelihood values were centered around 0 to alleviate the "fold"-effect. The third row shows AIC and BIC values for models with 2 to 20 components.

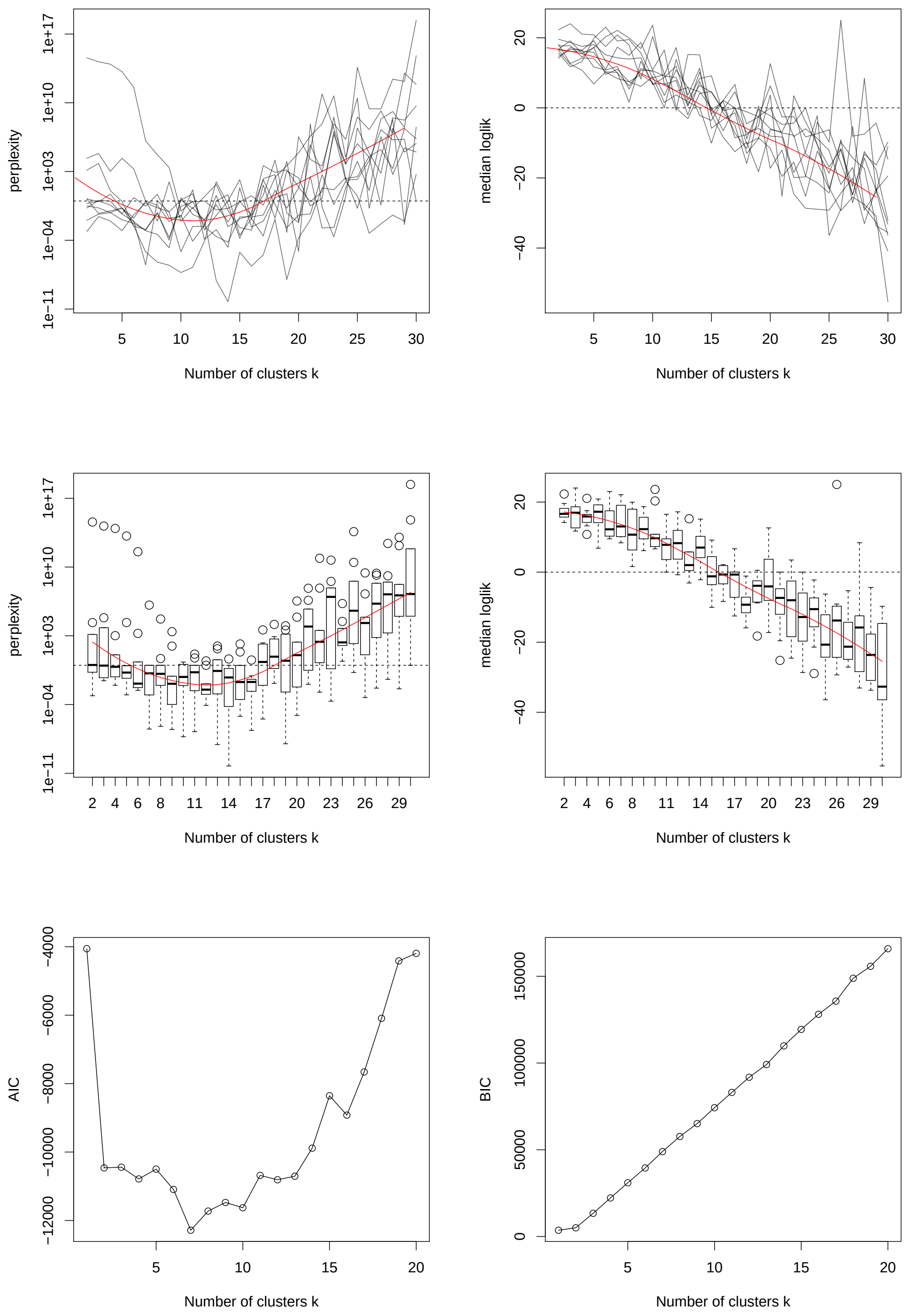

Figure B.1: Perplexity, median log-likelihood and AIC/BIC for model (a)

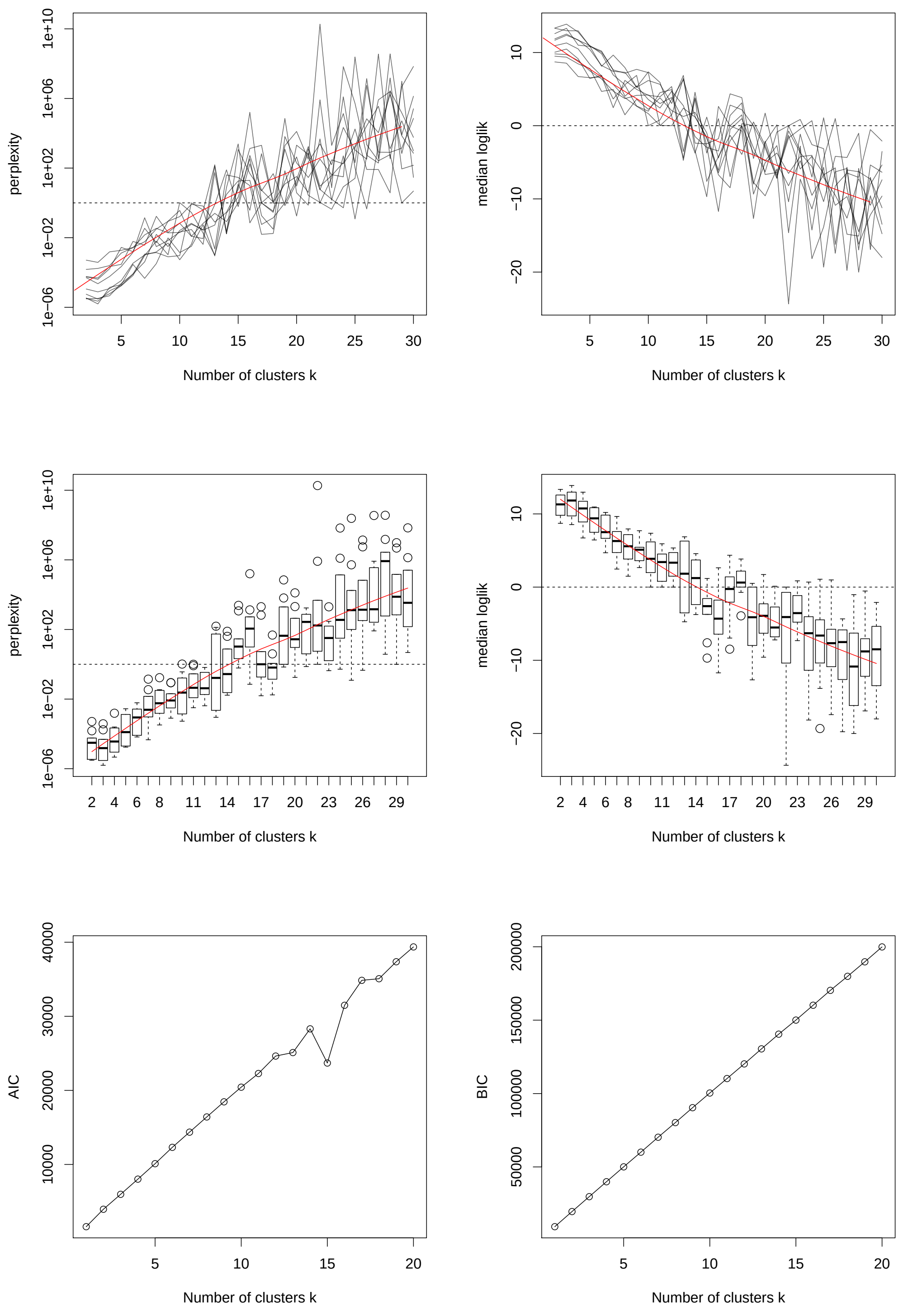

Figure B.2: Perplexity, median log-likelihood and AIC/BIC for model (b)

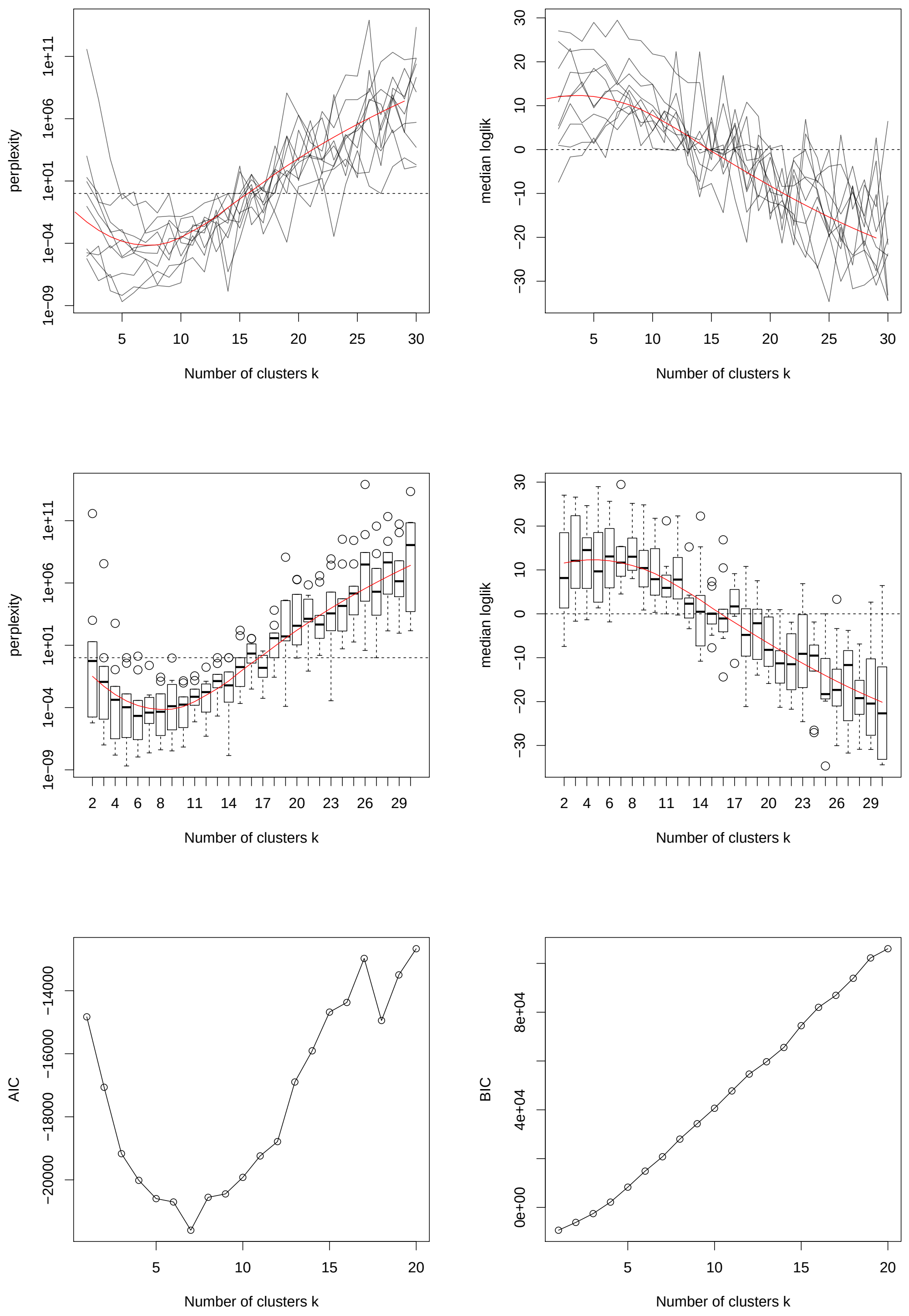

Figure B.3: Perplexity, median log-likelihood and AIC/BIC for model (c)

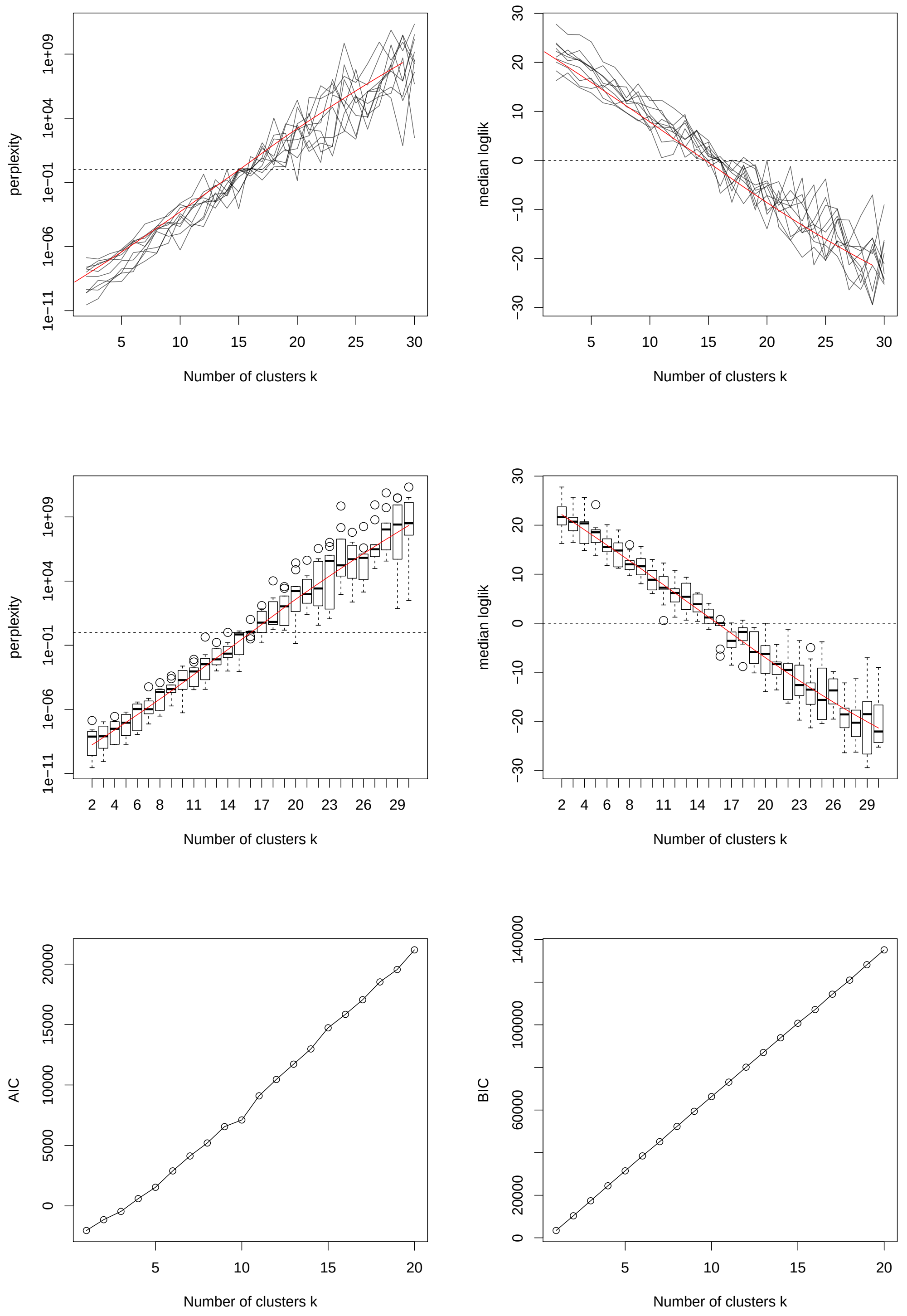

Figure B.4: Perplexity, median log-likelihood and AIC/BIC for model (d)

# Appendix C

# Electronic Appendix

The program files and simulation results are supplied in form of an electronic appendix on optical media. The software used was R in version 2.15.2 (R Development Core Team, 2011). Table C.1 also lists the R packages that were used.

| Package | Version |
|---|---|
| bimovMF | 0.01 |
| clue | 0.3-45 |
| doMC | 1.2.5 |
| knitr | 0.9 |
| movMF | 0.1-0 |
| multicore | 0.1-7 |
| plyr | 1.8 |
| Rstem | 0.4-1 |
| skmeans | 0.2-3 |
| slam | 0.1-26 |
| stringr | 0.6.2 |
| tm | 0.5-8 |
| XML | 3.95-0.1 |
| xtable | 1.7-0 |

Table C.1: Tables of used R packages

Following is a short overview of the contents of the electronic appendix. The root directory of the accompanying CD contains for one, this thesis in an electronic and in print version: `thesis_electronic.pdf` and `thesis_paper.pdf`. Also the bimovMF package is contained on the CD as tarball `bimovMF_0.01.tar.gz`. The directory `track` contains the output of `GNU httrack` containing the the download of the AJS website, that is, mostly the articles in PDF format, and also the HTML conversions of said articles by `PDFBox`. `pkg` contains the source of the bimovMF package, which contains most of R code used in this thesis. Under `thesis` are the sources located used to compile this thesis. Finally `data` lists the data sets, such as `abstracts.RData` containing a data-frame of abstracts, and the simulation and cross-validation results as well as the model estimates. Most program files also assume that the current working directory of the R interpreter is set to the `thesis` directory and assumes the directory structure is the same as on the electronic appendix CD. The thesis can be compiled using GNU make in the `thesis` directory: `make bibref paper`.

## C.1 The bimovMF package

The bimovMF package includes most of the program code developed during this thesis. It depends on packages from multiple repositories so the easiest way to install it is:

```
setRepositories(addURLs=c(dernst="http://dernst.org/R"))
install.packages("bimovMF")
```

and select the CRAN, omegahat and dernst repositories in the first dialog box. Alternatively the dependencies may be install by hand:

```
pkgs <- c("multicore", "movMF", "clue", "plyr", "skmeans")
pkgs <- c(pkgs, "slam", "stringr", "tm", "XML")
install.packages(pkgs)
install.packages("Rstem", repos="http://www.omegahat.org/R")
install.packages("bimovMF", repos="http://dernst.org/R")
```

The main function of the package is `bimovMF`, which is used to fit a bilingual movMF model to two data sets $x$ and $y$. Following is a list of options that can be used:

- `x`: A document-term-matrix in either dense or sparse matrix format (for the latter, only slam's simple triplet matrices are supported).
- `y`: A document-term-matrix analogous to `x`, with the same number of rows, or alternatively `NULL`. Also, individual rows may be set to `NA`.
- `k`: Number of components $k$ to be estimated.
- `P`: Initial $P$ matrix of posterior class probabilities (used as initialization). Might be supplied as list of such matrices in order to fit multiple

models with different initializations from which the one with the highest log-likelihood value is chosen. Such a list can be generated with `bimovMF:::make_start_P(n,k,Nruns)`.

- `maxiter`: Maximum amount of iterations for the EM algorithm. May be `NULL`.
- `nu`: Parameter $\nu$ to avoid numerical overflows in the $\kappa$ estimation. Should be rather small, for example `nu=0.01`.
- `minalpha`: Lower bound for $\alpha_h$. $\alpha_h$ values falling below this value have their components removed. 0 by default.
- `E`: Specifies the algorithm to be used, may be either "softmax" for EM or "DAEM".
- `converge`: Boolean, whether the algorithm should run until the convergence criterion is met, independent of the setting of `maxiter`. Only applicable to softmax/EM.
- `beta0` and `betarate`: Parameters controlling the annealing process of the DAEM algorithm. `beta0` is the initial $\beta$ value and `betarate` the multiplicative rate at which $\beta$ is increased.
- `.lapply` An implementation of lapply; a viable option would be `mclapply` from the multicore package, which would then compute the model runs in parallel.
- `di`: For internal use; tracks certain debug information throughout the estimation procedure.

The return value of `bimovMF` is named list containing the following elements:

- `P`: Estimated $P$ matrix.
- `theta` and `psi`: Estimated $\theta$ and $\psi$ matrices for all components.
- `alpha`: Estimated prior class probabilities $\alpha_h$.
- `niter`: Number of iterations used.
- `L`: A likelihood-object at the final iteration step.

The `bimovMF` function tries itself not to modify any global state, such as the global random number generator state, therefore initial parameter values are not automatically generated and have to be supplied by the user. It was found that this design helped with the parallelization and reproducibility of the estimation function.

The abstracts are extracted using the function `load_all_abstracts_from_html` which itself calls `parse_abstract_html` which are both found in `pkg/bimovMF/R/extract.R`.

For documentation purposes the package further includes functions used to conduct the simulations, those functions are found in `pkg/bimovMF/R/sim.R` and are named `simulation_by_kappa`, `simulation_ndim` (number of dimensions), `simulation_nobs` (by sample size) and `simulation_partial` (simulation on incomplete data sets). Functions for batch-processing are found in `pkg/bimovMF/R/batch.R`, for example `batch_sim` runs all simulations with seeds in one go. Analogously `batch_cv` runs the cross-validation of the models and `batch_movmf` runs the model estimates for the four kinds of models. For more details the reader is refered to the source files in question.

A short example to the usage of the `bimovMF` function:

```
> library(bimovMF)
> library(multicore)
> set.seed(667)
> dta <- rbimovMF_from_kappa(n=100, k=4, kappa=400, dx=100, dy=100)
> P <- make_start_P(n=100, k=4, Nruns=25)
> mod <- bimovMF(dta$x,dta$y, k=4, E="daem", P=P, .lapply=mclapply)

> NMI(dta$z, apply(mod$P,1,which.max))
[1] 1
> table(dta$z, apply(mod$P,1,which.max))
     1  2  3  4
  1  0  0  0 19
  2  0 26  0  0
  3  0  0 28  0
  4 27  0  0  0
```

First, a dataset with 100 observations is generated with $k = 4$ components, arbitrary parameters, but fixed $\kappa_{xh} = \kappa_{yh} = 400$ and 100 dimensions per language. Then starting values are generated for 25 iterations, then the 25 models are estimated in parallel using `mclapply` and the DAEM algorithm. `dta$z` contains the true cluster assignments; then the NMI of true and estimated clusters values is computed to be 1. Finally the contingency table between true and estimated cluster assignments is printed showing that all 100 observations are correctly classified, apart from label switching.