Clustering demo for Cixiao

Ben

2023-11-30

Libraries

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
library(tm)
```

```
## Loading required package: NLP
```

```
set.seed(1)
```

Simulating data for testing methodology

Defining a dictionary

With the code below I define a corpus just by taking a sentence from a recent BBC article.

```
bbctext <- "In Moscow men holding reindeer antlers above their heads are dancing around a stage. It is a
bbctext <- removePunctuation(bbctext)
corpus <- unique(strsplit(bbctext, split = " ")[[1]])
n_corpus <- length(corpus)
```

I now simulate some data for N users and M clusters/groups.

```
N <- 128
M <- 3
simmed_data <- data.frame(usernames = 1:N, cluster = sample.int(M,
    size = N, replace = T), forecast = NA, comment = NA)
head(simmed_data)
```

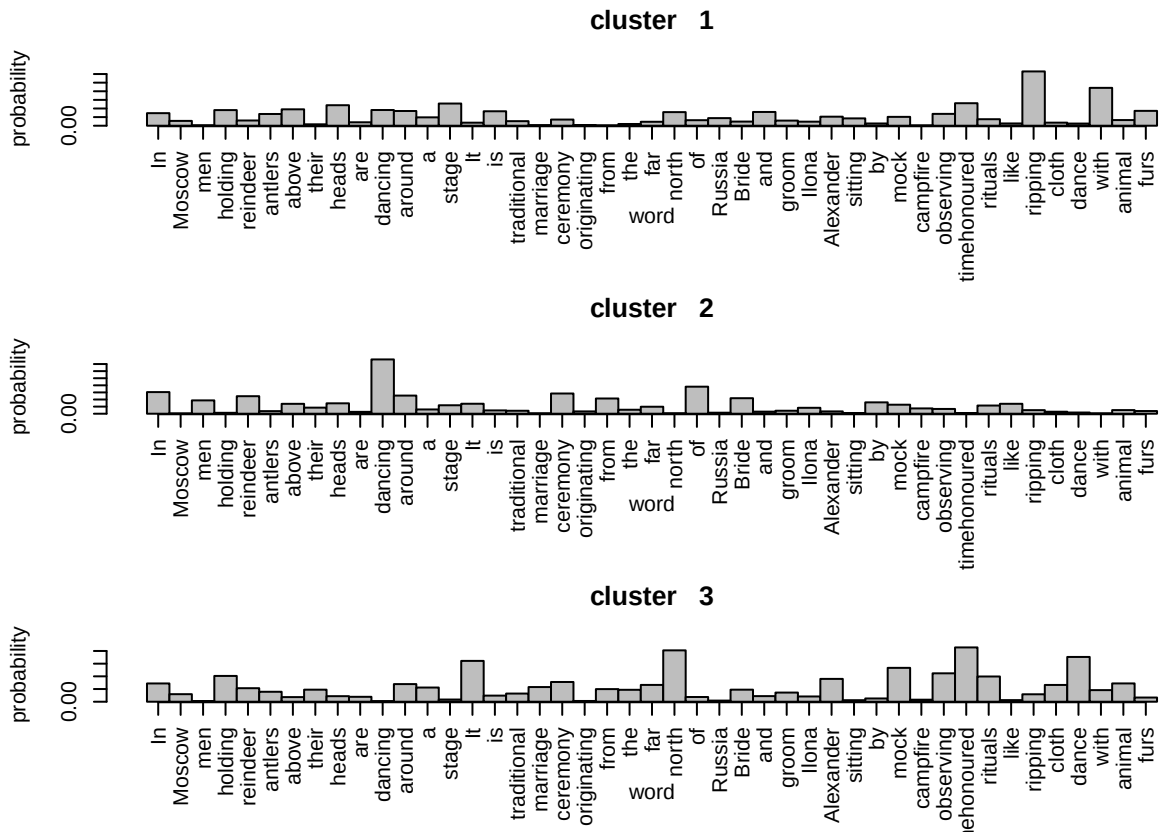
```
##   usernames cluster forecast comment
## 1         1       1       NA      NA
## 2         2       3       NA      NA
## 3         3       1       NA      NA
## 4         4       2       NA      NA
## 5         5       1       NA      NA
## 6         6       3       NA      NA
```

I pretend that each forecaster produces a numerical estimate/forecast. These estimates are random variables whose expectation depends on the forecaster's cluster.

```
forecast_means <- seq(-1, 1, length = M) * 2
for (m in 1:M) {
    ind <- which(simmed_data$cluster == m)
    simmed_data$forecast[ind] <- rnorm(length(ind), forecast_means[m],
        1)
}
```

I now simulate M distributions over words from a Dirichlet distribution. I simulate a comment for each user using words drawn from the distribution for their cluster.

```
text_probs <- matrix(rgamma(n_corpus * M, 1, 1), n_corpus, M)
text_probs <- apply(text_probs, 2, function(x) {
    x/sum(x)
})
par(mfcol = c(3, 1))
for (m in 1:M) {
    barplot(text_probs[, m], ylab = "probability", xlab = "word",
        main = paste("cluster  ", m), space = 0)
    axis(1, at = 1:n_corpus - 0.5, labels = corpus, las = 2)
}
```




```
comment_length <- 40
for (i in 1:N) {
    simmed_data$comment[i] <- paste(sample(corpus, size = comment_length,
        replace = T, prob = text_probs[, simmed_data$cluster[i]]),
        collapse = " ")
}

head(simmed_data)
```

```
##   usernames cluster  forecast
## 1         1       1 -1.227858
## 2         2       3  1.705804
## 3         3       1 -2.219516
## 4         4       2  1.580092
## 5         5       1 -2.424810
```

```
## 6         6       3  1.385050
##
## 1   of are furs Bride with timehonoured furs holding dancing ripping In antlers heads far Alexander
## 2 rituals reindeer like mock Ilona of Moscow Alexander Bride their cloth In mock timehonoured timeho
## 3                                          antlers antlers reindeer heads It dance ripping groom ritu
## 4                                                  of Bride dancing around reindeer dancing
## 5                                                       stage with timehonoured around obs
## 6           Alexander traditional north dance north around Alexander dance holding In mock Alexander
```

Recovering the cluster numbers

I now make a copy of the simulated data but I remove the cluster numbers, which we will try to recover using the forecasts and the comments.

```
simmed_data2 <- simmed_data
simmed_data2$cluster <- NA
head(simmed_data2)
```

```
##   usernames cluster  forecast
## 1         1      NA -1.227858
## 2         2      NA  1.705804
## 3         3      NA -2.219516
## 4         4      NA  1.580092
## 5         5      NA -2.424810
## 6         6      NA  1.385050
##
## 1   of are furs Bride with timehonoured furs holding dancing ripping In antlers heads far Alexander
## 2 rituals reindeer like mock Ilona of Moscow Alexander Bride their cloth In mock timehonoured timeho
## 3                                          antlers antlers reindeer heads It dance ripping groom ritu
## 4                                                  of Bride dancing around reindeer dancing
## 5                                                       stage with timehonoured around obs
## 6           Alexander traditional north dance north around Alexander dance holding In mock Alexander
```

We will estimate the cluster numbers using a Gibbs sampler (please read up on this method if you do not remember what it involves). We initialize the algorithm by simulating cluster numbers for the forecasters

```
simmed_data2$cluster <- sample.int(M, size = N, replace = T)
```

Then, conditioning on the simulated cluster numbers, we estimate the word distributions

```
alphamat <- matrix(1, n_corpus, M)
rownames(alphamat) <- corpus
colnames(alphamat) <- paste("cluster ", 1:M)
for (m in 1:M) {
    ind <- which(simmed_data2$cluster == m)
    word_counts <- table(factor(strsplit(paste(simmed_data2$comment[ind],
        collapse = " "), split = " ")[[1]], levels = corpus))
    alphamat[, m] <- 1 + word_counts
}
head(alphamat)
```

```
##          cluster  1 cluster  2 cluster  3
## In              82         54         60
## Moscow          17         12         18
## men             35         25         31
## holding         46         31         40
```

```
## reindeer         71         41         49
## antlers          40         19         22
```

The values in the alphamat matrix contain the alphas that parameterize the Dirichlet posterior for the word probabilities. We then sample from these posteriors and condition on these word probabilities

```
estimated_text_probs <- matrix(NA, n_corpus, M)
rownames(estimated_text_probs) <- corpus
colnames(estimated_text_probs) <- paste("cluster ", 1:M)
for (m in 1:M) {
    p <- rgamma(n_corpus, alphamat[, m], 1)
    estimated_text_probs[, m] <- p/sum(p)
}
head(estimated_text_probs)
```

```
##           cluster  1 cluster  2  cluster  3
## In       0.044046314 0.04423501 0.030409912
## Moscow   0.008085091 0.00616115 0.006908221
## men      0.012468042 0.01687544 0.017880492
## holding  0.025693469 0.02199155 0.026332224
## reindeer 0.043681282 0.02760372 0.027472931
## antlers  0.021501511 0.01608332 0.011639965
```

We also compute posterior expectations for the forecasts for each cluster

```
estimated_forecast_means <- rep(0, M)
for (m in 1:M) {
    ind <- which(simmed_data$cluster == m)
    estimated_forecast_means[m] <- rnorm(1, mean = mean(simmed_data2$forecast[ind]),
        sd = 1/(1 + length(ind)))
}
estimated_forecast_means
```

```
## [1] -1.95965602  0.03872276  2.02891158
```

Now we condition on the word distributions and forecast distributions for each cluster and simulate the cluster memberships for each user. This is another Bayesian calculation in which we compute the posterior distribution for cluster memberships, which is proportional to the prior (which for the time being I assumed is uniform) multiplied by the likelihood (please also make sure you are comfortable with this. It is very important).

```
cluster_log_probs <- matrix(NA, N, M)
rownames(cluster_log_probs) <- paste("user ", 1:N)
colnames(cluster_log_probs) <- paste("cluster ", 1:M)
for (i in 1:N) {
    word_count_i <- table(factor(strsplit(simmed_data2$comment[i],
        split = " ")[[1]], levels = corpus))
    for (m in 1:M) {
        cluster_log_probs[i, m] <- sum(word_count_i * log(estimated_text_probs[,
            m])) + dnorm(simmed_data2$forecast[i], estimated_forecast_means[m],
            log = T)
    }
}

cluster_log_probs <- t(apply(cluster_log_probs, 1, function(x) {
    x - max(x)
}))
```

```
cluster_probs <- t(apply(cluster_log_probs, 1, function(x) {
    exp(x)/sum(exp(x))
}))

for (i in 1:N) {
    simmed_data2$cluster[i] <- sample.int(M, 1, prob = cluster_probs[i,
        ])
}
head(cluster_probs)
```

```
##           cluster  1 cluster  2   cluster  3
## user  1 0.0287003989 0.97126714 3.246462e-05
## user  2 0.0003282205 0.07357177 9.261000e-01
## user  3 0.9199697525 0.08002655 3.692795e-06
## user  4 0.0002147832 0.01051158 9.892736e-01
## user  5 0.5762729730 0.42372639 6.361515e-07
## user  6 0.0521936139 0.01430221 9.335042e-01
```

Now we return to step in which we estimate the word distributions and forecast distribution conditional on the cluster memberships... and we repeat this many times. This takes about a minute.

```
N_its <- 2000
pb <- txtProgressBar(min = 0, max = N_its, initial = 0, style = 3)
for (its in 1:N_its) {
    setTxtProgressBar(pb, its)

    for (m in 1:M) {
        ind <- which(simmed_data2$cluster == m)
        word_counts <- table(factor(strsplit(paste(simmed_data2$comment[ind],
            collapse = " "), split = " ")[[1]], levels = corpus))
        alphamat[, m] <- 1 + word_counts
    }

    for (m in 1:M) {
        p <- rgamma(n_corpus, alphamat[, m], 1)
        estimated_text_probs[, m] <- p/sum(p)
    }

    for (m in 1:M) {
        ind <- which(simmed_data$cluster == m)
        estimated_forecast_means[m] <- rnorm(1, mean = mean(simmed_data2$forecast[ind]),
            sd = 1/(1 + length(ind)))
    }

    for (i in 1:N) {
        word_count_i <- table(factor(strsplit(simmed_data2$comment[i],
            split = " ")[[1]], levels = corpus))
        for (m in 1:M) {
            cluster_log_probs[i, m] <- sum(word_count_i * log(estimated_text_probs[,
                m])) + dnorm(simmed_data2$forecast[i], estimated_forecast_means[m],
                log = T)
        }
    }

    cluster_log_probs <- t(apply(cluster_log_probs, 1, function(x) {
```

```
        x - max(x)
    }))
    cluster_probs <- t(apply(cluster_log_probs, 1, function(x) {
        exp(x)/sum(exp(x))
    }))

    for (i in 1:N) {
        simmed_data2$cluster[i] <- sample.int(M, 1, prob = cluster_probs[i,
            ])
    }
}
save(simmed_data2, file = "simmed_data2.RData")
```

Now we can compare a simulated set of cluster memberships (simulated from our posterior distribution using the Gibbs sampler) and the true cluster memberships (which we know because we simulated them).

```
load(file = "simmed_data2.RData")
simmed_data2$cluster
```

```
##   [1] 2 3 2 1 2 3 3 1 1 3 3 2 2 2 1 1 1 1 3 2 3 2 2 2 2 1 2 2 1 1 1 2 3 2 3 1 1
##  [38] 1 1 3 1 2 3 1 2 2 3 1 1 3 3 1 1 1 1 2 1 1 1 1 2 3 3 1 3 3 1 3 3 2 2 2 2 3
##  [75] 1 3 2 2 1 2 2 2 2 3 1 2 2 3 3 3 1 1 1 3 1 1 3 3 3 2 1 1 2 3 3 1 3 1 2 1 2
## [112] 3 3 2 1 2 3 1 3 3 2 2 1 1 1 2 2 2
```

```
simmed_data$cluster
```

```
##   [1] 1 3 1 2 1 3 3 2 2 3 3 1 1 1 2 2 2 2 3 1 3 1 1 1 1 2 1 1 2 2 2 1 3 1 3 2 2
##  [38] 2 2 3 2 1 3 2 1 1 3 2 2 3 3 2 2 2 2 1 2 2 2 2 1 3 3 2 3 3 2 3 3 1 1 1 1 3
##  [75] 2 3 1 1 2 1 1 1 1 3 2 1 1 3 3 3 2 2 2 3 2 2 3 3 3 1 2 2 1 3 3 2 3 2 1 2 1
## [112] 3 3 1 2 1 3 2 3 3 1 1 2 2 2 1 1 1
```

We should see that the simulated clustering tends to put users in the same cluster if they really were from the same cluster. Obviously the label for the cluster might be different though.