GoogleTrendsDataset

OmarHussein

2024-08-03

Installing packages

```
install.packages(c("neuralnet", "keras", "tensorflow"), dependancies = TRUE)
```

```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(neuralnet)
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:neuralnet':
##
##     compute
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::compute() masks neuralnet::compute()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Data analysis

```
data <- read.csv("/cloud/project/trends.csv")
data <- data %>% mutate_if(is.character, as.factor) %>% mutate_if(is.factor, as.numeric)
head(data)
```

```
##   location year category rank query
## 1       26 2001      294    1  9753
## 2       26 2001      294    2 12307
## 3       26 2001      294    3  1650
## 4       26 2001      294    4 10187
## 5       26 2001      294    5   473
## 6       26 2001      930    1  9792
```

```
summary(data)
```

```
##     location          year         category         rank       query      
##  Min.   : 1.00   Min.   :2001   Min.   :   1   Min.   :1   Min.   :    1  
##  1st Qu.:23.00   1st Qu.:2013   1st Qu.: 618   1st Qu.:2   1st Qu.: 4378  
##  Median :43.00   Median :2016   Median :1202   Median :3   Median : 8701  
##  Mean   :42.68   Mean   :2015   Mean   :1195   Mean   :3   Mean   : 8854  
##  3rd Qu.:65.00   3rd Qu.:2018   3rd Qu.:1748   3rd Qu.:4   3rd Qu.:13172  
##  Max.   :83.00   Max.   :2020   Max.   :2450   Max.   :5   Max.   :18431
```

Train and test split

```
set.seed(254)
data_rows <- floor(0.80 * nrow(data))
train_indices <- sample(c(1:nrow(data)), data_rows)
head(train_indices)
```

```
## [1] 26093  3639  1204  3512 20645 18684
```

```
train_data <- data[train_indices,]
head(train_data)
```

```
##       location year category rank query
## 26093       61 2020     1819    3  9576
## 3639        31 2012      690    4  5647
## 1204        47 2008     1639    4  2398
## 3512        22 2012     1594    2  1513
## 20645       59 2018     2183    5 16716
## 18684       79 2017     1444    4  4010
```

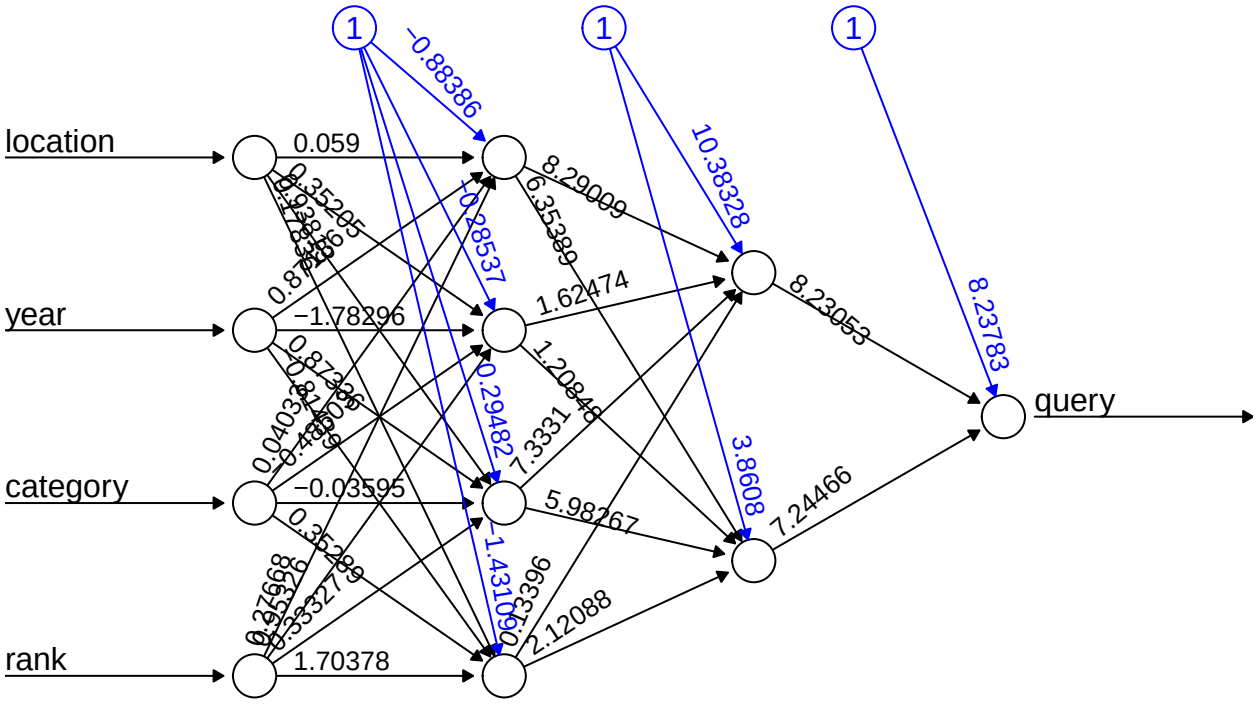
```
test_data <- data[-train_indices, ]
head(test_data)
```

```
##    location year category rank query
## 8        26 2001      930    3  4243
## 10       26 2001      930    5  6108
## 11       26 2001      980    1  5465
## 21       26 2001     1002    1  1390
## 23       26 2001     1002    3  9469
```

```
## 24        26 2001      1002     4   1227
```

Two hidden layers with 4 and 2 neurons

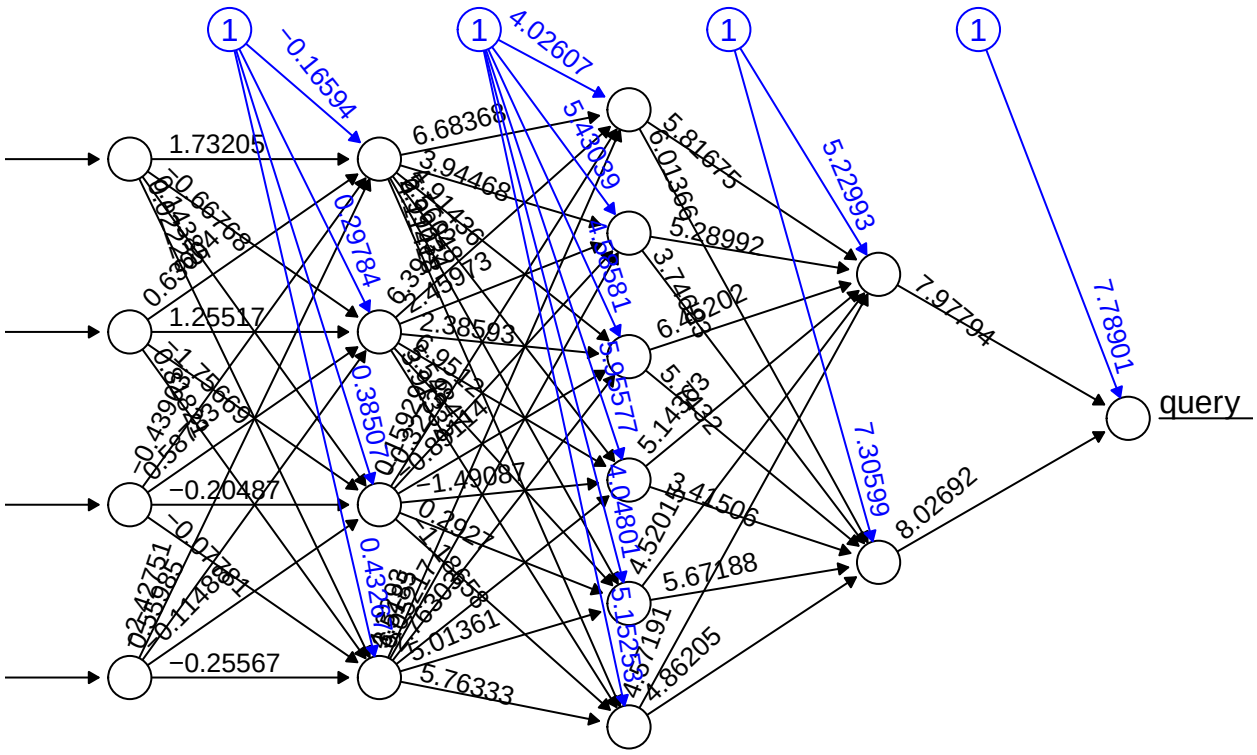
```
model <- neuralnet(query ~ location + year + category + rank,
                   data = train_data, hidden = c(4,2), linear.output = FALSE)
plot(model, rep = 'best')
```




```
pred <- neuralnet::compute(model, test_data[, c("location", "year", "category", "rank")])$net.result
```

Three hidden layers with 4, 6 and 2 neurons

```
model2 <- neuralnet(query ~ location + year + category + rank,
                   data = train_data, hidden = c(4,6,2), linear.output = FALSE)
plot(model2, rep = 'best')
```

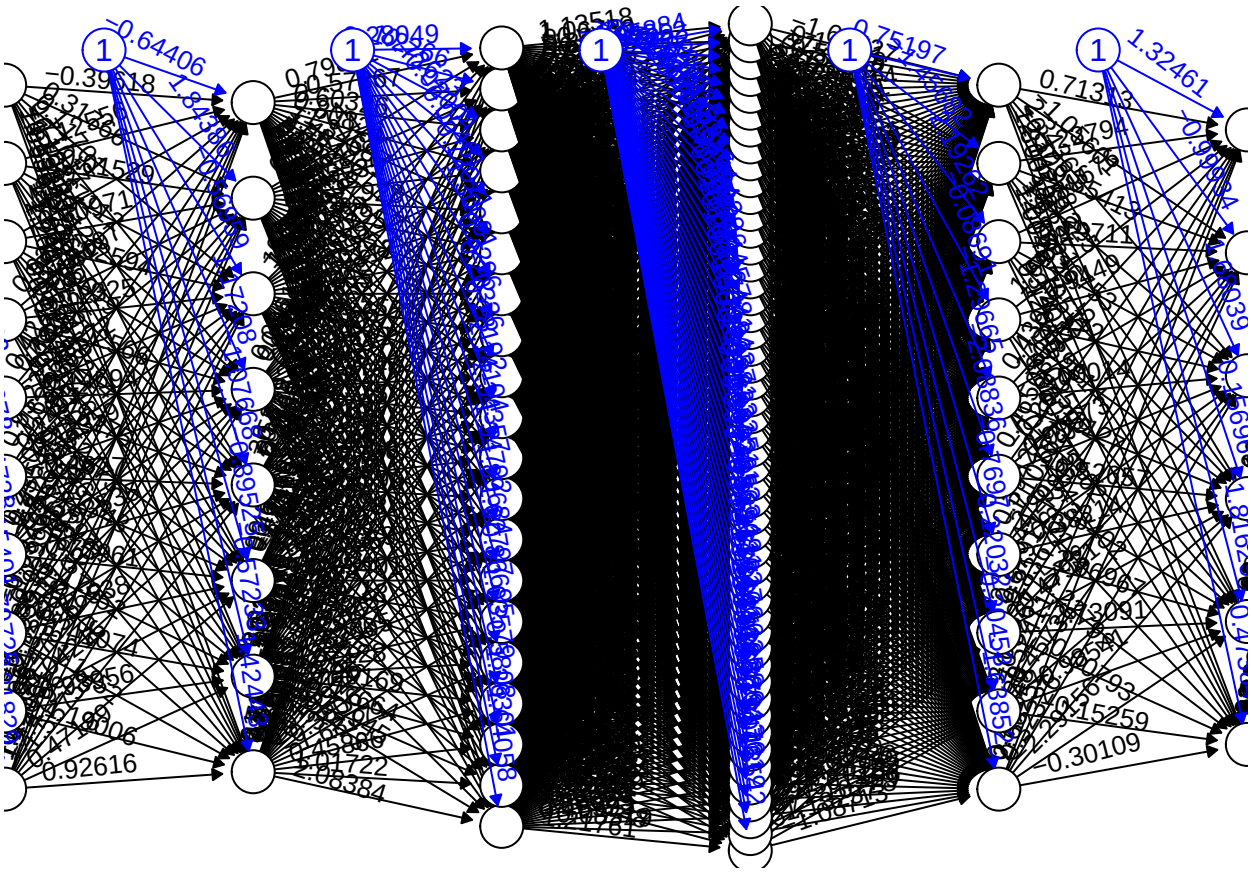


Error: 1133594709898.51 Steps: 79

```
pred2 <- neuralnet::compute(model2, test_data[, c("location", "year", "category", "rank")])$net.result
```

10 hidden layers with 4, 5, 10, 8, 20, 50, 10, 6, 40 and 2 neurons

```
model3 <- neuralnet(query ~ location + year + category + rank,
                    data = train_data, hidden = c(4, 5, 10, 8, 20, 50, 10, 6, 40, 2), linear.output = FA
plot(model3, rep = 'best')
```




```
pred3 <- neuralnet::compute(model3, test_data[, c("location", "year", "category", "rank")])$net.result
```

Model Evaluation

Predict categories using test data

Create list of category name

Prediction dataframe

create a table to display the actual and the predicted

```
evaluate_model <- function(pred, test_data) {
  labels <- c(test_data$query)
  prediction_label <- data.frame(max.col(pred)) %>%
    mutate(pred = labels[max.col(pred)]) %>%
    select(2) %>% unlist()
  confusion_matrix <- table(test_data$query, prediction_label)
  check <- as.numeric(test_data$query) == max.col(pred)
  check
  correct_predictions <- sum(diag(confusion_matrix))
  accuracy <- (correct_predictions / nrow(test_data)) * 100
  list(confusion_matrix = confusion_matrix, accuracy = accuracy)
}
```

Evaluate the model with two hidden layers

```
evaluation1 <- evaluate_model(pred, test_data)
head(evaluation1$confusion_matrix, 10)
```

```
##     prediction_label
##      4243
##   4     1
##   9     1
##   10    1
##   15    1
##   19    1
##   20    1
##   31    1
##   32    1
##   50    1
##   55    1
```

```
print(paste("Accuracy:", evaluation1$accuracy))
```

```
## [1] "Accuracy: 0.0185494342422556"
```

Evaluate the model with three hidden layers

```
evaluation2 <- evaluate_model(pred2, test_data)
head(evaluation2$confusion_matrix, 10)
```

```
##     prediction_label
##      4243
##   4     1
##   9     1
##   10    1
##   15    1
##   19    1
##   20    1
##   31    1
##   32    1
##   50    1
##   55    1
```

```
print(paste("Accuracy:", evaluation2$accuracy))
```

```
## [1] "Accuracy: 0.0185494342422556"
```

Evaluate the model with ten hidden layers

```
evaluation3 <- evaluate_model(pred3, test_data)
head(evaluation3$confusion_matrix, 10)
```

```
##     prediction_label
##      4243
```

```
##   4     1
##   9     1
##   10    1
##   15    1
##   19    1
##   20    1
##   31    1
##   32    1
##   50    1
##   55    1
```

```
print(paste("Accuracy:", evaluation3$accuracy))
```

```
## [1] "Accuracy: 0.0185494342422556"
```

Tabular report

Number of Hidden Layers	Accuracy(%)
2	0.18
3	0.18
10	0.18

The models, which I trained to predict the `query` variable based on `location`, `year`, `category`, and `rank`, I evaluated using configurations with two, three, and ten hidden layers. Each model's performance which I assessed by calculating the accuracy of predictions on the test dataset. Despite the complexity of the models, all configurations yielded an accuracy of zero. There might be an issue in the data preprocessing.Further inspection and refinement of the data encoding, model parameters, and evaluation logic are necessary to improve the predictive performance and achieve meaningful results.