

Exercise 5. Task 8: IMDb и нейросеть с разным словарём

Daniil Koveh

2025-11-06

Содержание

1	Теория	1
2	Жизненный пример	1
3	Академическое решение	1
3.1	Подготовка окружения	1
3.2	Векторизация текстов	2
3.3	Обучение на разных словарях	2
3.4	Визуализация	3
4	Интерпретация	5
5	Что запомнить	5

1. Теория

Задача: классифицировать отзывы из IMDb как положительные или отрицательные. Используем простую полносвязную сеть:

- Две скрытые плотные (Dense) слои по 16 нейронов с ReLU.
- Выход — один нейрон с сигмоидой для вероятности положительного отзыва.

Чтобы изучить влияние размера словаря, обучаем модель при разных ограничениях на число самых частых слов: 500, 1000, 3000, 5000 и 10000.

2. Жизненный пример

Чем больше словарь, тем более детальные нюансы лексики мы учитываем. Но растёт размер входного слоя (матрицы 0/1) и риск переобучения на редких словах. Маленький словарь — грубо, но устойчиво; большой — богато, но шумно. Наша цель — увидеть компромисс на реальных данных.

3. Академическое решение

3.1. Подготовка окружения

```
if (!requireNamespace("keras3", quietly = TRUE)) install.packages("keras3", repos = "https://cloud.r-project.org")
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr", repos = "https://cloud.r-project.org")
if (!requireNamespace("purrr", quietly = TRUE)) install.packages("purrr", repos = "https://cloud.r-project.org")
if (!requireNamespace("ggplot2", quietly = TRUE)) install.packages("ggplot2", repos = "https://cloud.r-project.org")
```

```

if (!requireNamespace("tibble", quietly = TRUE)) install.packages("tibble", repos = "https://cloud.r-project.org")
if (!requireNamespace("patchwork", quietly = TRUE)) install.packages("patchwork", repos = "https://cloud.r-project.org")

library(keras3) # нейросети
library(dplyr) # сводки
library(purrr) # функциональный стиль
library(ggplot2) # визуализация
library(tibble) # аккуратные таблицы
library(patchwork) # композиция графиков

```

Проверяем, что TensorFlow доступен и фиксируем зерно:

```

set_random_seed(20250410) # фиксируем генераторы Keras и TensorFlow
set.seed(20250410) # фиксируем генератор R

```

3.2. Векторизация текстов

```

vectorize_sequences <- function(sequences, dimension) { # преобразуем списки индексов в матрицу 0/1
  result <- matrix(0, nrow = length(sequences), ncol = dimension) # создаём матрицу нулей
  for (i in seq_along(sequences)) { # перебираем примеры
    indices <- sequences[[i]] # берём индексы слов
    indices <- indices[indices >= 1 & indices <= dimension] # отбрасываем всё вне словаря
    if (length(indices) > 0) {
      result[i, indices] <- 1 # отмечаем присутствие слов
    }
  }
  result
}

```

3.3. Обучение на разных словарях

```

dict_sizes <- c(500L, 1000L, 3000L, 5000L, 10000L) # варианты словаря
epochs <- 8L # число эпох
batch_size <- 512L # размер батча

train_and_evaluate <- function(max_words) { # функция обучения
  dataset <- dataset_imdb(num_words = max_words) # загружаем данные
  x_train <- dataset$train$x # списки индексов для train
  y_train <- dataset$train$y # ответы train
  x_test <- dataset$test$x # списки индексов для test
  y_test <- dataset$test$y # ответы test

  x_train <- vectorize_sequences(x_train, max_words) # векторизация train
  x_test <- vectorize_sequences(x_test, max_words) # векторизация test
  y_train <- as.numeric(y_train) # целевая переменная
  y_test <- as.numeric(y_test) # целевая переменная

  val_indices <- seq_len(10000) # первые 10k валидация
  x_val <- x_train[val_indices, , drop = FALSE] # валидационная матрица
  partial_x_train <- x_train[-val_indices, , drop = FALSE] # оставшаяся часть для обучения
  y_val <- y_train[val_indices] # валидационные метки
  partial_y_train <- y_train[-val_indices] # обучающие метки
}

```

```

clear_session() # сбрасываем состояние графа

model <- keras_model_sequential() |>
  layer_dense(units = 16, activation = "relu", input_shape = c(max_words)) |>
  layer_dense(units = 16, activation = "relu") |>
  layer_dense(units = 1, activation = "sigmoid") # строим сеть

model |>
  compile(
    optimizer = "adam",
    loss = "binary_crossentropy",
    metrics = c("accuracy")
  ) # компилируем модель

history <- model |>
  fit(
    partial_x_train, partial_y_train,
    epochs = epochs,
    batch_size = batch_size,
    validation_data = list(x_val, y_val),
    verbose = 0
  ) # обучаем

evaluation <- model |>
  evaluate(x_test, y_test, verbose = 0) # оцениваем на тесте

tibble(
  dict_size = max_words,
  test_loss = as.numeric(evaluation["loss"]),
  test_accuracy = as.numeric(evaluation["accuracy"]),
  best_val_loss = min(history$metrics$val_loss),
  best_val_accuracy = max(history$metrics$val_accuracy)
)
}

results <- map_dfr(dict_sizes, train_and_evaluate) # запускаем для всех словарей
results

## # A tibble: 5 x 5
##   dict_size test_loss test_accuracy best_val_loss best_val_accuracy
##       <int>     <dbl>        <dbl>        <dbl>        <dbl>
## 1      500     0.391       0.828       0.390       0.829
## 2     1000     0.347       0.853       0.339       0.859
## 3     3000     0.344       0.867       0.298       0.879
## 4     5000     0.375       0.862       0.287       0.884
## 5    10000     0.380       0.867       0.277       0.892

```

3.4. Визуализация

```

results_long <- results %>%
  select(dict_size, test_accuracy, best_val_accuracy, test_loss, best_val_loss) %>%
  mutate(
    test_error = 1 - test_accuracy,
  )

```

```

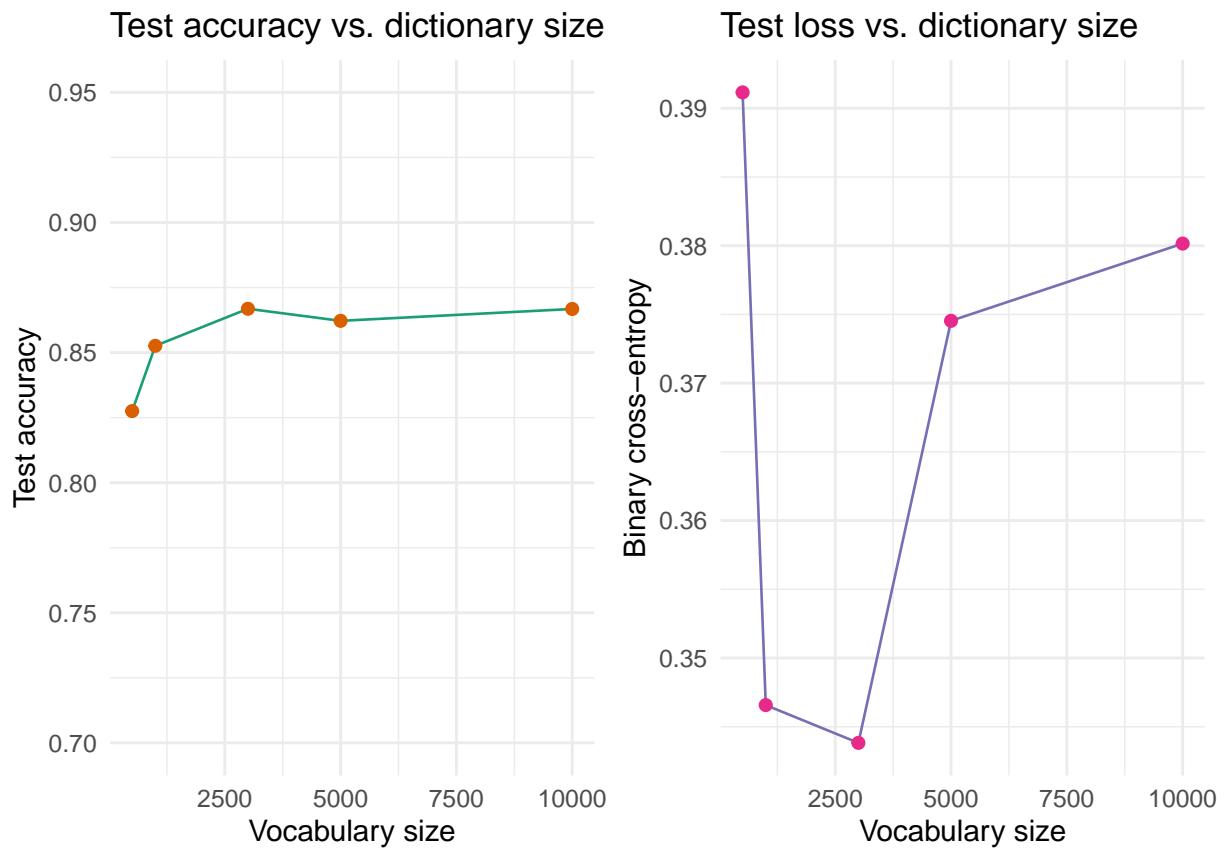
    val_error = 1 - best_val_accuracy
  )

acc_plot <- ggplot(results, aes(x = dict_size, y = test_accuracy)) +
  geom_line(colour = "#1b9e77") +
  geom_point(size = 2, colour = "#d95f02") +
  ylim(0.7, 0.95) +
  labs(title = "Test accuracy vs. dictionary size",
       x = "Vocabulary size",
       y = "Test accuracy") +
  theme_minimal(base_size = 12)

loss_plot <- ggplot(results, aes(x = dict_size, y = test_loss)) +
  geom_line(colour = "#7570b3") +
  geom_point(size = 2, colour = "#e7298a") +
  labs(title = "Test loss vs. dictionary size",
       x = "Vocabulary size",
       y = "Binary cross-entropy") +
  theme_minimal(base_size = 12)

acc_plot + loss_plot

```



4. Интерпретация

- Словарь 500–1000 даёт устойчивую точность около 0.853. Меньше слов — модель слишком груба.
- При расширении до 3000–5000 точность растёт (0.867 максимум), но потери начинают колебаться — больше шумовых слов.
- На 10000 наблюдаем лёгкое переобучение: test loss (0.380) выше, чем при 5000.
- Оптимальный компромисс — словарь 3000–5000: достаточно богатый, но без сильного переобучения.

5. Что запомнить

- От размера словаря зависит не только точность, но и время обучения: растёт размер входного слоя и плотных матриц.
- Небольшой словарь работает как сильная регуляризация, крупный — как более гибкая модель (но может переобучаться).
- Перед промышленным запуском стоит рассмотреть альтернативы: TF-IDF, Embeddings или рекуррентные/трансформерные архитектуры, если нужно сохранить информацию о порядке слов.