

Lecture 30:
Word Embedding & Intro to Deep Learning
Big Data and Machine Learning for Applied Economics
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

November 26, 2020

Announcements

- ▶ Problem Set 3 Grades are up (great job!!)
- ▶ Problem Set 4 is posted → *last class*
- ▶ Final Exam Date will be Friday Dec 11 8am to Sunday 13 8am
- ▶ Remember to turn in your project proposal by December 7

Recap: Text as Data

- ▶ Topic Models
- ▶ PCA Theory
- ▶ Factor Computation
- ▶ Factor Interpretation
- ▶ Latent Dirichlet Allocation (LDA): Example

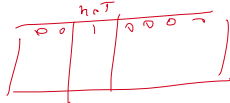
Agenda

- 1 Word Embedding
- 2 Deep Learning: Intro
 - Demo
- 3 Review & Next Steps
- 4 Further Readings

Word Embedding

- ▶ This is a new method that have come out of work in deep learning.
- ▶ Word embedding was originally motivated as a technique for dimension reduction on the inputs to a deep neural network.
- ▶ However, word embedding turns out to be valuable in its own right: it imposes a spatial structure on words, making it possible for those studying language to reason about distances between meanings and consider the algebra behind combinations of words in documents.

Word Embedding



Embedding

- ▶ In the original deep learning context, embedding layers replace each word with a vector value, such that, for example, hotdog becomes the location $[1, -5, 0.25]$ in a three-dimensional embedding space
- ▶ Compare this to the standard bag-of-words representation, where hotdog would be represented as a binary vector that is as long as there are words in the vocabulary, say, p .
- ▶ This binary vector will have $p-1$ zeros and a one in the hotdog dimension.
- ▶ The word embedding has translated the language representation from a large binary space to a smaller real-valued (and much richer) space.

Word Embedding

- ▶ There are a variety of different embedding algorithms—as many as there are different architectures for deep neural networks.
- ▶ The most common and general embeddings are built around word co-occurrence matrices.
- ▶ This includes the popular Glove and Word2Vec frameworks.
- ▶ What is co-occurrence?
 - ▶ Two words co-occur if they appear within the same sentence and within b words of each other. Where b is the “window size”
 - ▶ For a vocabulary size p , this leads to a sparse $p \times p$ co-occurrence matrix where each $[i, j]$ entry is the number of times that words i and j co-occur. Call this matrix C .
 - ▶ A word embedding algorithm seeks to approximate C as the product of two lower-dimensional matrices

Word Embedding

- ▶ A word embedding algorithm seeks to approximate C as the product of two lower-dimensional matrices

$$\underset{p \times p}{C} \approx \underset{p \times K}{UV'} \underset{K \times 1}{}$$

inner (dot) product

(1)

- ▶ Here, U and V are each $p \times K$ dimensional dense and real valued matrices.
- ▶ K is the dimension of the embedding space; hence, $K \ll p$ and both U and V are very tall and thin matrices.
- ▶ Each row of U and of V , u_j and v_j is then a K -dimensional embedding of the j th word.
- ▶ The implication is that these embeddings summarize the meaning of words as their inner product defines how much you expect them to co-occur.

Recall that the inner product is a standard measure of distance in linear algebra (e.g. $\underline{e'e}$)

Word Embedding

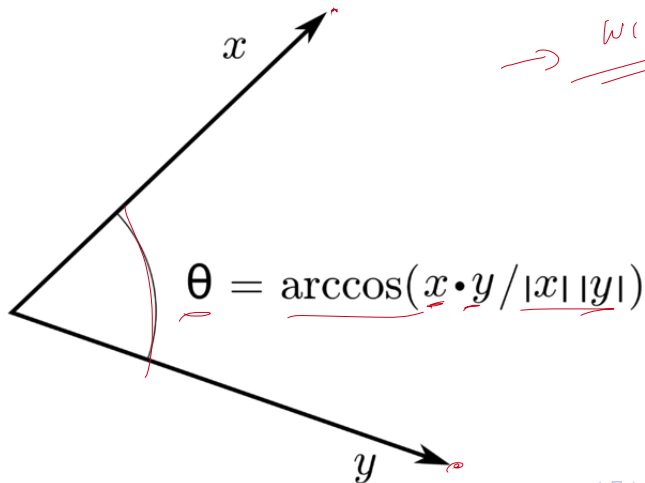
- ▶ One way to find U and V is to solve $C \approx UV'$ through the singular value decomposition (SVD).
- ▶ SVD is a factorization of a real or complex matrix, that serves for example, to find the eigenvalues and eigenvectors of square symmetric matrices (and hence in calculating principal components). (\sqrt{A})
- ▶ In practice, most of the software embedding solutions use alternatives to SVD that are designed to deal with the high amount of sparsity in C (since most words never co-occur in limited windows for standard corpora).
- ▶ Under many algorithms, especially when co-occurrence is symmetric, U and V will be mirror images of each other.
- ▶ Thus, it is standard to take one of these vectors u_j as the single embedding location for word j .

Word Embedding

- ▶ These locations were originally viewed as an intermediate output—as a processing step for inputs to a deep neural network.
- ▶ However, social scientists and linguists have discovered that the space of word locations contains rich information about the language of the documents used to train the embedding.
- ▶ Word embeddings preserve semantic relationships.
 - ▶ Words with similar meaning have similar representations.
 - ▶ Dimensions induced by word differences can be used to identify cultural concepts
- ▶ For example, the vector difference man - woman isolates a gender dimension in the space.
- ▶ The dimensions are useful because they produce quantitative measures of similarity between the associated concepts and specific words in the corpus.
- ▶ In this case, we can understand the gender connotation of a given word by taking the cosine of the angle between the vector representation of the word and the differenced vector representing the gender dimension (why?)

Word Embedding

- Recall the geometric interpretation of the angle between two vectors defined using an inner product



→ wikipedia !!!

Word Embedding

- ▶ Words with male connotations – e.g. male first names – are going to be positively correlated with $\text{man} - \text{woman}$.
- ▶ Female words, in turn, will be negatively correlated with the dimension.
- ▶ This framework provides an intuitive approach to measuring stereotypical associations in a given corpus.
- ▶ Bolukbasi et al (2016) is a nice example

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi¹, Kai-Wei Chang², James Zou², Venkatesh Saligrama^{1,2}, Adam Kalai²

¹Boston University, 8 Saint Mary's Street, Boston, MA


²Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and natural language processing tasks. We show that even word embeddings trained on Google News articles exhibit female/male gender stereotypes to a disturbing extent. This raises concerns because their widespread use, as we describe, often tends to amplify these biases. Geometrically, gender bias is first shown to be captured by a direction in the word embedding. Second, gender neutral words are shown to be linearly separable from gender definition words in the word embedding. Using these properties, we provide a methodology for modifying an embedding to remove gender stereotypes, such as the association between the words *receptionist* and *female*, while maintaining desired associations such as between the words *queen*

Word Embedding: Example 1

- ▶ They trained a standard word2vec embedding algorithm on the Google News corpora of news articles.
 - ▶ Then look at the differences between established gender words (for example, the vector for man minus the vector for woman, or father minus mother) to establish an axis in the embedding space that spans from masculinity to femininity.
 - ▶ They then calculate the location along this axis for a large number of terms that should be gender-neutral.
 - ▶ The embedding space has learned—from how the words are used in news articles—that these professions are stereotypically viewed as female and male occupations.
- 

Word Embedding: Example 1

Extreme <i>she</i>	Extreme <i>he</i>	Gender stereotype <i>she-he</i> analogies		
1. homemaker	1. maestro	sewing-carpentry	registered nurse-physician ✓	housewife-shopkeeper
2. nurse	2. skipper	nurse-surgeon ✓	interior designer-architect ✓	softball-baseball
3. receptionist	3. protege	blond-burly	feminism-conservatism ✓	cosmetics-pharmaceuticals
4. librarian	4. philosopher	giggle-chuckle	vocalist-guitarist	petite-lanky
5. socialite	5. captain	sassy-snappy	diva-superstar	charming-affable
6. hairdresser	6. architect	volleyball-football	cupcakes-pizzas ✓	lovely-brilliant
7. nanny	7. financier	Gender appropriate <i>she-he</i> analogies		
8. bookkeeper	8. warrior	queen-king	sister-brother	mother-father
9. stylist	9. broadcaster	waitress-waiter	ovarian cancer-prostate cancer	convent-monastery
10. housekeeper	10. magician			

Figure 1: **Left** The most extreme occupations as projected on to the *she-he* gender direction on w2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded. **Right** Automatically generated analogies for the pair *she-he* using the procedure described in text. Each automatically generated analogy is evaluated by 10 crowd-workers to whether or not it reflects gender stereotype.

Language from police body camera footage shows racial disparities in officer respect

Rob Voigt^{a,1}, Nicholas P. Camp^b, Vinodkumar Prabhakaran^c, William L. Hamilton^c, Rebecca C. Hetey^b, Camilla M. Griffiths^b, David Jurgens^c, Dan Jurafsky^{a,c}, and Jennifer L. Eberhardt^{b,1}

^aDepartment of Linguistics, Stanford University, Stanford, CA 94305; ^bDepartment of Psychology, Stanford University, Stanford, CA 94305; and ^cDepartment of Computer Science, Stanford University, Stanford, CA 94305

Contributed by Jennifer L. Eberhardt, March 26, 2017 (sent for review February 14, 2017; reviewed by James Pennebaker and Tom Tyler)

Using footage from body-worn cameras, we analyze the respectfulness of police officer language toward white and black community members during routine traffic stops. We develop computational linguistic methods that extract levels of respect automatically from transcripts, informed by a thin-slicing study of participant ratings of officer utterances. We find that officers speak with consistently less respect toward black versus white community members, even after controlling for the race of the officer, the severity of the infraction, the location of the stop, and the outcome of the stop. Such disparities in common, everyday interactions between police and the communities they serve have important implications for procedural justice and the building of police–community trust.

racial disparities | natural language processing | procedural justice | traffic stops | policing

some have argued that racial disparities in perceived treatment during routine encounters help fuel the mistrust of police in the controversial officer-involved shootings that have received such great attention. However, do officers treat white community members with a greater degree of respect than they afford to blacks?

We address this question by analyzing officers' language during vehicle stops of white and black community members. Although many factors may shape these interactions, an officer's words are undoubtedly critical: Through them, the officer can communicate respect and understanding of a citizen's perspective, or contempt and disregard for their voice. Furthermore, the language of those in positions of institutional power (police officers, judges, work superiors) has greater influence over the course of the interaction than the language used by those with less power (12–16). Measuring officer language thus provides a quantitative lens on one key aspect of the quality or tone of

Word Embedding: Example 3

Stereotypes in High-Stakes Decisions: Evidence from U.S. Circuit Courts

6/love

Elliott Ash, ETH Zurich

Daniel L. Chen, Toulouse School of Economics

Arianna Ornaghi, University of Warwick*

March 12, 2020

Abstract

Stereotypes are thought to be an important determinant of decision making, but they are hard to systematically measure, especially for individuals in policy-making roles. In this paper, we propose and implement a novel language-based measure of gender stereotypes for the high-stakes context of U.S. Appellate Courts. We construct a judge-specific measure of gender-stereotyped language use – *gender slant* – by looking at the linguistic association of words identifying gender (male versus female) and words identifying gender stereotypes (career versus family) in the judge’s authored opinions. Exploiting quasi-random assignment of judges to cases and conditioning on detailed biographical characteristics of judges, we study how gender stereotypes influence judicial behavior. We find that judges with higher slant vote more conservatively on women’s rights’ issues (e.g. reproductive rights, sexual harassment, and gender discrimination). These more slanted judges also influence workplace outcomes for female colleagues: they are less likely to assign opinions to female judges, they are more likely to reverse lower-court decisions if the lower-court judge is a woman, and they cite fewer female-authored opinions.

Word Embedding: Demo

```
library(text2vec)
load('shakes_words_df_4text2vec.RData')
head(shakes_words)
```

```
##           id      word
## 1 A_Lover_s_Complaint    nor
## 2 A_Lover_s_Complaint  gives
## 3 A_Lover_s_Complaint    it
## 4 A_Lover_s_Complaint satisfaction
## 5 A_Lover_s_Complaint    to
```

```
shakes_words_ls <- list(shakes_words$word)
it <- itoken(shakes_words_ls, progressbar = FALSE)
shakes_vocab <- create_vocabulary(it)
shakes_vocab <- prune_vocabulary(shakes_vocab, term_count_min= 5)
head(shakes_vocab)
```

```
## Number of docs: 1
## 0 stopwords: ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##      term term_count doc_count
## 1:  abess      5      1
## 2:  abilities  5      1
## 3:  accessory  5      1
## 4:    ace      5      1
## 5:  adders     5      1
```

Word Embedding: Demo

- ▶ The next step is to create the token co-occurrence matrix (TCM).
- ▶ The definition of whether two words occur together is arbitrary.

```
# maps words to indices
vectorizer <- vocab_vectorizer(shakes_vocab)
```

```
# use window of 10 for context words
shakes_tcm <- create_tcm(it, vectorizer, skip_grams_window = 10)
```

? 66 bel vectors

- ▶ Now we are ready to create the word vectors based on the GloVe model.

```
glove <- GlobalVectors$new(rank = 50, x_max = 10) ✓
shakes_wv_main = glove$fit_transform(shakes_tcm, n_iter = 10, convergence_tol = 0.01, n_threads = 8)
```

```
## INFO [16:55:06.317] epoch 1, loss 0.1242
## INFO [16:55:08.764] epoch 2, loss 0.0844
## INFO [16:55:11.249] epoch 3, loss 0.0762
## INFO [16:55:13.680] epoch 4, loss 0.0707
## INFO [16:55:16.109] epoch 5, loss 0.0666
## INFO [16:55:18.540] epoch 6, loss 0.0634
## INFO [16:55:20.980] epoch 7, loss 0.0609
## INFO [16:55:23.419] epoch 8, loss 0.0589
## INFO [16:55:25.849] epoch 9, loss 0.0572
## INFO [16:55:28.288] epoch 10, loss 0.0558
```

Word Embedding: Demo

```
dim(shakes_wv_main)
```

```
## [1] 9094 50
```

```
shakes_wv_context <- glove$components
```

```
dim(shakes_wv_context)
```

```
## [1] 50 9094
```

```
# Either word-vectors matrices could work, but the developers of the technique
```

```
# suggest the sum/mean may work better
```

```
shakes_word_vectors <- shakes_wv_main + t(shakes_wv_context)
```

```
rom <- shakes_word_vectors["romeo", , drop = F]
```

```
cos_sim_rom <- sim2(x =shakes_word_vectors, y = rom, method = "cosine", norm = "l2")
```

```
# head(sort(cos_sim_rom[,1], decreasing <- T), 10)
```

```
##      romeo      juliet      tybalt      nurse      benvolio      banished  
## 1.0000000 0.7712391 0.7575977 0.6697068 0.6517349 0.6436404
```

Word Embedding: Demo

```
test <- shakes_word_vectors["romeo", , drop = F] -  
  shakes_word_vectors["mercutio", , drop = F] +  
  shakes_word_vectors["nurse", , drop = F]  
  
cos_sim_test <- sim2(x = shakes_word_vectors, y = test, method = "cosine", norm = "l2")  
head(sort(cos_sim_test[,1], decreasing = T), 10)
```

nurse juliet romeo lady mother bed o wife
0.8904362 0.7584004 0.7179267 0.6440354 0.6374490 0.5880860 0.5756074 0.5638571
capulet dromio
0.5520459 0.5507196

Deep Learning: Intro

- ▶ Word Embedding uses neural nets, so let's turn to them
- ▶ Neural networks are simple models.
- ▶ Their strength lies in their simplicity because basic patterns facilitate fast training and computation.
- ▶ The model has linear combinations of inputs that are passed through nonlinear activation functions called nodes (or, in reference to the human brain, neurons).
- ▶ A set of nodes taking different weighted sums of the same inputs is called a layer, and
- ▶ The output of one layer's nodes becomes input to the next layer.

Deep Learning: Intro

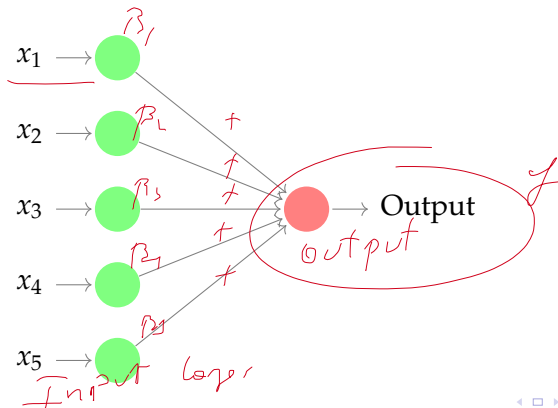
- Let's start with a familiar and simple model, the linear model

$$f = x\beta + u$$

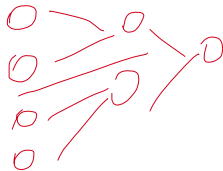
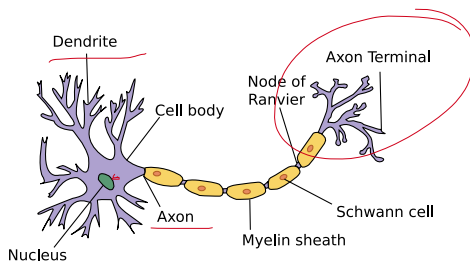
$$y = f(x) + u$$

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + u$$

(2)

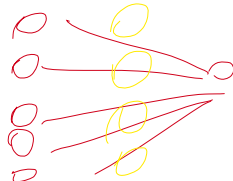


Deep Learning: Parallel to Biology?



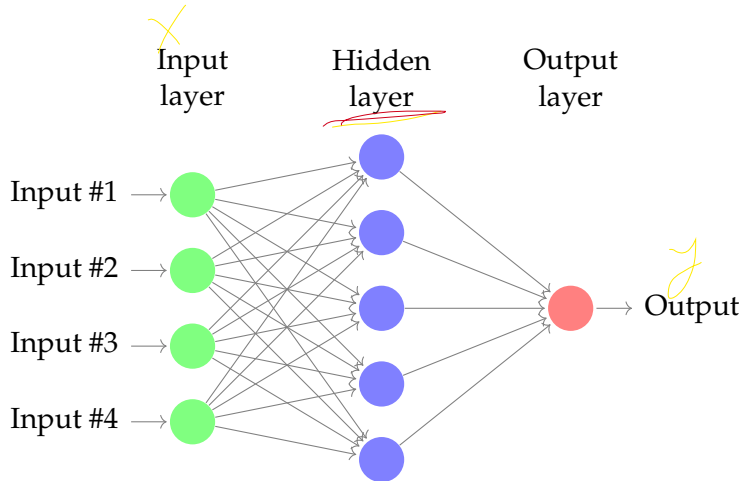
- ▶ Information x_i arriving from other neurons (or environmental sensors such as the retina) is received in the dendrites.
- ▶ In particular, that information is weighted by synaptic weights β_i determining the effect of the inputs (e.g., activation or inhibition via the product $x_i\beta_i$).
- ▶ The weighted inputs arriving from multiple sources are aggregated in the nucleus as a weighted sum $y = \sum_i x_i\beta_i$, and this information is then sent for further processing in the axon y
- ▶ From there it either reaches its destination (e.g., a muscle) or is fed into another neuron via its dendrites.

Deep Learning: Intro



- ▶ Linear Models May Go Wrong
 - ▶ For example, linearity implies the weaker assumption of monotonicity: that any increase in our feature must either always cause an increase in our model's output (if the corresponding weight is positive), or always cause a decrease in our model's output (if the corresponding weight is negative).
 - ▶ But the world is highly non linear (trees work really well)
- ▶ We can incorporate hidden layers
- ▶ The simplest deep networks are called multilayer perceptrons, and they consist of multiple layers of neurons each fully connected to those in the layer below (from which they receive input) and those above (which they, in turn, influence).

Deep Learning: Intro



Deep Learning: Demo

```
library(keras)  
fashion_mnist <- dataset_fashion_mnist()
```



Deep Learning: Demo

```
c(train_images, train_labels) %<-% fashion_mnist$train  
c(test_images, test_labels) %<-% fashion_mnist$test
```

Enter numbers
→ p, else

```
class_names = c('T-shirt/top', /  
                'Trouser', /  
                'Pullover', /  
                'Dress', /  
                'Coat', /  
                'Sandal', /  
                'Shirt', /  
                'Sneaker', /  
                'Bag', /  
                'Ankle boot')
```

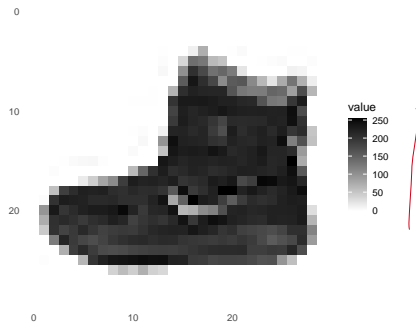
Deep Learning: Demo

```
library(tidyr)
library(ggplot2)

image_1 <- as.data.frame(train_images[1, , ])
colnames(image_1) <- seq_len(ncol(image_1))
image_1$y <- seq_len(nrow(image_1))
image_1 <- gather(image_1, "x", "value", -y)
image_1$x <- as.integer(image_1$x)

ggplot(image_1, aes(x = x, y = y, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black", na.value = NA) +
  scale_y_reverse() +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  theme(aspect.ratio = 1) +
  xlab("") +
  ylab("")
```

Deep Learning: Demo



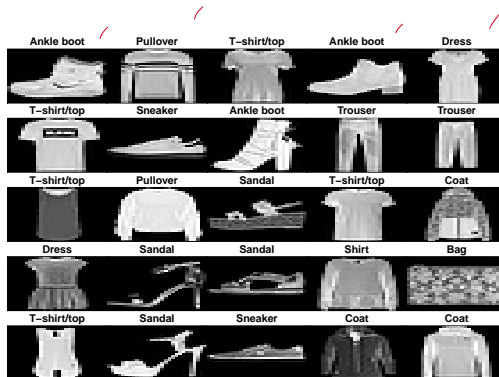
Deep Learning: Demo

```
train_images <- train_images / 255  
test_images <- test_images / 255
```

Normal, 20
0-1

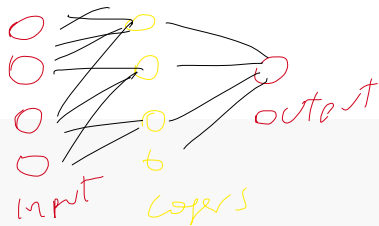
```
par(mfcol=c(5,5))  
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')  
for (i in 1:25) {  
  img <- train_images[i, , ]  
  img <- t(apply(img, 2, rev))  
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',  
        main = paste(class_names[train_labels[i] + 1]))  
}
```

Deep Learning: Demo



Deep Learning: Demo

```
model <- keras_model_sequential() =  
model %>%  
  layer_flatten(input_shape = c(28, 28)) %>%  
  layer_dense(units = 128, activation = 'relu') %>%  
  layer_dense(units = 10, activation = 'softmax')
```



```
model %>% compile(  
  optimizer = 'adam',  
  loss = 'sparse_categorical_crossentropy',  
  metrics = c('accuracy')  
)  
model %>% fit(train_images, train_labels, epochs = 5, verbose = 2)
```

Deep Learning: Demo

```
## Epoch 1/5
## 1875/1875 - 2s - loss: 0.5003 - accuracy: 0.8238
## Epoch 2/5
## 1875/1875 - 2s - loss: 0.3782 - accuracy: 0.8643
## Epoch 3/5
## 1875/1875 - 2s - loss: 0.3362 - accuracy: 0.8784
## Epoch 4/5
## 1875/1875 - 2s - loss: 0.3141 - accuracy: 0.8844
## Epoch 5/5
## 1875/1875 - 2s - loss: 0.2934 - accuracy: 0.8922
```

in training

```
score <- model %>% evaluate(test_images, test_labels, verbose = 0)

cat('Test loss:', score[1], "\n")
```

```
## Test loss: 0.3377942 ✓
```

```
## Test accuracy: 0.8792 ✓
```

Review & Next Steps

- ▶ Word Embedding
- ▶ Word Embedding: Demo
- ▶ Deep Learning: Intro
- ▶ Deep Learning: Demo
- ▶ Next class: More on Neural Nets
- ▶ Questions? Questions about software?

i



Further Readings

- ▶ Ash, E., Chen, D. L., & Ornaghi, A. (2020). Stereotypes in High-Stakes Decisions: Evidence from US Circuit Courts (No. 1256). University of Warwick, Department of Economics.
- ▶ Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola (2020) Dive into Deep Learning. Release 0.15.1. <http://d2l.ai/index.html> ✓
- ▶ Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advances in neural information processing systems (pp. 4349-4357).
- ▶ Clark, M (2018). An Introduction to Text Processing and Analysis with R. ✓
<https://m-clark.github.io/text-analysis-with-R/> Rstudio (2020). Tutorial TensorFlow https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_classification/
- ▶ Taddy, M. (2019). Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions. McGraw Hill Professional.
- ▶ Voigt, R., Camp, N. P., Prabhakaran, V., Hamilton, W. L., Hetey, R. C., Griffiths, C. M., ... & Eberhardt, J. L. (2017). Language from police body camera footage shows racial disparities in officer respect. Proceedings of the National Academy of Sciences, 114(25), 6521-6526.