COMP3420 Lesson 9

Greg Baker

2023-05-01

MACQUARIE University

Readings

- Jurafsky and Martin: Chapter 5
- Chollet: Chapter 11 until the start of 11.3.2

Recap/Lookahead

What we'll do today

- Create a classifier
- Talk about engineering it
- Explain it
- Try to improve the classifier
- Complain a lot about how hard it all is

The journey so far

- Two weeks ago: how text is stored (encoding)
- Two weeks ago: regexps for fast and simple processing
- Last week: what's a word?
- Last week: bag-of-words model
- Last week: TF-IDF (term frequency $\times$ inverse document frequency)

We can take some text and make some vectors from it.

Linking with the first half of the course

You had 2D arrays of numbers. You made classifiers.
Now you have 1D arrays of numbers. You will make classifiers.

Things that stay the same

- `Dense(16, activation='relu')`
- `Dense(1, activation='sigmoid')`
- `Dropout(0.2, input_shape=(16,))`

Things that stay the same

- `Dense(16, activation='relu')`
- `Dense(1, activation='sigmoid')`
- `Dropout(0.2, input_shape=(16,))`

Some small technical differences

- `keras.layers.Conv1D` → `Conv2D`;
  `keras.layers.MaxPooling1D` → `MaxPooling2D`,
  - Because text is a stream and images are 2-dimensional
- `keras.model.Sequential` → function trees (sometimes)
  - If you want to express layer bypasses, it's not Sequential any more

More small conceptual differences

- Text data is sometimes in files, but usually in databases.
  - Images are rarely in databases and usually shared in files
- Text sometimes uses really, really big models
- Context drift is a bigger problem
  - A model trained today might perform worse next year
- Explainability matters a lot more
  - With images, we don't often ask why

What's New

This week	`keras.layers.TextVectorization`
Next week	`keras.layers.Embedding`
Later	`keras.layers.Bidirectional`
Later	`keras.layers.RNN`
Later	`keras.layers.LSTM`
Later	Transformer architecture

(Some of these appear in video processing.)

Text Classification

What is Text Classification?

Classify documents into one of a fixed predetermined set of categories.

- The number of categories is predetermined.
- The actual categories are predetermined.

Examples

- Spam detection.
- Email filtering.
- Classification of text into genres.
- Classification of names by gender.
- Classification of questions.

Example: Spam Filtering

Distinguish this

```
Date: Mon, 24 Mar 2008
From: XXX YYY <xxx@yahoo.com>
Subj: Re: Fwd: MSc
To: Mark Dras
<madras@ics.mq.edu.au>

Hi, Thanks for that. It would
fit me very well to start
2009, its actually much better
for me and I'm planning to
finish the project in one year
(8 credit points).
```

from this

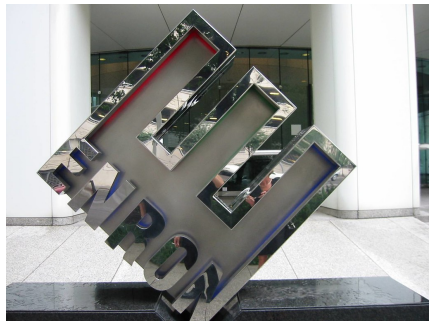
```
Date: Mon, 24 Mar 2008
From: XXX YYY
<xxx@yahoo.co.in>
Subj: HELLO
To: madras@ics.mq.edu.au

HELLO, MY NAME IS STEPHINE IN
SEARCH OF A MAN WHO
UNDERSTANDS THE MEANING OF
LOVE AS TRUST AND FAITH IN
EACH OTHER RATHER THAN ONE WHO
SEES LOVE AS THE ONLY WAY OF
FUN …
```

Enron

Enron and its Email Dataset: A Treasure for NLP Research

- **Enron:** A major energy, commodities, and services company, collapsed in 2001 due to accounting fraud and corporate corruption.
- **Email dataset:** As part of the federal investigation, more than 600,000 emails from 158 Enron employees were released, now publicly available.
- **Significance for NLP research:**
  - **Real-world data:** Emails contain a rich variety of topics, sentiments, and communication styles, providing a diverse corpus for training and evaluation.
  - **Spam detection:** Includes a mix of spam and non-spam emails, making it ideal for developing spam filters and classifiers.



ETL

Data engineering

Things we (in academia) often forget to tell you:

- **Data engineering:** The process of designing, building, and maintaining systems to collect, store, process, and analyze large datasets.
- **Lucrative career:** High demand for skilled data engineers in the job market, with competitive salaries and opportunities for growth.
- **Essential in NLP projects:** Data engineering is a crucial step to ensure high-quality, structured data for NLP models.

Key concepts to earn $$$:

Data pipelines	Write code to get data from one place to another.
Data storage	Understand various storage options, such as relational databases, NoSQL databases, and data warehouses
Data integration	Combine data from different sources and formats, ensuring consistency and accuracy.

prog1.py 8-23 (Data Extraction)

```
def read_file_from_zip(zf, file_path):
    f = zf.open(file_path, 'r')
    try:
        return io.TextIOWrapper(f,
            encoding='cp1252').read()
    except UnicodeDecodeError:
        pass
    f = zf.open(file_path, 'r')
    try:
        return io.TextIOWrapper(f,
            encoding='utf-8').read()
    except UnicodeDecodeError:
        pass
    f = zf.open(file_path, 'r')
    return io.TextIOWrapper(
        f,
        encoding='ascii',
        errors='backslashreplace').read()
```

prog1.py 25-38 (Labelling)

```
zip_file = 'enron1.zip'
data = []
with zipfile.ZipFile(zip_file, 'r') as zf:
    for file_info in zf.infolist():
        fname = file_info.filename
        if not fname.endswith('.txt'):
            continue
        if fname.endswith('Summary.txt'):
            continue
        content = read_file_from_zip(zf, fname)
        label = 'ham' if
            fname.startswith('enron1/ham/') else 'spam'
        data.append({'email_text': content,
            'spam_or_ham': label})

emails_df = pd.DataFrame(data)
```

prog1.py 38-44 (Data engineering)

```
emails_df = pd.DataFrame(data)
print(emails_df.sample(5))
db = sqlite3.connect('enron.sqlite')
emails_df.to_sql('enron',
                 db,
                 index=False,
                 if_exists='replace')
```

prog2.py (How to use it)

```
#!/usr/bin/env python
import sqlite3
import pandas
db = sqlite3.connect('enron.sqlite')
emails_df = pandas.read_sql('select * from enron', db)
print(emails_df.sample(5))
```

Classifiers

keras.layers.TextVectorization

```
#!/usr/bin/env python
import sqlite3
import pandas
db = sqlite3.connect('enron.sqlite')
emails_df = pandas.read_sql('select * from enron', db)
print(f"{emails_df.shape=}")

from tensorflow import keras
vectorizer = keras.layers.TextVectorization()
vectorizer.adapt(emails_df.email_text)
text_vectors = vectorizer(emails_df.email_text)
print(f"{text_vectors.shape=}")
print(text_vectors[:3])
print(vectorizer.get_vocabulary()[:5])
```

keras.layers.TextVectorization design flaws

- You have to give the vocabulary either as a constructor, or with `.adapt()`
- It's a layer, but you can't put it into a model without `.adapt()`'ing it to the data that will be passed to the model.
  - Which you can't do if the model has randomly-selected validation data
    - Well you can, but then you are leaking training data into the validation data.
- So you have to:
  - Split train, validation and test data
  - Adapt on train, vectorize train
  - Vectorize validation
  - Vectorize test
- Tough luck if you wanted to do cross-validation

Things we haven't told you about layers

- A layer can be part of a model ✓
- A layer has initialisation arguments ✓
- **A layer is also a function**:
  - Can be used outside of a `Model`
  - Can operate on a list / `numpy` array
  - Can operate on a `tf.Dataset`
  - Can operate on a `tf.Tensor` (either eager or lazy)

prog3.py 12–26 (Core of today's class)

```
12  emails_df['spaminess'] = np.where(
13      emails_df.spam_or_ham == 'spam', 1.0, 0.0)
14
15  tv, test_data = train_test_split(emails_df)
16  train_data, validation_data = train_test_split(tv)
17
18  vectorizer = keras.layers.TextVectorization(
19      output_mode='tf_idf', ngrams=2)
20  vectorizer.adapt(train_data.email_text)
21  train_vectors = vectorizer(train_data.email_text)
22  validation_vecs = vectorizer(validation_data.email_text)
23  test_vectors = vectorizer(test_data.email_text)
24
25  model = keras.models.Sequential(
26      [keras.layers.Dense(1, activation="sigmoid")])
```

prog3.py 27–41 (You already knew this)

```
model.compile(
    loss='binary_crossentropy',
    metrics=["accuracy"])

history = model.fit(
    x=train_vectors,
    y=train_data.spaminess,
    validation_data=(validation_vecs,
                     validation_data.spaminess),
    callbacks=[keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=3
    )],
    epochs=20)
```

Now what?

- Function notation
- Explainability
- Lexico-statistical guidelines
- Just try a few different things (you might want to hold out more test data)

Function Notation

Instead of:

```
model = keras.models.Sequential(
    [keras.layers.Dense(20, activation="relu"),
     keras.layers.Dense(1, activation="sigmoid")])
```

Functional (layers as functions on lazy tensors):

```
from keras.layers import Dense
inputs = keras.Input()
layer1 = Dense(20, activation="relu")(inputs)
output = Dense(1, activation="sigmoid")(layer1)
model = keras.Model(inputs=[inputs], outputs=[output])
```

- Can express complex graphs
- What the textbook uses

Explainability

Two reasons that data science projects happen

Inference We want to make automatic predictions, as accurately as possible.

Explainability We want to understand *why*. What features are discriminative and what effect do they have?

The very first question on any project: which goal are you aiming for?

Sometimes you don't have a choice

- **GDPR** General Data Protection Regulation, a comprehensive data privacy regulation in the European Union. AI decisions with legal or significant impact must be explainable, ensuring transparency and accountability.
  - **Extra-territorality** Applies to organizations processing EU citizens' data, *even if your company is in Australia*

- **Similar laws** Legal frameworks in other countries also emphasize AI explainability with extra-territorality:
  - **China** Draft Data Security Law and Personal Information Protection Law include provisions to protect citizens from unfair or opaque AI decisions.

Examples

- Software that scans resumes that filters out unqualified candidates.
- A Turn-it-in competitor that fails students for cheating
- Car insurance claim assessment that analyses the client's version of the accident

All of these have a *legal effect* (cause a contract to go ahead / be rejected): therefore they all have an explainability requirement.

The dearth of good sources on this

- I can't find any good readings on GDPR and PIPL.
- Lots of random web pages, some of which are somewhat trustworthy.
- Some very dull journal articles

What you need to know: *GDPR and PIPL require explainability if the model is used for something important that might affect someone's life.*

Sometimes you do have a choice

There are advantages to explainable models.

- Easier to debug
- Can be a unique selling point or product differentiator
- Being ethical is a good idea

Cynthia Rudin: Explainable is better than extra accuracy

- **Cynthia Rudin:** Professor of Computer Science, Electrical and Computer Engineering, and Statistical Science at Duke University.
- **Research focus:** Interpretable machine learning, AI fairness, and robustness.
- **Explainable AI:** Developing models that provide clear explanations for their decisions, fostering trust and transparency.
- **Accountability:** Enables humans to understand and verify model decisions, ensuring responsible AI deployment.
- **Regulatory compliance:** Helps organizations meet legal requirements for explainability and transparency.
- **Improved decision-making:** Facilitates better collaboration between humans and AI, enhancing overall performance.

Is Professor Rudin correct?

One of the most important questions in AI today.

prog3f.py — an explainable model (functional version of prog3.py)

```
vectorizer = TextVectorization(output_mode='tf_idf',
    ngrams=2)
vectorizer.adapt(train_data.email_text)
train_vectors = vectorizer(train_data.email_text)
validation_vectors =
    vectorizer(validation_data.email_text)
test_vectors = vectorizer(test_data.email_text)

vocab_size = vectorizer.vocabulary_size()
inputs = keras.Input(shape=(vocab_size,))
output = Dense(1, activation="sigmoid")(inputs)
model = keras.Model(
    inputs=[inputs],
    outputs=[output])
```

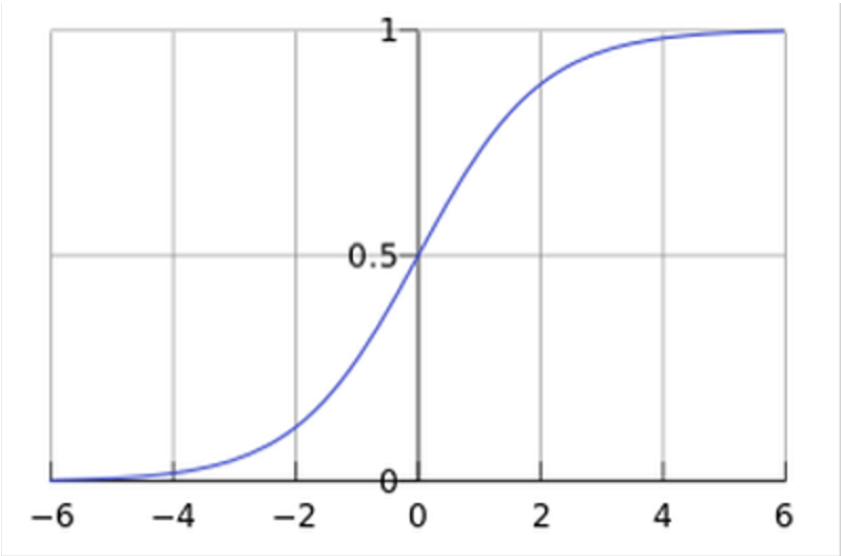
Our first text processing architecture

Logistic Regression Review

$$y = \frac{1}{1+e^{a-W\cdot Z}}$$

Given some pairs of (y_n, Z_n) find "good" values of a and W. (W and Z will be vectors).

- a shifts it left-right
- W controls the steepness in different variables.



Logistic Regression + TFIDF

- Very common first pass solution
- Works quickly, works well
- **Explainable**
- Easy to optimise
- The coefficients of the logistic regression (the weights of the model) tell you what features were highly discriminative.

wordeffects.py 48–56 (Explaining how our model works)

```
vocab = vectorizer.get_vocabulary()
weights = model.get_weights()[0][:,0]

# Put them into a Pandas series so we can manipulate
# them easily.
strengths = pd.Series(index=vocab, data=weights)

print(strengths.nlargest(5))
```

Output

```
subject get     0.193929
be removed      0.180512
removed         0.180237
click here      0.177232
here to         0.174491
dtype: float32
attached    -0.228672
hpl nom     -0.221602
daren       -0.196141
2000 see    -0.192181
meter       -0.190120
dtype: float32

systemMemory: 16.00 GB
```

Most spam-like words and phrases

Most ham-like words and phrases

Concepts to understand

A big positive number This word or phrase is strongly associated with being in this class.

A big negative number This word or phrase strongly associated with *not* being in that class.

Numbers around zero This word or phrase isn't very useful to classifier — it wasn't associated with being in class or not.

If all the coefficients are close to zero, then the classifier will always predict the same class. (Why?) It means there is **data irrelevancy**.

Evaluation

Why is evaluation part of explainability?

- How good is your explainable AI system?
- Why should I believe its explanations?

Of course, in addition you have to have a debugged system and evaluation helps.

Intrinsic vs. Extrinsic Evaluations

Intrinsic Evaluations

- An intrinsic evaluation focuses on the task independently of other tasks.
- We have some ground truth or way of saying that it is right or wrong.
- **Example:** When we run this, do we get the right answer? ("Yes, 95% recall.")
- Often used with a proof-of-concept before you have customers.

Extrinsic Evaluations

- Evaluate the impact of a task on the final goal of the complete system.
- **Example:** Does the user mark messages that we thought were Ham as Spam? Or the other way around? ("Only 0.2% of messages were over-ridden by the user.")
- Often used with A/B testing to determine whether a new component goes live into production.

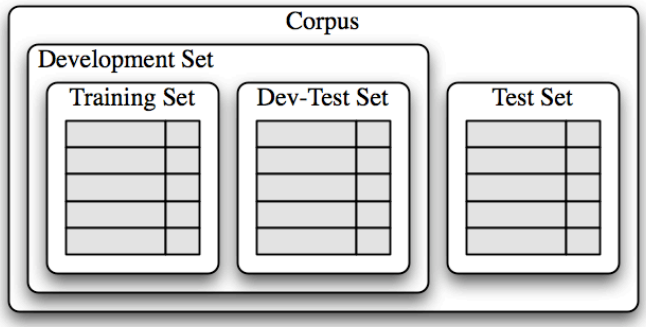
Methodology review

Development and Test Sets

- Put aside a test set and don't even look at its contents. Use the remaining data as a development set.
- Separate the development set into training and dev-test/validation sets.
- Use the training set to train the statistical classifiers.
- Use the dev-test/validation set to fine-tune the classifiers and conduct error analysis.
- Use the test set for the final system evaluation once all decisions and fine-tuning have been completed.

Important

Test your system with data that has not been used for development (Why ...?)



Positives and Negatives

- Whenever a system needs to make a binary choice, it is classifying the text into two classes: a positive and a negative class.

  Positive: What we want to select.

  Negative: What we do not want to select.

- The choice of what is a positive or a negative class depends on the application.
  - In information retrieval, documents relevant to the query belong to the positive class.
  - In spam filtering, spam documents belong to the positive class.
- We can see that the concept "positive" may not agree with our intuitions!

True, False, Positives, Negatives

In a contingency table, we can group results of the system into four categories:

True positive (tp) : The system correctly detects a positive.

True negative (tn) : The system correctly detects a negative.

False positive (fp) : The system wrongly classifies a negative as a positive.

False negative (fn) : The system wrongly classifies a positive as a negative.

Contingency table

system decision	actual case: positive	actual case: negative
positive	tp	fp
negative	fn	tn

Example: Spam Filtering

In spam filtering, "spam" emails belong to the positive class.

	actual case	
system	spam	not spam
marked spam	tp	fp
not marked spam	fn	tn

- If our spam filter classifies a legitimate email as spam, this is a false positive.

Question

False positives in spam filtering are usually more dangerous than false negatives; why?

Precision and Recall

Formulas

- $\text{precision} = \frac{\text{tp}}{\text{marked as positive by the system}} = \frac{\text{tp}}{\text{tp+fp}}$
- $\text{recall} = \frac{\text{tp}}{\text{should be marked as positive}} = \frac{\text{tp}}{\text{tp+fn}}$

Example

From a total collection of 200 documents, a retrieval system returned 30 documents, but 5 were not relevant. It also missed 12 documents.

Example

Contingency table

	actual case	
system	relevant	not relevant
retrieved	25	5
not retrieved	12	158

Values of measures

- precision $= \frac{25}{25+5} = \frac{25}{30}$
- recall $= \frac{25}{25+12} = \frac{25}{37}$

Further precision and recall notes

- You can usually trade between them.
  - If you need more precision, and can lose recall, then bias more towards saying "no".
  - If you need better recall, and can lose precision, then bias more towards saying "yes".
- **Extrinsic evaluation** In real-world systems there's usually a cost ($$$) to a false positive and a cost to a false negative. Aim to minimise that cost.

F-Measure

- The F-measure combines precision and recall into a single measure.

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \times \text{recall}}{\beta^2\text{precision} + \text{recall}}$$

- The most common combination is when $\beta = 1$, referred to as F_1 :

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- This is the harmonic mean of precision and recall.
- For our previous example, $F_1 = 0.746$
- Don't need to memorise these formulas: just understand that it captures precision and recall and why it is useful for imbalanced data sets.

Accuracy

- Accuracy is the number correctly classified out of the whole set.
  - accuracy $= \frac{\text{correct decisions}}{\text{all data}} = \frac{\text{tp+tn}}{\text{tp+fp+tn+fn}}$
  - For previous example, accuracy is 183/200
- Sometimes used (wrongly) to refer to precision.

Beware of unbalanced data

What happens if you have unbalanced classes, e.g. there are 100 documents, 90 of them belong to the negative class, and the system classifies everything as a negative?

- Recall: $\frac{0}{10} = 0$
- Precision: $\frac{0}{0} =$ NAN
- Accuracy: $\frac{90}{100} = 0.9$

Log loss

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

- N = Number of samples
- y_i = True label for sample i
- $\hat{y}_i$ = Predicted probability of sample i belonging to class 1

This is what logistic regression minimises.

No need to memorise

Cheat sheet for classifier metrics — worth memorising!

Metric	**Useful for**	**Why**
Accuracy	Balanced data	Easy to understand
Precision	Fraud, theft, crime	Don't accuse a good customer incorrectly
Recall	Medicine	Never ignore a potential illness
F1	Imbalanced data	Balance of precision and recall
Log loss	Machine learning	Allows incremental improvements

Balanced data = each target has a similar number of training examples.

metrics.py 38–56 Metrics in Keras

```
model.compile(loss='binary_crossentropy',
    metrics=["accuracy",
             keras.metrics.Precision(),
             keras.metrics.Recall()])
# keras.metrics.F1Score() is in development builds only
    (2023-04)
history = model.fit(
    x=train_vectors,
    y=train_data.spaminess,
    validation_data=(validation_vectors,
                     validation_data.spaminess),
    callbacks=callbacks,
    verbose=0,
    epochs=20)
print("Training set evaluation:")
print(model.evaluate(test_vectors,
                     test_data.spaminess,
                     return_dict=True,
                     verbose=2))
```

Improving your classifier

Improving a classifier is very hard

Note that most of the "improvement" examples in the Chollet book don't do much better than the initial baseline!

Reminder from last week

How "useful" a typical word is likely be to a classifier:

$$\frac{C}{V} = \frac{N}{kN^{\beta}L} = \frac{N^{1-\beta}}{kL}$$

where

- C is the number of *documents* in the corpus.
- V is the total number of "words" in the corpus (which might mean bigrams, or BPE fragments)
- $L = \frac{N}{C}$ is the average length of a document in the corpus
- β close to 1 means the classifier doesn't get smarter as we add more documents
- β close to 0 (lots of repetition) means a classifier will get better rapidly

Root causes of poor models: Overfitting

The model is *memorizing* the data, and not *generalizing* to create good rules.
Unless your model is very simple (e.g. just one sigmoid layer), it *will* overfit if trained long enough.
Symptoms: validation loss is increasing while training loss is decreasing. Often $\frac{C}{V} < 1$.

What to do about overfitting?

- Shrink vocabulary (cut off uncommon words, cut off very common words)
- Get more data (use Herdan's Law to figure out how much)
- Use word cores (byte-pair encoding, lemmatization)
- Add dropout or regularization

State-of-the-art

Some truths that won't be in an exam

There is an effect called "double descent" and "grokking" where you add a lot more parameters and more training to an overfitting model, and start seeing better accuracy.
This is very weird, and is an active area of research.

Root causes of poor models: Underfitting

Your model is losing valuable information somewhere.

- How is the word "not" being handled?
- Are you expecting it to use word order but you supply the data as a bag-of-words model?

Symptoms: Train accuracy/recall/F1 and validation accuracy/recall/F1 are bad, even when $\frac{C}{V} > 25$

What to do about underfitting?

- Add more parameters (more layers, or more neurons)
- Use bigrams/trigrams
- Try a different architecture
- Include word position or word relationship information, such as with an embedding (next week's lesson)

Data irrelevancy

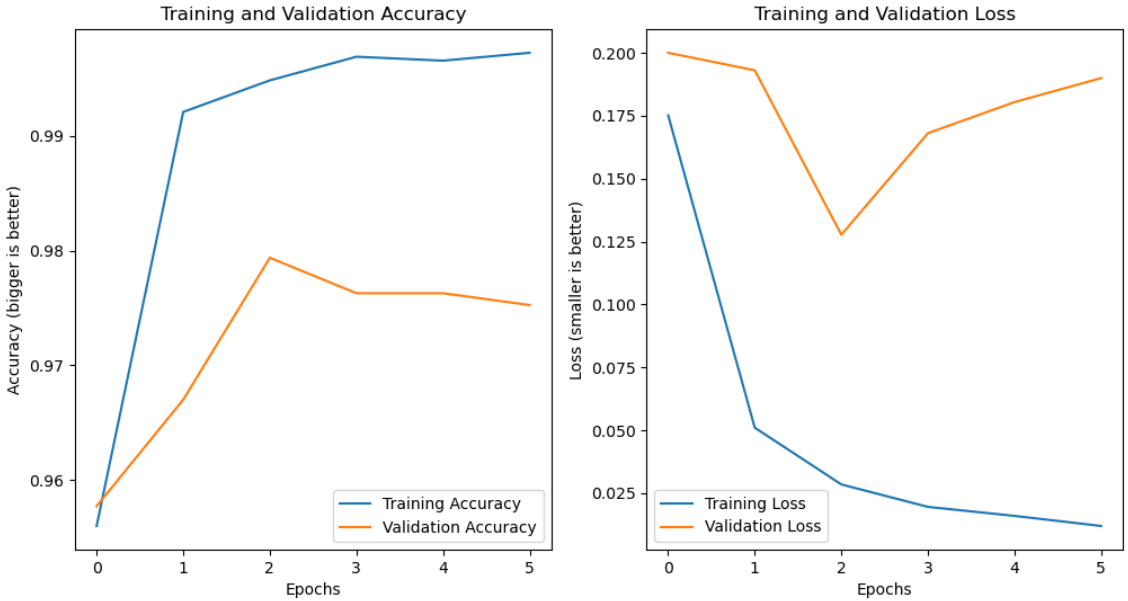
There is no relationship between the input data and the output labels.
e.g. classifying whether an essay was human-written or written by generative AI usually doesn't work because there aren't any differences in the text.
Symptoms: Looks like underfitting, but doesn't improve when you address it.

enron_training_graph 42–59

```
history = model.fit(
    x=train_vectors,
    y=train_data.spaminess,
    validation_data=(validation_vectors,
                     validation_data.spaminess),
    callbacks=callbacks,
    verbose=0,
    epochs=20)

fig, axes = plt.subplots(ncols=2, figsize=(12,6))
axes[0].plot(history.history['accuracy'],
         label='Training Accuracy')
axes[0].plot(history.history['val_accuracy'],
         label='Validation Accuracy')
axes[0].set_title('Training and Validation Accuracy')
axes[0].set_xlabel('Epochs')
axes[0].set_ylabel('Accuracy (bigger is better)')
axes[0].legend()
```

Outputs




```
Corpus size =  5172
Vocabulary size = 216906
Corpus : vocabulary ratio = 0.023844430306215594
Test evaluation:
{'loss': 0.1550474762916565, 'accuracy': 0.9798917174339294}
```

Address the overfitting

What we'll do:

- Shrink vocabulary: no more bigrams, and remove all words that only appear once. ✓

What we could do (but won't in this lecture):

- Add dropout
- Increase data size
- Lemmatize

keras.layers.TextVectorization – more design flaws

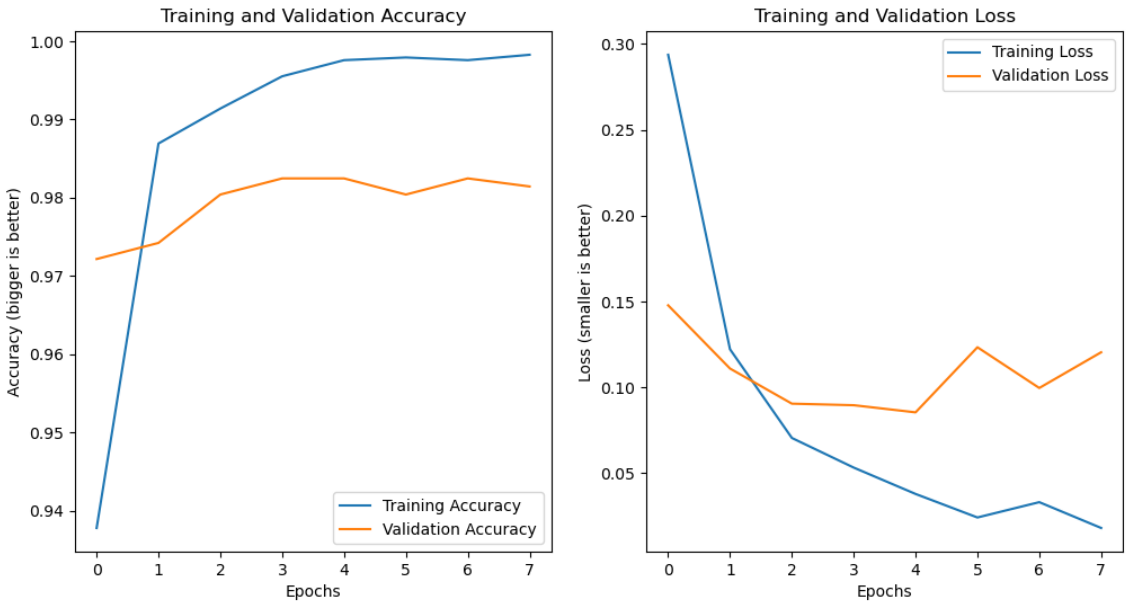
- You can't specify "words that appear more than N times"
- You can only specify a maximum vocabulary size

enron-better.py 25–36

```
counter = collections.Counter()
for text in train_data.email_text:
    for word in nltk.word_tokenize(text.lower()):
        counter[word] += 1
ideal_vocab_size = 1  ;# OOV token counts as 1
for c in counter.values():
    if c > 1:
        ideal_vocab_size += 1

vectorizer = TextVectorization(
    output_mode='tf_idf',
    max_tokens=ideal_vocab_size)
```

Outputs




```
Corpus size =  5172
Vocabulary size = 16446
Corpus : vocabulary ratio = 0.3144837650492521
Test evaluation:
{'loss': 0.10100845247507095, 'accuracy': 0.9868522882461548}
```

Summary

- Keras models for text are very similar to models for images
- Explainable AI is a sub-field where transparency is valued more than accuracy.
  - GDPR and China's PIPL *require* explainable AI.
  - Projects where understanding why something is happening is more important than classifying new data
- Logistic regression combined with TFIDF lets you build explainable classiers very quickly that work well for simple tasks.
- Logistic regression is inherently explainable (you just need the weights, and the word vectors)
- Metrics: accuracy, recall, precision, F1 and log loss.
- How to respond to overfitting and underfitting

What's Next

Week 10

- Embeddings

Reading

- Chollet: Section 11.3.3
- Jurafsky and Martin, Chapter 9 (optional)