Sentiment Analysis

Sentiment Analysis

Dr Daniel Chalk

Sentiment Analysis

Important :

You must download the Large Movie Review Dataset v1.0 available here :

https://ai.stanford.edu/%7Eamaas/data/sentiment/

and extract to the same directory as your .py files for this session

Sentiment Analysis - Recap

Sentiment Analysis uses AI-based methods to try to automatically identify the ‘sentiment’ / ‘tone’ of a piece of text, to identify if it is positive, negative or neutral.

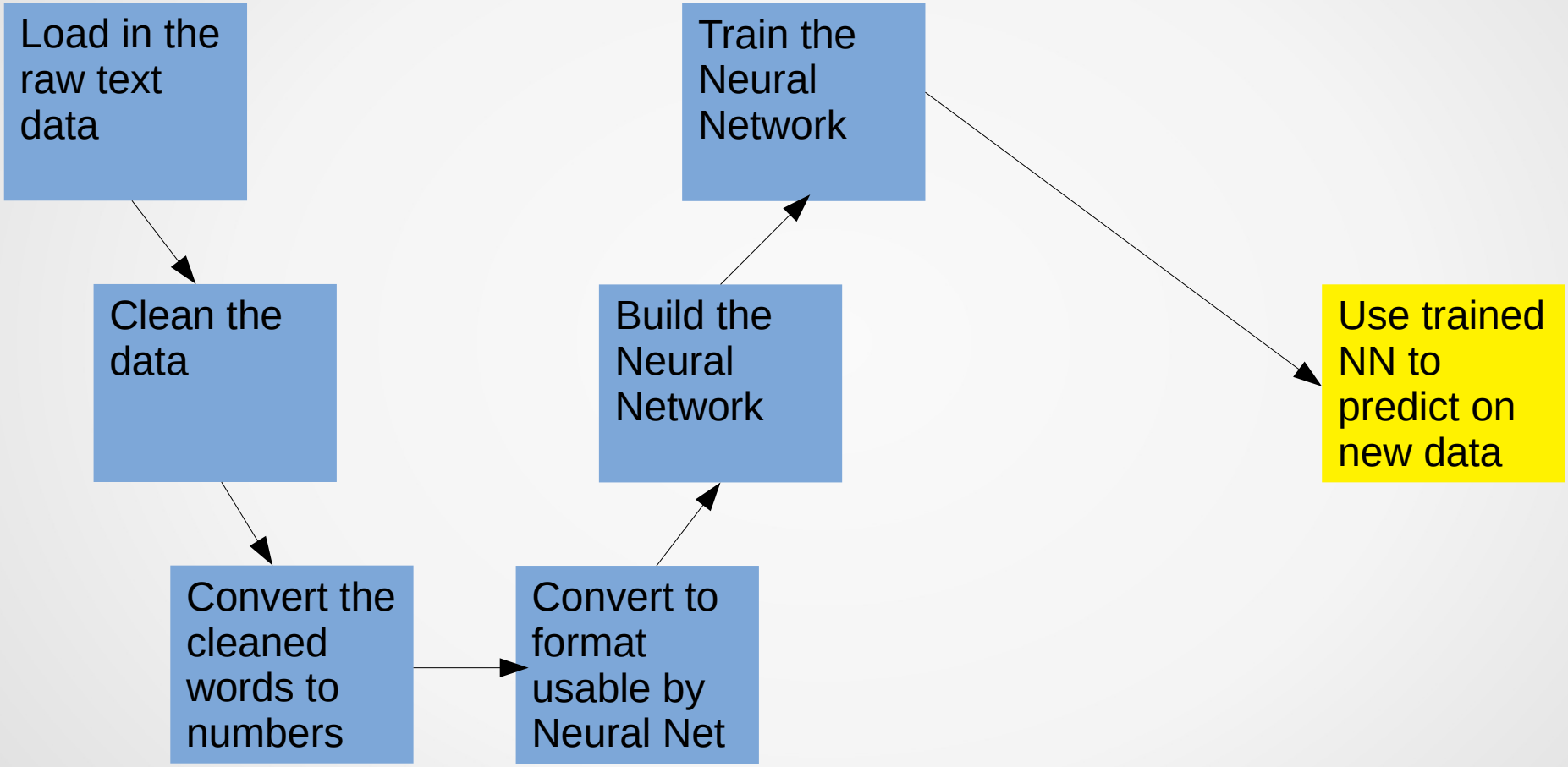
In a similar way to the way in which SpaCy uses AI to predict whether a word (or group of words) is a Named Entity, here we are predicting whether a piece of text is positive (1), or negative (0). In some cases, we may also assess sentiment as being positive (1), negative (-1) or neutral (0).

Applications of Sentiment Analysis

Potential Applications :

- automatically classifying service user survey data
- analysing reviews of movies, books etc
- analysing social media posts, and flagging up negative comments to be addressed
- looking for positive or negative references to your organisation on websites etc

Sentiment Analysis – The Overall Process



Clean the Data

When we undertake sentiment analysis, we need to carry out a number of steps first to process and clean up the data.

We don’t tend to do this with Named Entity Recognition, because things like the presence or absence of capital letters, and words like “and” and “the” might be important.

But when we’re trying to determine sentiment, we’re wanting to look at key words, and less about the syntactic structure of the text.

Therefore, there a number of things we normally want to do with our text before analysing the sentiment.

Clean the Data

1. Convert everything to lowercase. We don't care if someone wrote "Rubbish" or "rubbish" (or even "ruBbIsh") - for sentiment analysis, we want to consider them all as the same word.

2. Tokenize the text – break down text into list of individual words.

3. Remove punctuation (and, optionally, numbers) – these things are rarely useful for determining sentiment, although numbers might be, depending on the context.

4. Stem the words – find words with common stems and treat them all as the same word (e.g "watching", "watch", "watched" all have the same stem, and we want to treat them all the same way semantically for the purpose of the determining sentiment)

5. Remove stopwords (such as "a", "the", "and" etc. They won't help us)

Convert cleaned words to numbers

Neural Networks (and computers in general) need to deal, at the basic level, in numbers, not text.

So we need to convert our words into numbers that can then be fed through our Neural Network and be numerically manipulated etc until combinations that should result in Positive reviews have an output of 1, and negative reviews have an output of 0, for example.

To do this, we represent words according to the inverse of their frequency in the text, so that the most frequent words have a number of 1. This allows us to easily chop out words that occur infrequently, from which learning will be difficult, and which are likely less important for learning anyway.

Convert to format for Neural Network

We need to convert the Neural Network inputs (now numbers) into a format acceptable to the Neural Network.

This will involve :

- Separating the data into “inputs” (the sequences of words (as numbers) making up each bit of text we want to classify) and “labels” (a 1 or 0 to indicate whether each input is positive or negative, for training).
- We need all inputs to be the same size, so we decide on a size (number of words), and truncate all longer inputs, whilst padding out all shorter inputs with a dummy word value of 0.
- Split the data we have into training, validation and testing sets, so the NN can learn, adjust how well it’s learning, and then test out how well it’s learned.

Build and Train the Neural Network

Essentially we :

- Build the Neural Network up by specifying one layer at a time
- Compile the Neural Network to assemble it

then :

- Specify how many “epochs” (learning time units) we want the NN to go through, and the “batch size”, which specifies how many inputs will be shunted through the network at once
- Start the model training itself
- Output metrics so we can monitor how well its learning (whilst looking out for potential issues, such as overfitting)

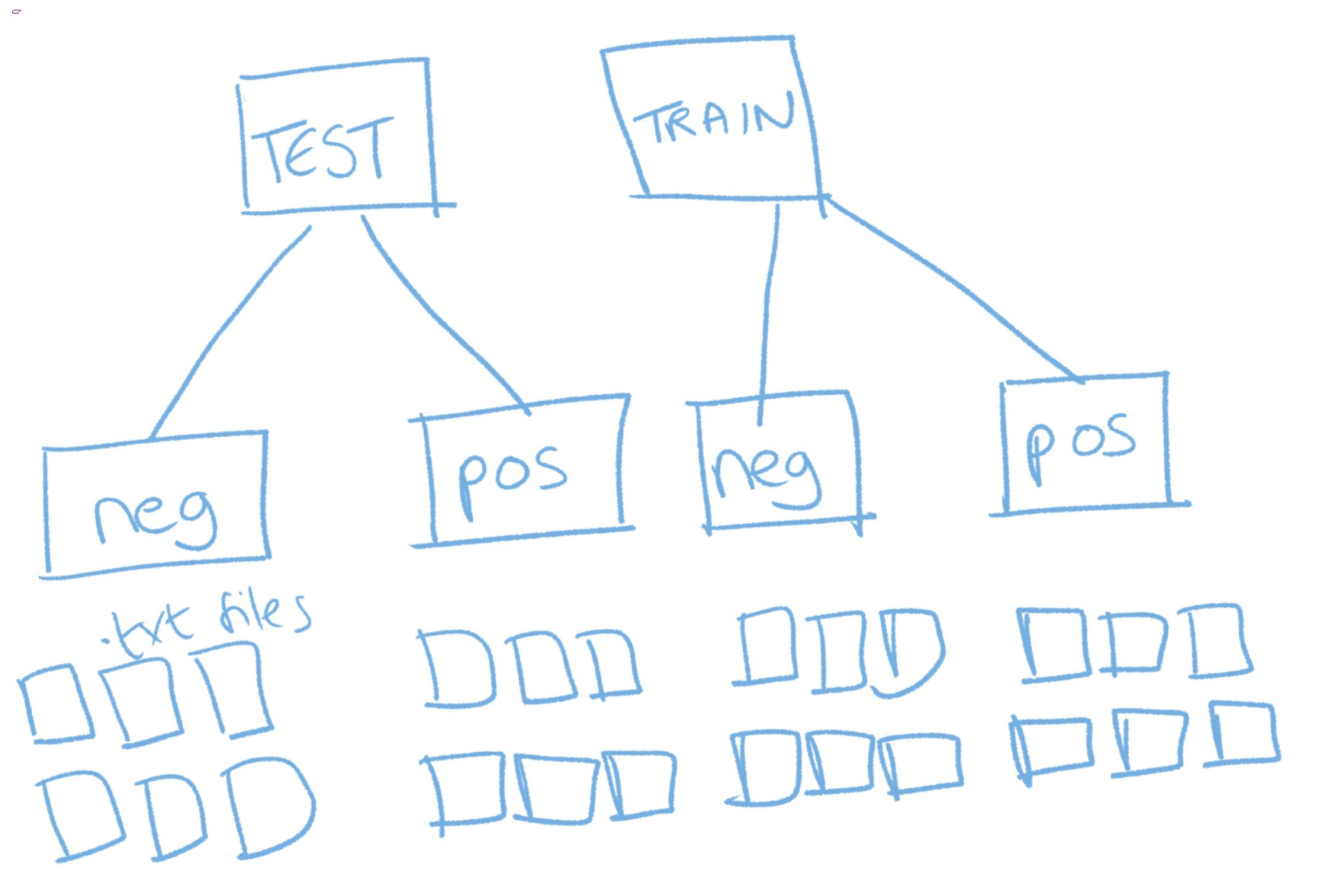
Analysing the Sentiment of Movie Reviews

Let’s look at an example of how we might use Sentiment Analysis to automatically classify movie reviews from the Internet Movie Database (IMDB) as “Positive” or “Negative”.
(sa_with_own_data.py)

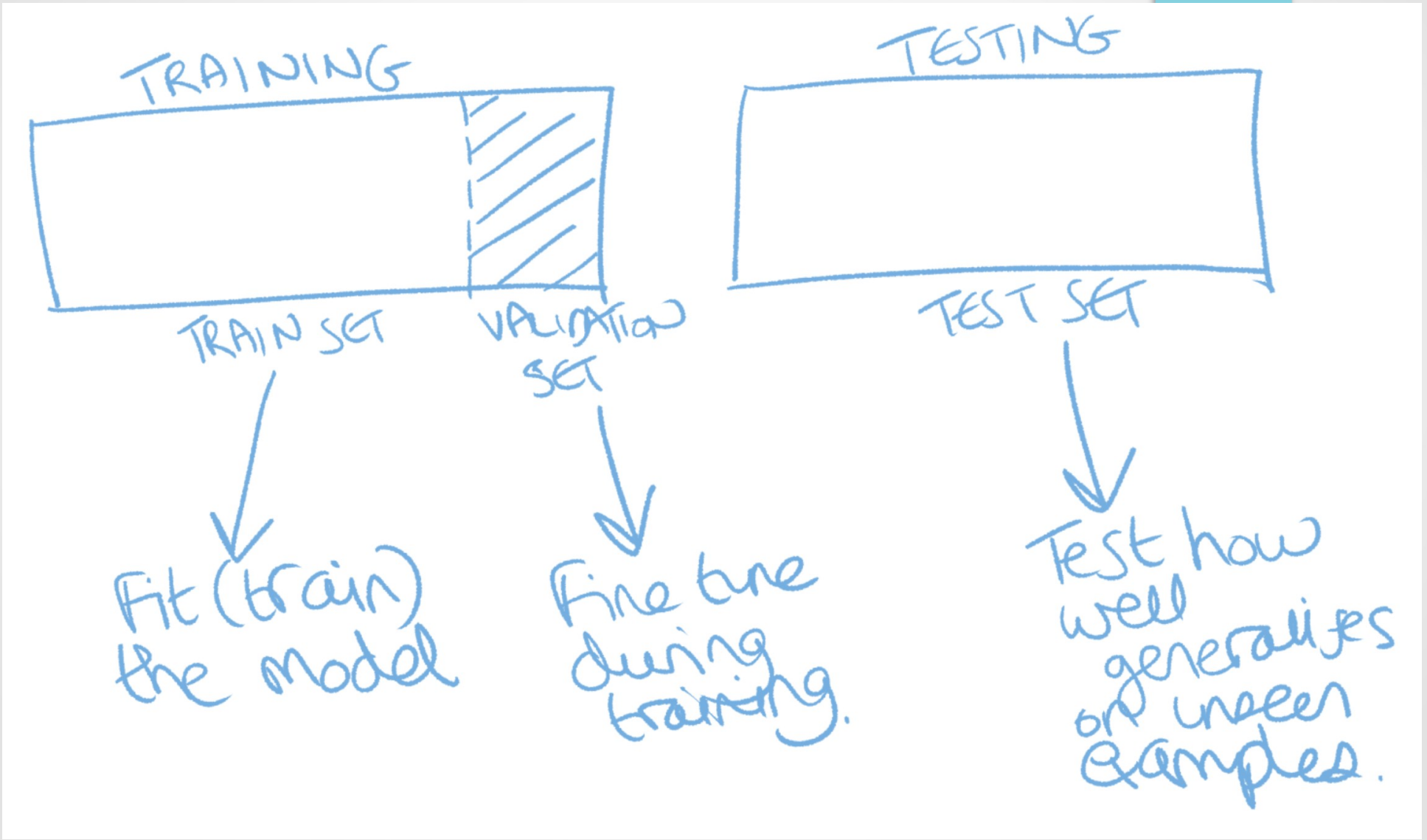
We’ll use a platform developed by Google called *TensorFlow*, and the API to access its features known as *Keras*.

But first, let’s talk through what this code is going to do.

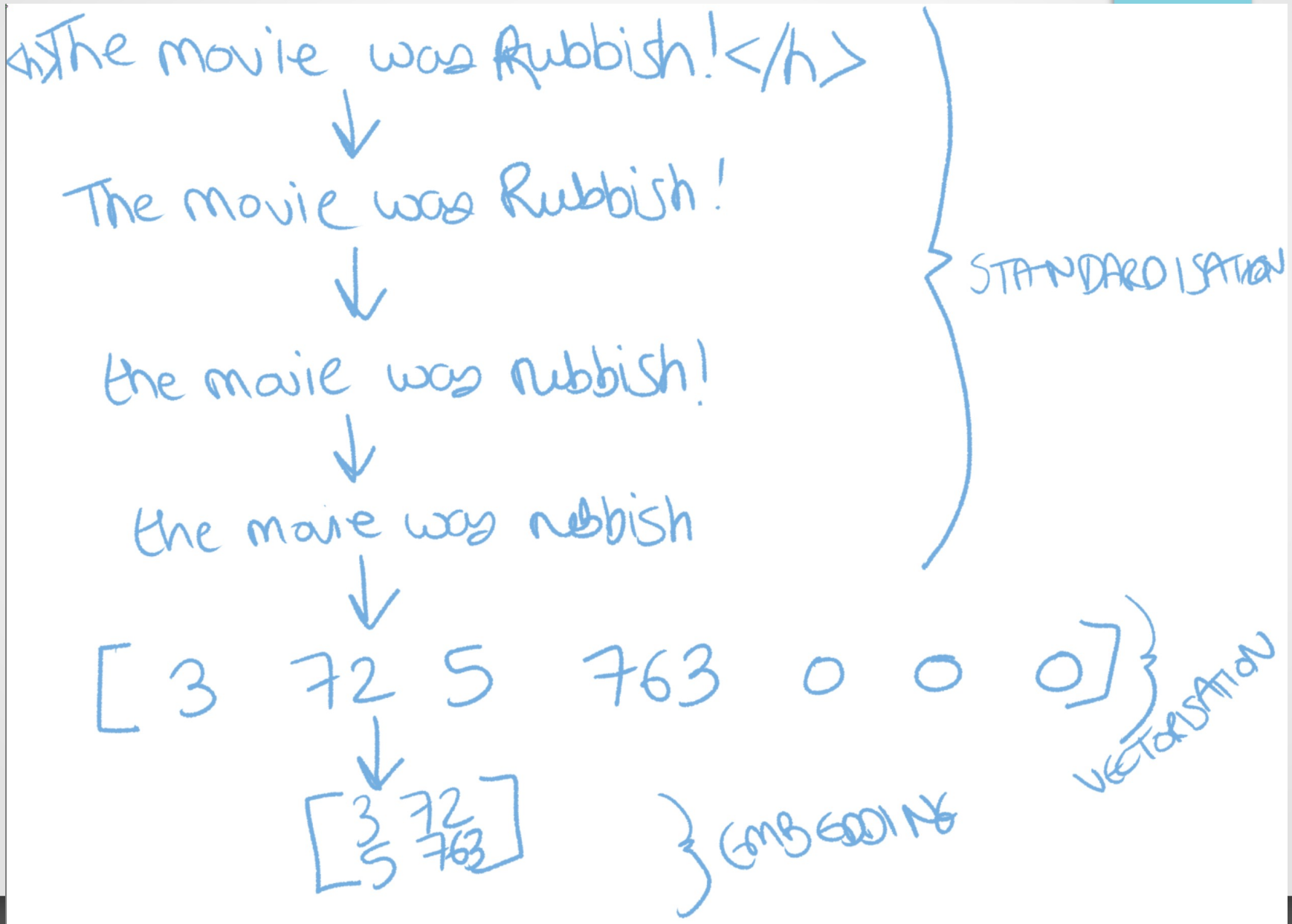
Directory structure for our examples



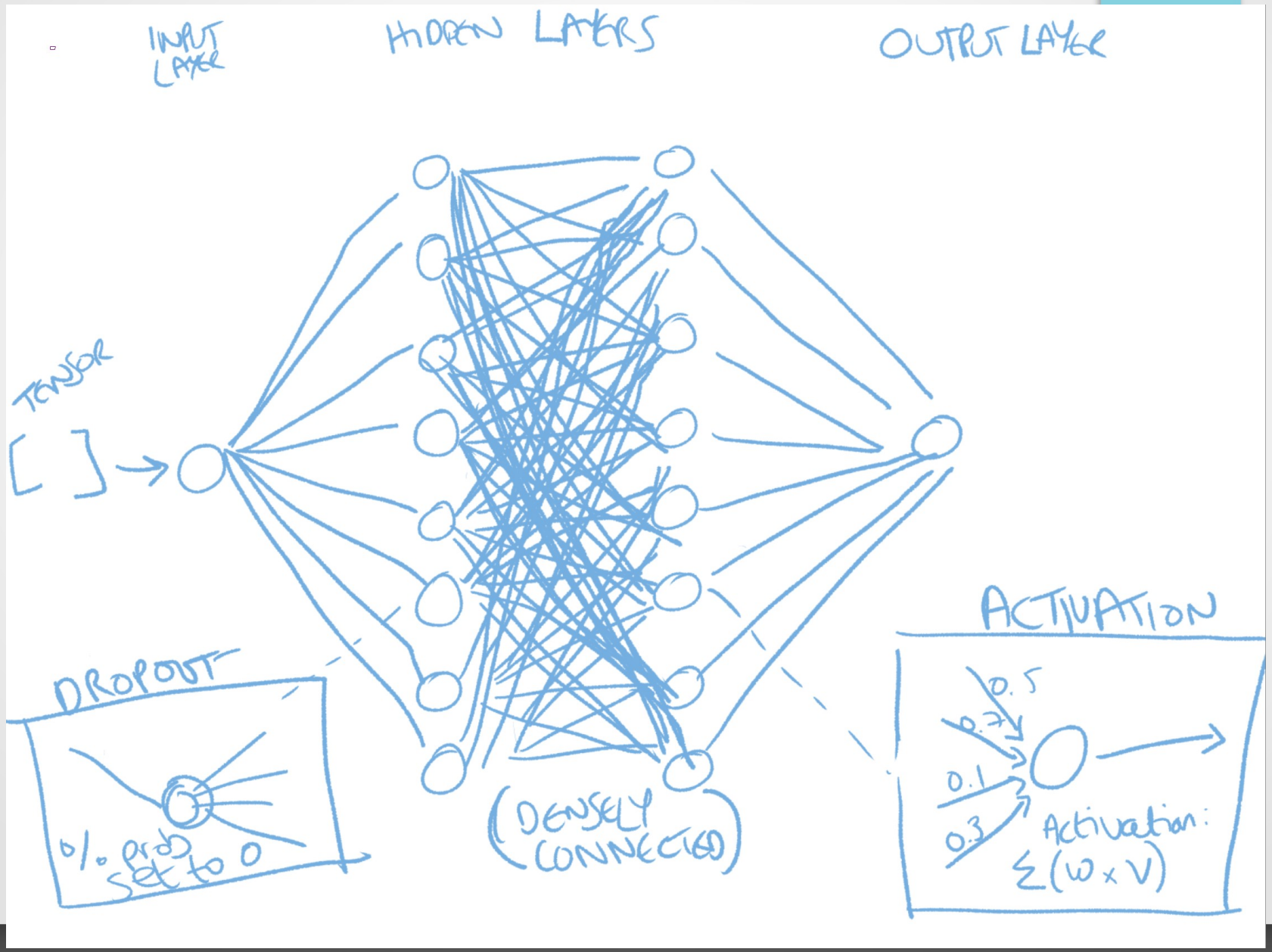
Training, Validation and Test Sets



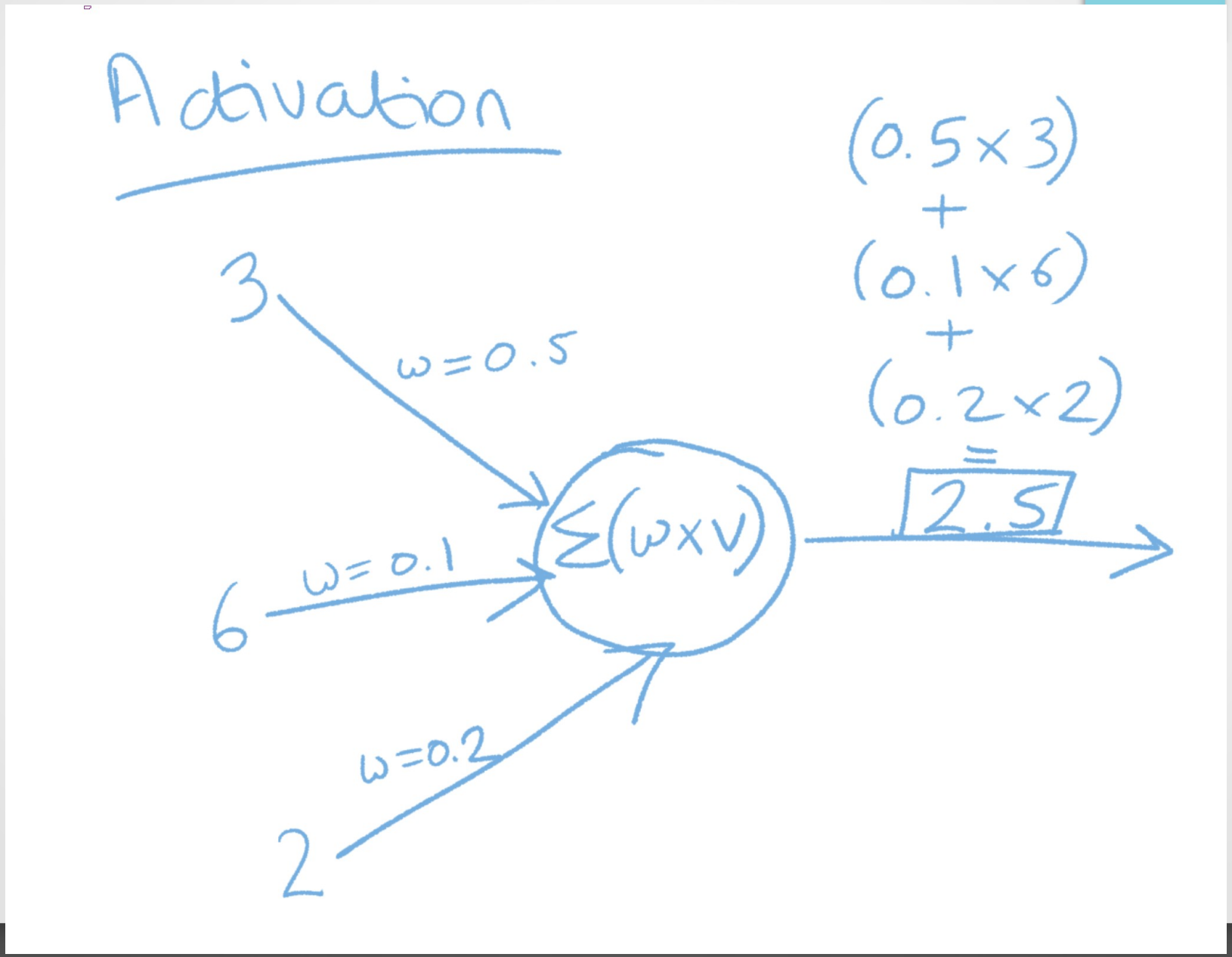
Standardisation, Vectorisation and Embedding



Neural Network



Activation



text_dataset_from_directory

Let’s now look at how we might create a more detailed model that uses our own data. For the purposes of the demonstration, we’ll again use some IMDB data, but this time it we will use the raw review data in the aclImdb directory, and we’ll use the text_dataset_from_directory utility, which loads in raw positive and negative examples as long as they’re stored in a certain way.

Let’s look at the sa_with_own_data.py file.

Overfitting

When we trained our model before we saw that, whilst training loss decreased and training accuracy increased in each new epoch, validation loss and accuracy seemed to peak early and we soon ran into “diminishing returns”.

This indicates overfitting – we’ve just taught the model to be excellent at predicting the features unique to the training examples, but we’re not improving predictive performance for unseen examples, thereby severely limiting our ability to *generalise* (which is the whole point of training the model).

To combat this, we can stop training before we reach that point. We could either experiment with the number of epochs to run, or we could use a handy tool called EarlyStopping to automatically stop training when we reach that point.

EarlyStopping

EarlyStopping is a *callback* and is imported from tensorflow.keras.callbacks

Callbacks operate meta to the training, and monitor the training process so they can take action if certain conditions are met.

EarlyStopping allows the training to stop early if a metric that is being *monitored* doesn't improve by a *threshold* with a certain level of *patience* (number of epochs).

Let's have a look at how we can add EarlyStopping to our code (sa_with_own_data_early_stopping.py; line 272 onwards)

Exercise 1

Create a .csv file called “ex_1_reviews.csv”. In this file, write a small sample of short movie reviews, with a new review on each line. Ensure this file is in the same folder as wherever your program is saved.

Then, take a copy of sa_with_own_data_early_stopping.py and add functionality that :
- reads in the reviews in the csv file and stores them in a list
- iterates through the list of new reviews, and for each review :
    - applies the exported model you train in earlier in the code to the review
    - prints a copy of the review text, along with the the predicted sentiment for each review according to one of four categories – “Confident Negative” (output value less than 0.25), “Possible Negative” (output value 0.25 – 0.5), “Possible Positive” (output value 0.5-0.75) or “Confident Positive” (output value 0.75-1). You should also print the raw output value alongside this categorisation.

Remember – when reading in from a csv, each row will be a list of comma separated entries. Even if the row just has one entry – it will still be stored in a list at position row[0].

To generate predictions on new, unseen text, we need to call the predict() method of the exported model, and feed in a review(s) as a list of strings (if you’re just predicting one review at a time, you should still pass it in as a list containing just a single review).

Play around with new reviews and the model. Try changing some of the aspects of the model (take a copy first). For example :
- try halving and doubling the number of training epochs, and changing the EarlyStopping thresholds and patience. What happens to accuracy (training, validation and test)? Why do you think this is happening?
- try adding another hidden layer to the network
- try changing the number of hidden units in the hidden layer(s)
- try changing the batch size – make it much bigger, or much smaller (be careful when making it smaller, as compute time will increase significantly). Discuss what you think’s happening.
- try changing the training / validation split

Try writing some mixed reviews too. How does the model perform for these kinds of reviews?

Can you train the model to better predict your new reviews? Work in groups – you have 1 hour.

A Further Challenge

Let’s consider another potential challenge in sentiment analysis.

Consider the following fictional data from a patient survey question asking “What did you feel about your stay in the hospital?”. Would you rate this as “Positive”, “Negative” or “Neutral”, and why?

“The care from the staff was absolutely fantastic – they made me feel safe and nothing was too much trouble. But when I was discharged I had to wait over 5 hours for my prescribed medications at the pharmacy – this is atrocious.”

Aspects

Clearly, there are both positive and negative elements to the above review.

If we were to consider the overall review as **positive**, then we are making the assumption that the care from the staff was more important to the reviewer than the wait at the pharmacy.

If we rate it as **negative**, then we are assuming that the wait at the pharmacy was more important than the care from the staff.

If we rate it as **neutral**, then we are assuming that both aspects are weighted equally by the reviewer, such that their overall feelings about their stay are considered neither positive nor negative.

So, what can we do…?

Aspects

There are two primary solutions that we could adopt :

- We could have a dual output to our sentiment analysis, such that a review can be considered both positive and negative.

- We can break down the review into the individual **aspects** (e.g. the care from staff, the wait at the pharmacy) and analyse the sentiment of these individual aspects instead. This is known as **Aspect-Level Sentiment Analysis**.

Both approaches are valid, and, as ever, it depends on what you're hoping to achieve.

If we're trying to understand what things were good and what things were bad (vs just classifying reviews), then Aspect-Level Sentiment Analysis may provide a better solution.

Aspect-Level Sentiment Analysis

The key difference with Aspect-Level Sentiment Analysis is that we need to split our text into different *aspects* first. Once we've done that, we can simply analyse sentiment in the same way we would for any text.

Consider the following paragraph. What are the individual aspects in this text?

I tried to get through using the supplied telephone number first, but nobody answered after 10 minutes so I gave up. I then decided to go the surgery in person, but I was waiting for ages here too. The receptionist I eventually spoke to was very rude. When I finally saw a nurse, she was incredibly helpful, and put my mind at ease. I thought the coffee from their coffee machine was excellent too – makes a change!

Aspect-Level Sentiment Analysis

I tried to get through using the supplied telephone number first, but nobody answered after 10 minutes so I gave up. I then decided to go the station in person, but I was waiting for ages here too. The desk sergeant I eventually spoke to was very rude. When I finally saw an officer, she was incredibly helpful, and put my mind at ease. I thought the coffee from their coffee machine was excellent too – makes a change!

Aspects :
1. The wait trying to get through on the telephone
2. The wait when attending the surgery in person
3. Rudeness of the receptionist
4. Helpfulness of the nurse
5. Quality of the coffee from the coffee machine

Back to Noun Phrases

So how do we find aspects automatically? One option might be to look for **Noun Phrases**.

Remember, noun chunks are phrases that have a noun as their head – the noun + the describing words around that noun.

For the next 5 minutes, identify the Noun Phrases (and head nouns) in our paragraph. Stay on the Zoom call whilst you do this :

I tried to get through using the supplied telephone number first, but nobody answered after 10 minutes so I gave up. I then decided to go the surgery in person, but I was waiting for ages here too. The receptionist I eventually spoke to was very rude. When I finally saw a nurse, she was incredibly helpful, and put my mind at ease. I thought the coffee from their coffee machine was excellent too – makes a change!

Back to Noun Phrases

Identify the Noun Phrases in our paragraph :

I tried to get through using the supplied telephone **number** first, but nobody answered after 10 **minutes** so I gave up. I then decided to go the **surgery** in person, but I was waiting for ages here too. The **receptionist** I eventually spoke to was very rude. When I finally saw a **nurse**, she was incredibly helpful, and put my **mind** at ease. I thought the **coffee** from their coffee machine was excellent too – makes a change!

Back to Noun Phrases

Let’s compare the noun phrases we identified with the aspects we identified :

NOUN PHRASES
the supplied telephone number
10 minutes
the surgery
the receptionist I eventually spoke to
a nurse
my mind
the coffee from their coffee machine

ASPECTS
The wait trying to get through on the telephone
The wait when attending the surgery in person
Rudeness of the receptionist
Helpfulness of the nurse
Quality of the coffee from the coffee machine

Back to Noun Phrases

The noun phrases we've identified have found all of the aspects that we identified. Therefore, to undertake aspect-level sentiment analysis we could :

- look for noun phrases
    SpaCy has built in functionality to do this. When we run the article=nlp(raw_read) command, as we did before for Named Entity Recognition, SpaCy applies its magic. The list of identified noun chunks out is stored in article.noun_chunks

- apply sentiment analysis to the sentences containing the identified noun phrases

Back to Noun Phrases

Let’s look at a simple example of how we might do this. Can it improve the quality of sentiment analysis for my review :

“The acting was great. The special effects were rubbish.”

Let’s take a look at aspect_level_sa.py and find out!

A word of caution

When undertaking sentiment analysis in the real-world, be mindful of a very real issue that you may encounter.

Any form of machine learning needs a good representation of all sides of the data to learn effectively. So, in the case of sentiment analysis, we need a good sample of positive and negative (and neutral, if we're including it) text.

But, in some cases, you will find that the vast majority of your data is overwhelming positive, for example. Which means that the machine will simply learn to say that everything is positive, because it'll be right most of the time by doing that.

Exercise 2

In your groups, spend the next 15 minutes discussing how you could refining the approach we demonstrated to aspect-level sentiment analysis (finding aspects via noun phrases, and then applying sentiment analysis on the sentences containing those noun phrases).

Can you think of other ways you could approach this, or ways to refine the above method to get rid of irrelevant aspects, or duplication (where the same aspect is mentioned in multiple sentences).