Natural Language Processing

Natural Language Processing

Dr Daniel Chalk

What is Natural Language Processing?

Natural Language Processing (NLP) is focused on the way in which human language and computers interact.

There are two key sides to Natural Language Processing :

- Trying to get computers to understand natural language (speech recognition, building AI that can understand human interaction etc)
- Getting computers to learn how we form our language so that they can help us automate information extraction from free text

We focus on the second of these areas.

Automating free text information extraction

There are three key ways in which we can automate the process of extracting information from free text :

1. Named Entity Recognition – extracting named entities (such as people, places, organisations etc) from unstructured text, and categorising them.

2. Sentiment Analysis – automatically identifying whether unstructured text is positive, negative or neutral in tone. Can also work alongside Sarcasm Analysis.

3. Topic Modelling – automatically grouping text data into clusters based on their similarity, to automatically identify topics.

An Example

Let’s consider an example.

Let’s imagine we have a hospital that has asked patients to fill out a survey about their experiences in the hospital. If the survey contains free text comments, then we may want to :

- Identify whether or not their comments are positive, negative or neutral (Sentiment Analysis)
- Identify specific named entities that they talked about positively, negatively or neutrally (Named Entity Recognition)
- Identify clusters of similar reviews to identify common themes (Topic Modelling)

Named Entity Recognition

What do we mean by a “Named Entity”?

A Named Entity is a real world object that can be denoted with a proper name. The entity can have either physical existence (e.g. Dan Chalk, University of Exeter) or abstract existence (Discrete Event Simulation, Woodwork).

Types of Named Entity

There are lots of potential categories for Named Entities. SpaCy is a Python package we will be using in this course, and has its own set of categories (to which users can add their own) :

PERSON	People (including fictional)
NORP	Nationalities or Religious or Political Groups
FAC	Facilities (Buildings, airports, bridges etc)
ORG	Organisations (Companies, institutions etc)
GPE	Geo-Political Entities (Countries, cities, counties)
LOC	Non-GPE locations (mountain ranges, lakes etc)
PRODUCT	Objects, vehicles, foods etc (not services)
EVENT	Named storms, battles, wars, sports events etc
WORK_OF_ART	Titles of books, songs, films etc
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or periods
TIME	Times smaller than a day
PERCENT	Percentages, including the “%” character
MONEY	Monetary values, including the unit (e.g. £2.50)
QUANTITY	Measurements, such as weight or distance
ORDINAL	”First”, “Second” etc
CARDINAL	Numerals that do not fall under another type

Exercise 1

Let’s practice our understanding of named entities with an example.

Study the following text. Identify all of the named entities in the text, and categorise them according to the SpaCy categories listed on the previous slide. If you feel that none of the existing categories adequately categorises a named entity, then create and name your own category.

You have 15 minutes.

Exercise 1

Yesterday, Derek went to Waitrose and purchased a loaf of bread, and a pint of milk. It cost him a total of £1.30, which he thought was quite reasonable. When he was in France a few weeks ago he felt that bread was priced a little too high. He's often wondered if he should learn a bit about bread making, but he'd have to learn the basics of cookery first!

Noun Phrases

Now we know what Named Entities are, how do we get a computer to find them?

We need rules that we can give the computer that explain where to find Named Entities.

As Named Entities are all “things” (nouns), to find Named Entities we need to look for “Noun Phrases”.

Noun Phrases

Noun Phrases are phrases that are either headed by a noun or an indefinite pronoun (more on that shortly), or perform the same grammatical function as a noun.

Put another way, a noun phrase includes a noun (e.g. cat) and the modifiers which identify it (my cat, John's cat, the cat with white stripes, the cat who ate all the tuna).

Let's look at some examples.

GRAMMAR IS FUN, REMEMBER! LET'S ALL TRY TO REMEMBER THAT! YES...

Noun Phrases

Examples of Noun Phrases :
(noun phrases are underlined, head noun in bold)

The **staff** were very helpful, but the **food** was terrible.

A green-eyed **Mike** enviously watched Dan's amazing **lecture**.

At many **universities**, **academics** consume vast quantities of **cake**.

Exercise 2

Identify the noun phrases and head nouns in the following sentences :

1. Two men walked into their local bar, and thought that their situation sounded like the start of a joke.

2. Many students felt that Dan’s tutorial was the best, but then again, people are often wrong.

3. Most people tend to use Windows, but increasingly people are turning to Linux, as it is a much better Operating System.

4. Suet can be used to make many different traditional dishes.

5. I think my black cat thinks I’m rather dull, as he often yawns when I’m talking about my daily routine.

6. The above sentences show the random stream of rubbish that enters my head at any one time.

You have 10 minutes.

Definite Noun Phrases

GRAMMAR IS FUN. KEEP REMEMBERING THAT.

A **definite noun phrase** is one where the head noun clearly refers to something specific - something previously mentioned or known, something unique, or something being identified by the speaker. The head noun is prefaced by the **“definite article”** (“the”) (potentially followed by adjectives etc – we’ll come back to that).

Examples : I ran over to the dog. I sat next to the very fluffy cat.

Indefinite Noun Phrases

WOW THIS IS SO MUCH FUN.

An **indefinite noun phrase** is one where the head noun refers to a singular noun where the reader / listener does not know exactly which one to which we are referring. The head noun is prefaced by the **“indefinite article”** (“a” / “an”) (potentially followed by adjectives etc – we’ll come back to that).

Examples : I ran over to a dog. I sat next to a very fluffy cat.

Back to Noun Phrases

So why should you care about any of this? (Other than its obvious fun-conferring benefits)

Because Named Entities are found in…

…can anyone guess…?

Back to Noun Phrases

So why should you care about any of this? (Other than its obvious fun-conferring benefits)

Because Named Entities are found in…

…can anyone guess…?

Definite Noun Phrases.

So if we're looking for Named Entities, we need to find Definite Noun Phrases.

Not quite as simple as it may seem...

We know that named entities are found in definite noun phrases, right?

And we know that definite noun phrases will have head nouns prefaced by “the”.

So it should be easy to identify them automatically right? We can just look for text that starts “the” followed by a noun…?

Not quite as simple as it may seem...

WRONG!!

Not quite as simple as it may seem...

Why do you think it isn’t that simple…?

Not quite as simple as it may seem...

There are two core problems here :

1) We don’t just refer to nouns by themselves, we may use adjectives, for example to describe the nouns (e.g. we might say “the cat” or “the fluffy cat” - we’d need to pick up both)

2) We need to identify the head noun in a noun phrase, and there may be more than one contender (e.g “The vicar, Brian” “My cat, Bob”)

Let’s consider the first problem initially. To do that, we need to understand a bit more about grammar, about parsing, and about “regular expressions”.

Parts of Speech - POS

Parts of Speech (or POS) refers to the categories that we assign to words that describe their syntactic functions.

There are 9 Parts of Speech in the English Language. They are (FUN?):

noun	name of a person, place, idea or thing
pronoun	used in place of a noun (eg me, he, she)
verb	an action, or expresses being
adjective	describing words for things–modifies noun or pronoun
adverb	as above, but for modifying verbs
determiner	limits or determines a noun (2 dogs, the rabbit)
preposition	governs a noun or pronoun to express a relation with another word or element in the clause (by, to, at etc)
conjuction	joins words, phrases or clauses (and, but, when etc)
interjection	word used to express emotion (Hey!)

The labelling of words according to their POS category is known as “POS Tagging” in Natural Language Processing.

Exercise 3

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

THIS IS FUN! **THIS IS FUN!** **THIS IS FUN!**

Exercise 3

POS Tag the following paragraph. You have 20 minutes :

The grey horse in the field was slowly eating some grass, when she looked up and saw a bird fly onto the fence. The bird was then joined by 400 more birds. Wow! Clearly the author had been watching too much Alfred Hitchcock.

Parsing

Parsing refers to the process of breaking down a sentence into its component parts and describing the syntactic roles of each component.

For example, through parsing we could identify the noun phrases in some text, and the POS Tags.

We commonly use Regular Expressions to help us in the Parsing process.

Regular Expressions

A Regular Expression (or regex) is a sequence of characters that define a search pattern. They are used in many applications, including search engines, to define the rules for the pattern of text that you want to search for.

When parsing, we can use regular expressions to specify rules for identifying certain syntactic structures in sentences.

Regular Expression Syntax

When forming regular expressions, we use certain syntax. Here is some of the common syntax we use for parsing :

{ }	denotes the start and end of a regular expression
<VBD>	match a word with POS tag of VBD (VBD = Verb, Past Tense)
x?	match ‘x’ optionally
x+	match a sequence of 1 or more ‘x’s
x*	match a sequence of 0 or more ‘x’s
x|y	match x or y
.	match any character
(x)	treat x as a group

So, for example, we could use this to form rules to find definitive noun phrases :

```
{<NN>}          # match noun on its own (e.g. cat)
{<JJ>?<NN>}     # match an optional adjective followed by a noun
```

But, can anybody spot any problems with this approach?

Issues with the regex approach

Some potential problems :

1. This is a pain in the bottom. It turns out there are HUGE numbers of potential combinations of POS Tags that could form definite noun phrases. And some of the rules you write may find things that AREN'T definite noun phrases, as well as ones that are.

Issues with the regex approach

Some potential problems :

2. Let's imagine we're looking for named entities in notes made by paramedics at the scene of an incident. Which of these do you think is a more realistic reflection of the kind of notes you'd expect to see :

a) It was morning, and a slight chill was in the air. I arrived to find that Mrs Smith had clearly taken an unpleasant fall, and was in need of help. Quickly, and without a moment's thought, I rushed to her and asked if she was in any pain.

b) Arrived AM. Mrs Smith had fallen, rapid assistance given. Asked if she was in pain.

Clearly b) is more representative. Because it's rare than in these kinds of situations (or even situations where we're filling out surveys etc), that many of us would write in perfect grammar. But the regex approach necessarily assumes that we do.

An alternative solution

So if the regex approach to finding definite noun phrases isn't ideal for our problem of finding named entities, because it is too difficult and too reliant on the text following consistent grammatical rules, what can we do?

Fortunately, there is an alternative : SpaCy – a Python library for Natural Language Processing that includes, amongst many other things, AI-based methods for Named Entity Recognition.

We'll cover the use of SpaCy for Named Entity Recognition in the next session 11B – Named Entity Recognition in NLP. But here's a brief intro to how it works.

SpaCy

SpaCy is a well regarded and widely used Python library that makes many Natural Language Processing tasks, particularly parsing-based tasks, very easy.

https://spacy.io/

To install SpaCy, just type **pip install spacy** from your command prompt or terminal.

SpaCy for Named Entity Recognition

Let's talk a little bit about how SpaCy works for Named Entity Recognition.

Rather than use regex-based rules to try to search for patterns that might indicate definite noun phrases, SpaCy uses an AI-based approach to *predict* whether a word or group of words based on the word (and POS Tag of the word) itself, and the words (and their POS Tags) that come before and after it at various distances (including punctuation).

It does this based on prior learning on corpuses (collections of written texts) where it has learned the kinds of syntactic patterns it expects to see when a Named Entity is present.

Neural Networks

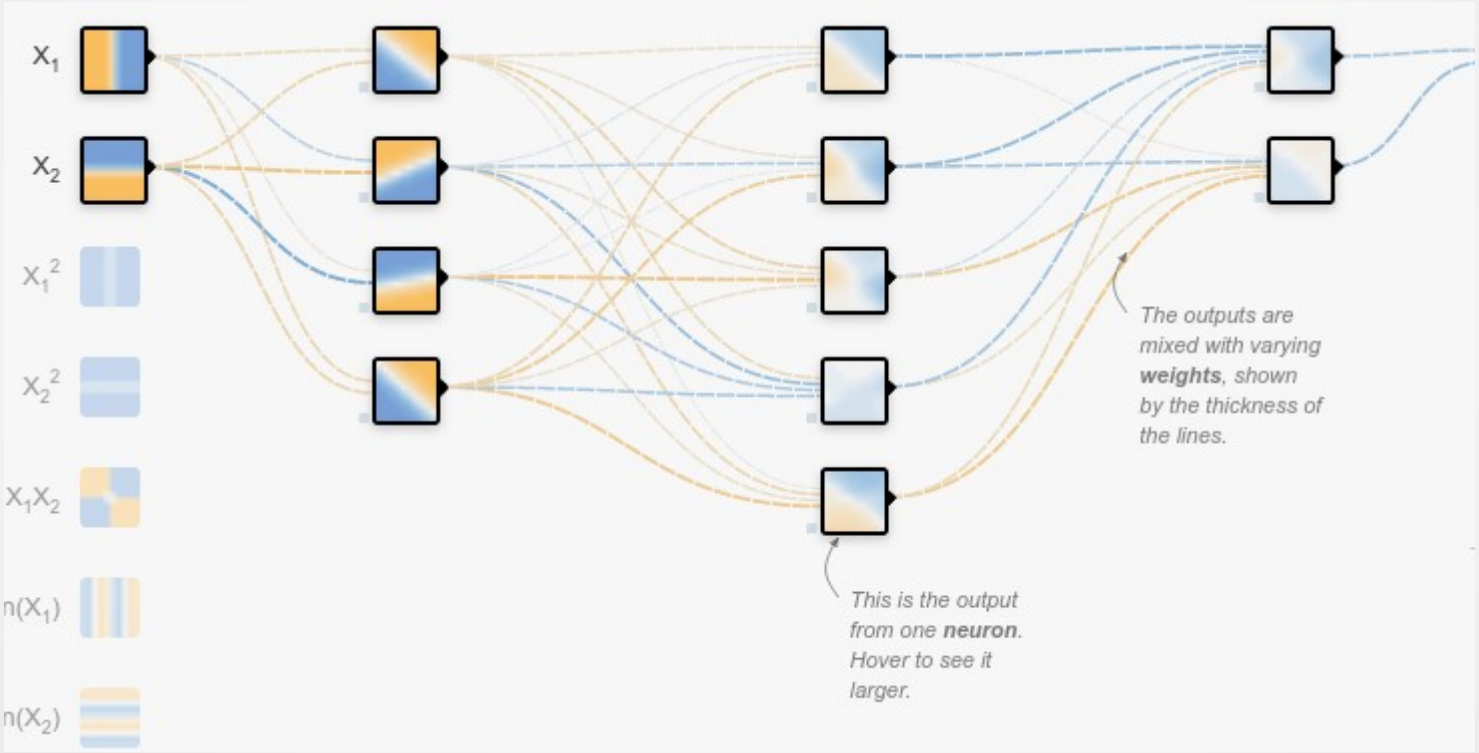
SpaCy learns using something called a **Convolutional Neural Network**. We're not going to go into detail about this at this stage, but we will just give you a very basic understanding of what a Neural Network does.

Neural Networks

Neural Networks are setup to emulate the way in which a brain works – *neurons* connected by *synapses* with *signals* being sent along the synapses to the neurons.

Neural Networks have multiple *layers* of *nodes* (emulating neurons) and *connections* between these nodes on different layers (emulating synapses), with inputs to and outputs from nodes being sent as signals to the other nodes, and a *weight* associated with each signal.

Neural Networks



https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.48682&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Neural Networks

In SpaCy, the inputs to the neural network are the POS-tagged groups of words with a corpus, converted to numbers, and it's trying to adjust it's weights and internal functions so that when it's told, for example, that word 3 in the group of words is a Named Entity (an output of 1), it is able to come to the same output based on the POS Tags of Word 3, word 2, word 4, word 1 etc.

Key takeaway message :

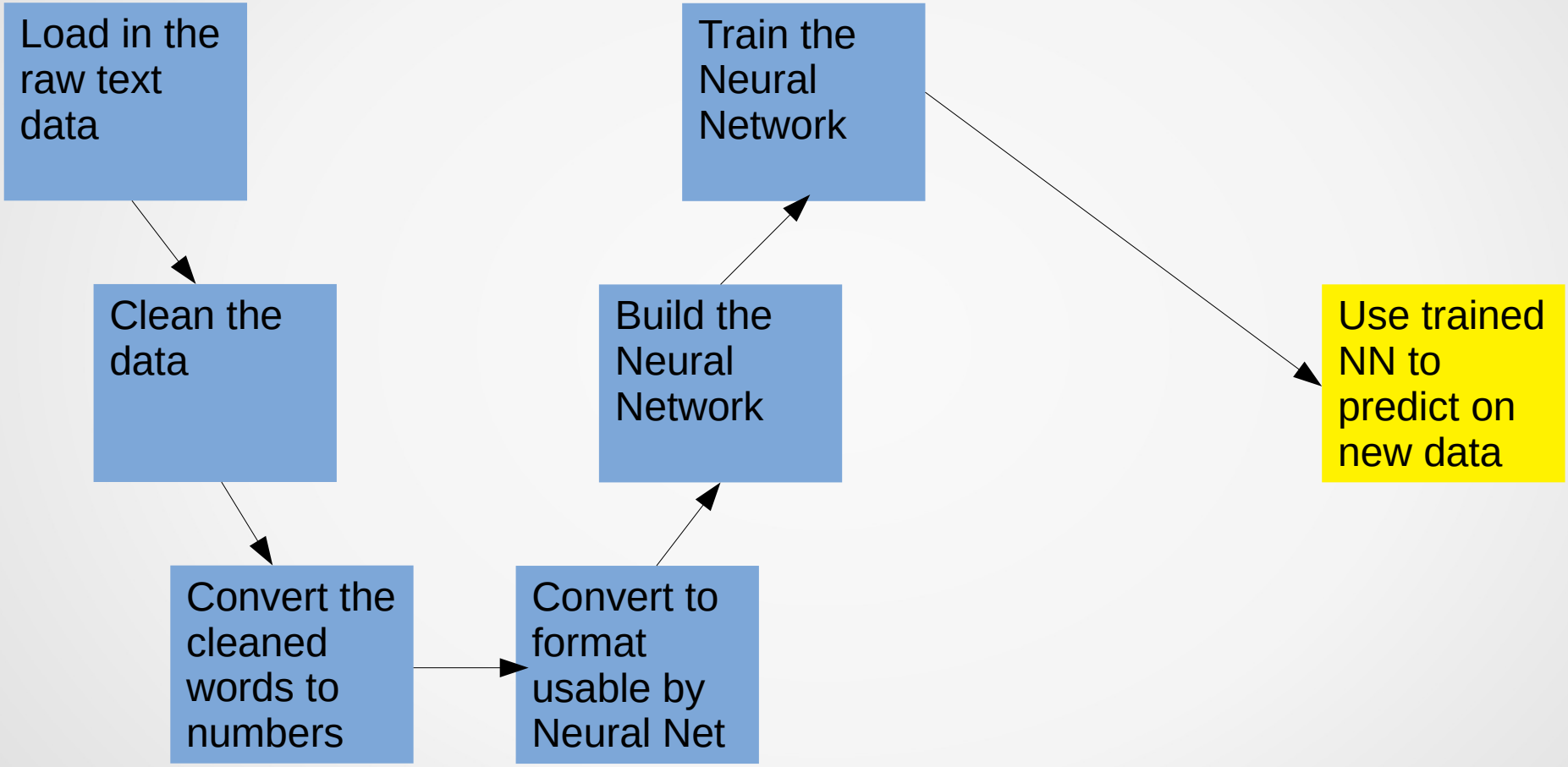
SpaCy predicts whether a word / words represents a Named Entity based on prior learning from massive amounts of text, where it has learned the kinds of syntactic patterns that surround a Named Entity.

Sentiment Analysis

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), negative (-1) or neutral (0).

Sentiment Analysis – The Overall Process



Exercise 4

For the next 20 minutes, in groups I want you to think about :

1) what are some potential applications of Sentiment Analysis in your own organisations?

2) what might be some of the challenges involved in Sentiment Analysis?

Applications and Challenges 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

A potential challenge - Sarcasm

“I loved having to wait for 5 hours to see a doctor”

“I was extremely impressed – I’d never seen a restaurant with more dirt per square inch”

“Well done! You managed to ignore every request for help. Quite an achievement!”

I would think most of us would consider the above sentences to be sarcastic. But because there is positive language, a machine may well find it difficult to identify that these are not positive statements (even we humans struggle sometimes).

Reflect on how you know the above sentences are sarcastic. Why do you think they are?

Sarcasm Detection

Sarcasm Detection is a branch of NLP that tries to use AI methods to learn when text is sarcastic in tone.

The most effective way to detect if a statement is sarcastic is to compare it with known truths. For example, if we know that people don’t like to wait for services, then the statement “I loved having to wait for 5 hours to see a doctor” can be seen as unlikely to be true, and therefore sarcasm.

But, as you can imagine, this means Sarcasm Detection is not a trivial task, and a better solution is likely to take samples of your data to determine the % of cases that demonstrate sarcasm. In the majority of cases, this is likely trivial.

But do check out : http://www.thesarcasmdetector.com/

A Further Challenge

There is another challenge in sentiment analysis, relating to what the sentiment of text actually refers. We'll talk more about that, and show you how to undertake sentiment analysis, in session 11C – Sentiment Analysis.