

Natural Language Processing CS Challenge Diary

“Develop a tool to **define human understandable patterns** in sentences which can be used to **extract structured information**. For example, a sentence might have adjectives and **job titles** describing a person. **Examine existing work** to explore how these human understandable rules could be automatically inferred using AI techniques applied to **labelled sentence examples**. For example, using **decision trees**.”

Acronyms:

KDD: Knowledge Discovery in Databases

DEFT: Deep Exploration and Filtering of Text

NLTK: Natural Language ToolKit

Definitions (Updated as of 29/01/2020):

The idea of this section is that while I’m reading through documentation, I’ll write the word here and afterwards, go looking for an explanation of the word.

Text Mining: The mining process applied on free text to discover actionable insights, useful information and patterns within it.

DEFT: Access implicitly expressed, actionable information within a *corpora*.

Tokenizing: Grouping tokens together. Word Tokenizers separate a *corpora* by word. Sentence Tokenizers separate a *corpora* by sentence.

Lexicon: A grid of words and their various meanings depending on the context in which they are used.

| Lemma | Topic | Rating |
|-------------|----------|--------|
| waiter | service | |
| waitress | service | |
| wait | | bad |
| quick | | good |
| .*schnitzel | food | |
| music | ambience | |
| loud | | bad |

Figure 1: Sample Lexicon. Source: Introduction to Sentiment Analysis by Thomas Aglassinger

Stopwords: Words which have little to no effect on the overall meaning of the sentence. These are removed before processing large datasets however, leaving them in while looking at smaller datasets won’t hinder your model.

Stemming: Used to normalise similar sentences. For example, the tenses within two similar sentences may not affect their overall meaning. A disadvantage to this process is that sometimes, the results can non-existent words.

Lemmatizing: Grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word’s *lemma*.

Lemma: How a word appears in the dictionary.

Training Data: Used to teach the machine to perform a process.

Sample Data: Used to test the machine.

Polysemous: The coexistence of multiple meanings for the same word or phrase.

Disambiguation: Determine the correct category from a set of possible categories; determine the pronunciation of a letter based on what letters surround it, determine the POS of a word based on the contextual words surrounding it, determine where to attach a prepositional phrase based on the set of other phrases, determine the meaning of a polysemous word in a given context.

Segmentation: Determine the correct boundary of a segment from a set of possible boundaries.

Part of Speech Tagging: Training a machine to determine what part of the target language a single word in a *corpora* belongs to; Proper Noun, Possessive, Preposition etc.

Sentiment Analysis:

Chunking: Identify the named entity within a *corpora*, then identifying other words within the *corpora* that affect that named entity.

Pipeline: Turns text into a Doc object. By default, the processes carried out are tokenization, dependency parsing and entity recognition.

Tuple: an ordered sequence of values.

Methodology:

Given that the requirements for the final product are still unclear, I do not know which methodology will be most effective. This is a point for discussion.

**Update*:* As of the meeting on 06/02/2020, I'm calling my approach to this project 'The Log-Cabin on Everest' methodology which is the idea that meeting the end goal of this project is as hard as climbing Everest and that we should aim to make something of lasting value to other people which will help them when they try to scale the mountain (meet the end goal of the project).

Introduction to Pre-Processing 18/01/2020

Pre-Processing describes how the data is manipulated before it is passed into a Neural Network.

I will use `sent_tokenize` and `word_tokenize` from NLTK to group tokens together. Word Tokenizers separate a *corpora* by word. Sentence Tokenizers separate a *corpora* by sentence. In `nlTK`, tokenizing will break a *corpora* up into strings.

Eliminating 'stopwords' improves efficiency of the algorithms that handle the data later on. These words have little to no effect on the overall meaning of the sentence.

Chunking vs Chinking

Chunking uses regular expressions to identify parts of a *corpora* which satisfy the regex. This is mainly used to group words into 'Noun Phrases'; the idea that you group a noun with words that are related to/affecting it.

Chinking affects a Chunk. You remove a small part of your Chunk that you do not want, then you keep the larger Chunk.

It is likely that I will use both of these processes during the pre-processing of data. I had to consult a regex cheatsheet because I am unfamiliar with the process of defining a regex.

19/01/2020: Researching Ideas

Seeking Training Data

I am still unsure as to the exact dataset that I want to process and extract insight from. I have considered using the Tweepy library to web-scrape recent Tweets which include a keyword; perhaps the name of a politician or location. I feel that this would be most insightful as analysis of a current affair could be carried out to gauge how the public feel about a decision. However, my doubt regarding this style of project is that other developers have done this before. I would need to identify a way of making my project unique. Perhaps I could add in a factor to the sentiment analysis such as the number of interactions with the tweet since it had been posted. Ideally, I would like the machine to interpret these tweets and decide whether it is for or against the motion in question and to explain to me why it came to that decision.

An advantage to using Twitter is the brevity of each tweet which makes them quicker to process as opposed to news articles that would be scraped from news sources. However, using social media to gauge the political climate raises the uncertainty that comes from rumours and speculation being posted, which could influence the machine, especially if words with a high sentiment score are using such as expletives. Exaggerations and sarcasm also pose a challenge for the process of sentiment analysis. Having realised this potential setback, I investigated the approaches that PwC are taking to overcoming it.

PwC Case Study 1: Sarcastic Tweets affecting Elections

Phil Mennie, Social Media Governance Leader from PwC explains in the article linked below, Twitter “hashtag hijacking” is a risk that organisations face on social media. UKIP published “#WhyImVotingUKIP” in an attempt to encourage voters to explain to others why they are voting for this party. However, users who disagreed with those values used the hashtag to poke fun at the party. For a machine, it is highly challenging to distinguish between serious remarks and hyperbolic or sarcastic remarks, given that he struggled to interpret some of them. Phil suggests that analysing a user’s previous tweets may indicate whether the tweet in question is sarcasm or not.

The article encouraged me to look beyond the tweet when searching for text to analyse. Considering factors such as the demographics of the user and who they follow may indicate their political persuasion.

<https://www.pwc.co.uk/services/risk-assurance/insights/can-sentiment-analysis-spot-sarcasm.html>

PwC Case Study 2: SocialMind NLP Tool

According to the Consumer Financial Protection Bureau Supervision and Examination Manual, penalties for unfair or deceptive practices using social media can reach \$1,000,000/day. PwC’s Analytics services state that a social media presence is not enough to ensure marketing success. How a brand is perceived on social media can either positively or negatively affect the brand’s reputation considerably. Regulators have started using social

media to detect unlawful practices. Dissatisfied customers on social media can be disastrous for business performance. Their negative experience will deter others from using services from the brand in question.

SocialMind was developed to provide insight to the client about what customers are saying about their services, as well as their competitors'. It listens to all relevant social media data and client-specified websites. Dashboards are compiled after the data has been processed to visualise its results.

I feel motivated to implement elements of this tool in my project such as using additional data to just the text from the Tweet to extract information. Presenting the information in the form of an interactive dashboard makes the processed information more readable.

<https://www.pwc.com/gx/en/issues/innovation-technology/analytics/assets/social-mind.pdf>

Named Entity Recognition

This process identifies proper nouns such as "John" and "Smith".

WordNet is a lexical database designed by Princeton and implemented in NLTK. It can be used to find the meanings of words, synonyms and antonyms.

If you input a word, WordNet can return its definition. If there are multiple definitions of that word, it will return the first one unless a second argument is entered.

You can print a list of synonyms or antonyms for that word.

WordNet: Similarity

The NLTK library allows you to compare two words and return a decimal of how similar they are to each other. A score towards 1 suggests a synonym. The lower the score, the more likely it is that the other word is an antonym.

Text Mining: Docx

Python has a python-docx library which facilitates the generation and manipulation of .docx documents. It is likely that I will use this library to output a written report after having tested for sentiment analysis, identifying which words appeared most times in positive product reviews and vice versa.

Developer Diary Entry 22/01/2020: Mathematics 1

Following research carried out on <https://pythonprogramming.net>, the first algorithm to investigate is the Naïve Baye classifier. It is often used in text classification. It assumes that the presence of a feature in a class is unrelated to the presence of any other feature. This algorithm doesn't require much processing.

Posterior = Prior Occurences * Likelihood / Evidence

The diagram illustrates the Naive Bayes Formula. At the top, 'Likelihood' points to $P(x|c)$ in the numerator, and 'Class Prior Probability' points to $P(c)$ in the numerator. The denominator is $P(x)$, which is labeled 'Predictor Prior Probability'. The entire fraction is labeled 'Posterior Probability' with an arrow pointing to $P(c|x)$. Below the fraction, the expanded formula is shown: $P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Figure 2: Naives Baye Formula

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Training vs Testing

It is good practice to use two datasets (A and B) where A is used as the training set and B is used as the testing set. This eliminates bias.

What have I covered so far? (Up to 23/01/2020)

NLTK Programming Theory

- Tokenizing
- Stopwords
- Stemming and Lemmatizing
- Chunking and Chinking
- Part of Speech Tagging
- Training data vs Testing data
- Named Entity Recognition
- Corpus pre-trained Datasets
- WordNet lists, definitions and similarity scoring

Miscellaneous Programming Theory

- OS library basic features
- Python-Docx basic features
- Regex

Mathematics

- Naïve Bayes Equation

Real-life Implications and Examples of NLP:

- Viv.ai
- SocialMind
- Overcoming the processing of sarcastic/hyperbolic data

Current Goals for the week 26/01/2020 – 02/02/2020:

- ~~1. Identify the target data I will process with my tool~~
- ~~2. Fix teething issues with spaCy and Power BI* (Get the English language toolkit working and acquire a licence for Power BI)~~
- 3. Begin learning about applying sentiment analysis**
4. Investigate how to use Flask and SQLite for making a Webserver.
5. Learn the fundamentals of Tweepy or Scrapy
6. *Carry out research using various academic sources to learn more about decision trees.*

27/01/2020

2. Overcome a Challenge 1: Spacy's en_core_web_sm platform not linking

Source of challenge: The platform was being downloaded to the Python 3.8 environment but the Spacy package is stored in the 'spacy' environment. In the admin console, these are the steps to resolve the issue.

```
(base) C:\Windows\system32>activate spacy
```

```
(spacy) C:\Windows\system32>C:
```

```
(spacy) C:\Windows\system32>cd C:
```

```
C:\Windows\System32
```

```
(spacy) C:\Windows\system32>cd ..
```

```
(spacy) C:\Windows>cd ..
```

```
(spacy) C:\>cd Users
```

```
(spacy) C:\Users>dir
```

```
(spacy) C:\Users\rjdp>python -m spacy download en_core_web_sm
```

Result:

```
Requirement          already          satisfied:          en_core_web_sm==2.0.0          from
https://github.com/explosion/spacy-models/releases/download/en\_core\_web\_sm-2.0.0/en\_core\_web\_sm-2.0.0.tar.gz#egg=en\_core\_web\_sm==2.0.0          in
c:\users\rjdp\miniconda3\envs\spacy\lib\site-packages (2.0.0)
```

Linking successful

```
C:\Users\rjdp\Miniconda3\envs\spacy\lib\site-packages\en_core_web_sm
```

```
-->
```

```
C:\Users\rjdp\Miniconda3\envs\spacy\lib\site-packages\spacy\data\en_core_web_sm
```

You can now load the model via `spacy.load('en_core_web_sm')`

26/01/2020 Fundamentals of Spacy

Training a model is ideal if I want my tool to generalise based on examples provided. It can prove effective if you are tracking a named entity.

A rule-based approach is more useful if you can express a pattern with a regex, such as tracking an IP address. Regexes operate on single tokens rather than corporas.

Sentiment Analysis

<https://ep2018.europython.eu/conference/talks/introduction-to-sentiment-analysis-with-spacy>

Thomas Aglassinger notes that it can be useful to make an enum of whatever factors you are assessing sentiment on eg; for product reviews, factors such as durability, value for money, cost effectiveness, dispatch time etc may be ideal headings.

There are 3 common ways of expressing sentiment; Positive/Negative, Star Rating, Float. The most accurate is float, so I will be using it.

From 0.0 to 1.0, I will use these standards:

| RATING | MEANING |
|--------|-----------|
| 0.0 | VERY POOR |
| 0.25 | POOR |
| 0.5 | AVERAGE |
| 0.75 | GOOD |
| 1.0 | VERY GOOD |

26/01/2020 Rule Based Matching Existing Software Case Study 1: Explosion.ai's Matcher

This application lets the user create their own rules and run them against their text. You can specify values, POS tags or Boolean flags. Essentially, it is a high-level abstraction of rule based matching which lets the user construct rules without having to manually code them.

The image shows a user interface for creating rules, divided into three stacked panels. Each panel has a red minus icon in the top left and a red plus icon in the top right. Below each panel is a purple button labeled 'add attribute'.

- Panel 1:** Contains a dropdown menu with 'IS_STOP' selected and a green checkmark icon to its right.
- Panel 2:** Contains two dropdown menus. The first is 'LEMMA' with 'match' selected. The second is 'POS' with 'NOUN' selected.
- Panel 3:** Contains three dropdown menus. The first is 'IS_TITLE' with a green checkmark. The second is 'IS_SPACE' with a green checkmark. The third is 'POS' with 'VERB' selected.

Figure 3: UI for the Rule Creation Tool

```
pattern = [{ 'IS_STOP': True },
            { 'LEMMA': 'match', 'POS': 'NOUN' },
            { 'IS_TITLE': True, 'IS_SPACE': True,
              'POS': 'VERB' }]
```

Figure 4: Result of the criteria from Figure 3

27/01/2020 University Case Study 1: NLP at Washington University “Decision Trees and NLP: A Case Study in POS Tagging”

Part of the Project Specification requires me to analyse existing work. A case study from the University of Washington identifies a linguistic knowledge acquisition bottleneck which suggests that each slightly new NLP variation requires linguistic knowledge bases to be built from scratch. Hence, “automatically acquired language models” would be very useful since training for all grammatical exceptions and sub-regularities is heavily time consuming if someone was to manually encode it. A Corpus-based approach such as NLTK is an abstraction/black-box system simulated by massive statistical tables. This abstract nature means that some linguistic trends and phenomena could be overlooked. In 1995, Magerman states that most NLP problems are classification problems, hence, a machine learning approach is suitable. There are two types of linguistic problems; **Disambiguation** and Segmentation.

This case study has two parts; POS Disambiguation tasks, as well as Unknown Word Guessing. The target natural language is modern Greek, which hasn’t been investigated widely from a computational perspective. Washington recorded a 93-95% performance rate in POS Disambiguation and 82-88% performance rate of guessing the POS of unknown words. Three tree induction algorithms are used to produce decision trees. The first two produce generalised decision tree while the third produces binary decision trees (see Figure 5 below).

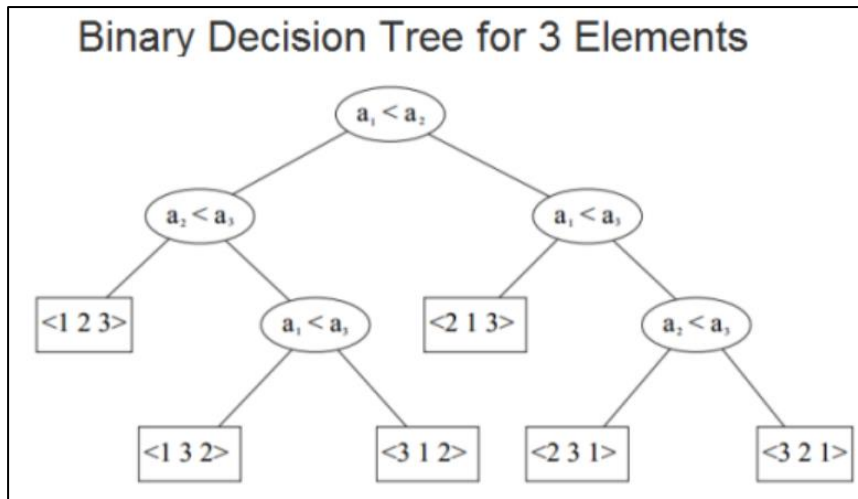


Figure 5: A model Binary Decision Tree

Figure 5 depicts the decisions that are made to identify the order of 3 numbers.

Is Element 1 less than Element 2?

YES: Is Element 2 less than Element 3?

YES: ORDER IS 1, 2, 3

NO: Is Element 1 less than Element 3?

YES: ORDER IS 1, 3, 2

NO: ORDER IS 3, 1, 2

NO: Is Element 1 less than Element 3?

YES: ORDER IS 2, 1, 3

NO: Is Element 2 less than Element 3?

YES: ORDER IS 2, 3, 1

NO: ORDER IS 3, 2, 1

---Analysis of this research paper will continue on 31/01/2020---

28/01/2020 4. Using Flask to make a simple application that runs via a browser (Part 1).

I found a series of videos by Corey Schafer which demonstrate how flask can be applied in the creation of a web application and have closely followed it, given that I have never used this library previously.

https://www.youtube.com/channel/UCCezIgC97PvUuR4_gbFUs5g

Installation:

In Anaconda, type the following:

`activate <name_of_virtual_env>`

pip install flask

pip install flask-wtf

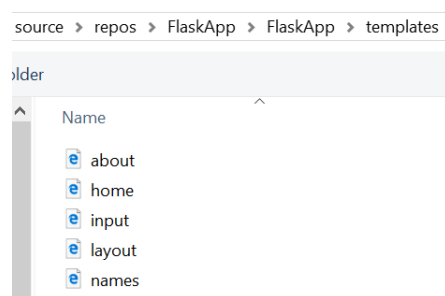
Figure 6 is the source code for the Hello World flask application. It is worth explaining what each important line of code does.

| | |
|---|--|
| <pre>from flask import Flask app = Flask(__name__) @app.route('/') def hello_world(): return 'Hello, World!' if __name__ == '__main__': app.run()</pre> | <div style="border: 1px dashed black; padding: 5px; margin-bottom: 10px; text-align: center;">Import command</div> <div style="border: 1px dashed black; padding: 5px; margin-bottom: 10px; text-align: center;">Instantiate Flask app</div> <div style="border: 1px dashed black; padding: 10px; margin-bottom: 10px;"> <p>Specify behaviour which happens when the '/' route is the URL.</p> <p>In this case, it just prints "Hello, World" onto the screen.</p> </div> <div style="border: 1px dashed black; padding: 5px; text-align: center;">Run the Flask app</div> |
|---|--|

Figure 6: Hello World Flask App

`@app.route("/dir_name")` makes a new webpage which can display html using the `render_template('template_name',args)` function.

My editor is VS Code. There was no templates folder which was created automatically as described in several tutorials. Hence, I created one and populated it with templates which were used by the web pages. One of these templates, `layout.html` was a base template which was extended upon by the others.



Once you have coded or downloaded some templates, you can create a new webpage which calls them in the function for that page's route.

Making a secret key protects against forgery attacks. It is recommended that you use 16 random characters for this. An effective way to generate a security tag is to go to Anaconda and type:

`python`

```
import secrets
```

```
secrets.token_hex(16)
```

29/01/2020 University Case Study 2: Decision Trees at Trinity College, Dublin

The resource I am referencing was published by Dr. Kevin Koidl, School of Computer Science and Statistic Trinity College Dublin and can be viewed at <https://www.scss.tcd.ie/~koidlk/cs4062/Lecture11DecisionTrees.pdf>

This is a summary of the knowledge I have gained from reading Dr. Koidl's notes:

Decision Tree learning is used for inductive inference, the process of reasoning a statement from specific to a generalisation. For example, in a sample, if all the sample cars are black, then there is a hypothesis; "Are all cars black?". You can model any algorithmic decision-making process as a tree. Dr. Koidl reiterates what was stated in case study 1; that there is a linguistic knowledge acquisition bottleneck. Figure 7 describes the parts of any decision tree:

Decision trees **classify instances** by sorting top down.

A **leaf** provides the classification of the instance.

A **node** specifies a test of some attribute of the instance.

A **branch** corresponds to a possible values an attribute.

An **instance** is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example.

This **process** is then repeated for the subtree rooted at the new node.




Figure 7: Parts of a Decision Tree

The ID3 Algorithm is works by taking each attribute and evaluating it to see how well it can classify training examples on its own. The attribute which scores the highest is used as the root node. A descendant of the root node is created for each possible value eg; {"Yes", "No"}, {"A", "B", "C"}. Training examples are placed at their most suitable node. You repeat the process using the training examples at each point in the tree to determine what attribute should be tested at that point. The algorithm never backtracks/reconsiders so it is called 'Greedy search'. "Information gain measures how well a given attribute separates the training examples according to their target classification." This is the best way to quantify the worth of an attribute.

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Questions I would like to ask during the meeting on 30/01/2020

- Is it too late to shift from C ii) to C i)? The skills that I have acquired so far and research I have carried out will still be useful for the project which C i) describes, however that project has a more definite structure to it with more deliverables. Considering the 12-week time limit on this project, I would be much more comfortable working something that has a more measurable sense of progression. I also feel that this project is more coherent with my other modules, since Databases involves lots of SQL work. Having

thought about it a lot, I just feel that the quality of work I will produce will be so much greater and that it will benefit my Databases module as well.

Review of Meeting: 30/01/2020

It would have been unwise to change project brief because of how this course is being assessed; how you go about what you produce as opposed to what you produce. I wanted to change because I felt that the other brief had a lower ceiling, so I would have been more likely to finish it entirely.

31/01/2020: The How-to Guide

There are several high-quality SpaCy guides available which cover the theory behind the library in detail. It would be unrealistic to improve the syntax explanations provided on the SpaCy website. To produce an alternative guide which is also useful to someone looking to learn about Rule-Based matching, I feel that it would be beneficial to **combine**, then **condense** the information I recorded about this process, then produce a concise series of steps with **troubleshooting**. Interactive Learning is another area that I am interested in. How-to guides which contain only text, screenshots, links and videos do not captivate a beginner as much as ones which let the user try out what they have learnt. SpaCy's official site achieves this through editable code snippets, Explosion.ai. I have under 10 days left to produce this how-to guide so while I would like to implement some form of quality user interaction, this may not be achieved within the time constraint especially since I will need to learn how to link displaCy or an NLTK drawing tool to Flask. This weekend, I will check StackOverflow to see if anyone has done this before. I'm unsure as to how full my troubleshooting page will be.

During my research, I came across interesting research articles from Washington University, PwC and other sources which could benefit someone who now understands the fundamentals of NLP and would like to see them applied in industry/research.

http://faculty.washington.edu/fxia/courses/LING572/decison_tree99.pdf

<https://www.pwc.co.uk/services/risk-assurance/insights/can-sentiment-analysis-spot-sarcasm.html>

In terms of the HTML page that my 'How-to' guide is being published on, I intend for it to be of reasonably good quality, given that I have just under one week to produce it. I am not overly concerned about struggling to finish this webpage at a high standard, having learnt all the necessary Flask and produced material to publish in the past week. As stated above, a key feature of this webpage will be troubleshooting, something that lots of the other tutorials which focus more on teaching the theory omit. I am going to showcase lots of what SpaCy is capable of, but also some examples of the shortcomings of NLP such as dealing with homographs and informal text (eg; tweets).

I have researched the format for a good quality 'how-to' guide on University of Bath's Digital Marketing & Communications guidance portal linked below;
<https://www.bath.ac.uk/guides/creating-a-how-to-guide/>

01/02/2020 Overcoming a Challenge 2: Images not loading onto my Flask website.

When designing the front-end for the webpage, locally stored images weren't loading properly. I tried various approaches to solving this problem; loading the images from a file hosting website, from Google Drive, and from inside the 'templates' folder. Eventually, I realised that images needed passed into the HTML document via 'url_for' from the Flask library. I created a 'static' folder and set the app's url_path to it. This resolved the problem. It was frustrating trying to work out if the images not loading was a directories problem, a HTML problem or a Flask problem. However, this means that in future, I know exactly how to approach this.

01/02/2020 Designing Tutorial 1

As stated previously, the tutorials which cover the theory behind rule-based matching cannot be improved upon in the sense that the definitions are official. However, what I can do is provide example test cases of data which helps a novice user of SpaCy understand HOW and WHY input data causes matches for a given token. I have also included the pre-loaded URL links to the official SpaCy Matcher from Explosion.Ai, encouraging them to tweak the attributes inside the token(s) to see what the effect is.

The most useful tutorials that I learnt from all started out simple and got progressively more difficult with each page/example; a trend that I intend to replicate in the design of this website. There is no point starting off with an explanation of complex cases involving large texts compared against multiple multi-attribute tokens because that will overwhelm them, discouraging them from trying to learn. However, it is also important to make the final examples noticeably more technical than the starting ones.

I feel comfortable designing and explaining test cases which include all the attributes that a token can assume, and which apply tokens to texts which gradually increase in length and complexity.

One decision that I will make tomorrow is whether to include the shortcomings of Rule-based matching on this Tutorial Page, or on Troubleshooting which will be designed on 04/02/2020.

02/02/2020 Creating the code to be referenced in Tutorial 1

Today I created an application called WebsiteTutorial1.py which aims to teach users how Rule-Based Matching works by showing it working forwards (passing pre-written text into the matcher to see if any of its text satisfies a pre-written rule) and backwards (showing the user a rule and getting them to enter text which satisfies it). None of the tutorials I used to learn Rule-Based Matching used an approach to teaching like this, despite how useful and hands-on I feel it is. This way, the user is introduced to Rule-Based Matching with example cases which get gradually more complex and then can test their understanding.

Having read the official SpaCy tutorial, as well as several Rule-Based Matching tutorials on YouTube cited below, I decided that a code walkthrough would also be useful for the reader, especially if the source code was linked for them to edit. I replicated this by hosting WebsiteTutorial1.py on GitHub. On the website, I will annotate the code screenshots to describe the SpaCy features such as the arguments of matcher.add().

In my solution, I also used displaCy as my visualisation tool. Given how useful this feature is, my tutorial will show how it can be implemented in the main "check_for_matches" method to show how each token affects the others. Not all of the tutorials mentioned that you can edit the appearance of the output by passing in an options argument so I will address this and show the effects of tweaking these options. For example, you can enhance the appearance of the visualisation by changing the font.

It is also useful to have displaCy create a separate diagram for each sentence. This is done by using the `.sents` property of the target text to create a list of sentences. In tutorial 2, I will show the effect of using a different style of diagram.

03/02/2020 Creating the code to be referenced in tutorial 2

Named Entity Recognition is the other area of NLP which I feel that I can provide a useful tutorial about. Once again, it requires the SpaCy library as well as the `'en_core_web_sm'` vocabulary. I intend to make a program to teach the reader about the various types of Proper Noun that NER can be used for; Locations, Money, People, Artwork etc. because I am currently working on improving my own understanding about this. Since I have shown a limitation of Rule-Based Matching, it is also useful to show the user a test case which makes it difficult for NER to categorise a word. According to [towardsdatascience.com](https://towardsdatascience.com/spacy-s-ner-engine-was-trained-on-ontonote5/), SpaCy's NER engine was trained on OntoNote5.

My test case for identifying a shortcoming of NER is 'Turkey' (Country) vs 'Turkey' (Food).

"Turkey tastes amazing at Christmas!" returns GPE for Turkey which is incorrect as in this context, it isn't referring to a country. This test case justifies a need for Rule-Based Entity recognition.

Another obscure test case is people with places for names; India is a girl's name as well as a country. Austin is a city in Texas, as well as a boy's name. Other examples are Brooklyn and Adelaide. None of the tutorials described how to go about distinguishing between semantically identical tokens in entity recognition so it would be beneficial to readers if my website addressed this challenge.

However, I am concerned that I will run out of time to implement an Entity Ruler example which solves this, as well as being able to contextualise my approach in the form of a tutorial given that the deadline is in less than 100 hours. Once I finish writing tutorial 1, my focus will shift to making tutorial 2 as engaging as possible. Fortunately, having read about Flask from the start of the project, I am more comfortable in producing a webpage given that the process for tutorial 2 is a replication of tutorial 1 and uses the same layout, but will have a different kind of interactive feature. Therefore, it is a matter of balancing programming the short tutorial application and outlining the steps for others to reproduce it. I intend to overcome this lack of time by dedicating time that would have been spent on improving the aesthetic of my frontend to reading up on possible approaches to creating a function that returns the correct answer for these obscure test cases. I reckon there are two potential approaches to training this tool to identify the correct noun type that these words belong to;

1. Train for the common cases: Create a rule that if the word after 'India' is a name, deduce that India is also a name in this context.
2. Implement another library; NLU, a subsidiary of NLP is about structuring the input so that the machine can act upon it. RASA feels like a good option, given that you can perform intent detection. Intent detection would be perfect in this scenario. For example, "I want to buy a Chinese." returns highly likely that the writer is talking about food as opposed to the nationality.

Unfortunately, RASA server is linked to TensorFlow which would not install for me even after creating a virtual environment for Python which was an older version (3.6). I will need to spend time playing around with Conda to install this package. Given that it's not necessary for the How-to guide, I decided to postpone its installation until next weekend. This was quite frustrating but hopefully I can get it working if needs be.

I watched a YouTube tutorial on training SpaCy for NER but when I was following along, there was a mistake in the code where 'model = None' was showing an 'Invalid Syntax' message despite me doing exactly what he did. I read the comments, thinking "have other people also got stuck on this?" and there are in fact, two other people also couldn't solve it. So I thought that for my tutorial, if I can make a trainer which runs perfectly with no syntax errors, I've made something which benefits at the very least, two other people. Here is a link to the GitHub repository containing that code from the tutorial which doesn't work in Visual Studio:

<https://github.com/Jcharis/Natural-Language-Processing-Tutorials/blob/master/Training%20the%20Named%20Entity%20Recognizer%20in%20SpaCy.ipynb>

The error appears here:

```
## plac is wrapper for argparse
@plac.annotations(
    model=("Model name. Defaults to blank 'en' model.", "option", "m", str),
    output_dir=("C:\\Users\\This PC\\Documents\\JLabs\\JFlow", "option", "o", Path),
    n_iter=("Number of training iterations", "option", "n", int))

# Define our variables
model = None
output_dir=Path("C:\\Users\\This PC\\Documents\\JLabs\\JFlow")
n_iter=100
```

'Unexpected Token: model'

I thought to myself; "That's got nothing to do with model and must be caused by the plac annotations. So I moved those below the variables and the if statement had the exact same error message. For my tutorial, I can exclude plac because I'm not using a notebook style of command prompt and that code has no effect on the NLP code. This will fix the problem. Other people in the comments complained that their model was performing poorly. I wouldn't attribute that to the code, but rather poor-quality training data or them incorrectly labelling where the named entity is, which is easily done. I was manually going through each sentence to find the start and end position of the 'Ronan' which took a while at the start.

Follow-up

This is my meta.json for the model after running it 10 times. I noticed that with each consecutive run, it made less mistakes. By the 7th run, it correctly identified 'Ronan' 100% of the time which is very exciting as SpaCy's existing library never picks up my forename correctly.

```
"accuracy":{
  "token_acc":99.8698372794,
  "ents_p":84.9664503965,
  "ents_r":85.6312524451,
  "uas":91.7237657538,
  "tags_acc":97.0403350292,
  "ents_f":85.2975560875,
  "las":89.800872413
},
```

I've searched around for an answer as to what some of the other fields in accuracy mean but I haven't found anything so I'll post in the NLP Facebook Group next week so that an expert in the field can explain it to me.

03/02/2020 Interactive tool for teaching NER – Writing to displaCy and receiving a displaCy diagram in a new page

I will implement a form on the website which accepts text as an input, then outputs all NER tokens beside their NER type. Much like how the official SpaCy tutorials encourage readers to interact with the site, I feel that interactive learning will make the user understand the theory in a more applied sense. This is achieved by importing the requests tool from Flask and using it to manage GET and POST requests that involve a text box containing the target text.

04/02/2020 Implementing the tool conceptualised on 03/02/2020.

I created the input box on the Tutorial2 form for the user to enter their text. I didn't dwell too much on the appearance of the form; I am working towards a rapidly approaching deadline so demonstrating the results of my way of thinking seems more important. My main challenge today was finding out how show a displaCy render through Flask. The first step is generating the NER render which is achieved through the `render = 'displaCy.render(text, style='ent')'`, then passing `render` as an argument in the `render_template()` for the page that it will be displayed on. The text from the user is sent to this page via a POST request. Once I got familiar with this process, I passed in the `get_ents()` method from the Tutorial code but instead of returning the entities, I returned the count of entities to add more material to the blank looking results page. If I have time, I'm going to investigate counting the number of one entity which appears in the text. I think this can be achieved by using the collections library. However, that seems quite complicated as there are so many NER types. after reading the examples of test cases that demonstrate SpaCy NER performing well and in reverse, performing poorly.

05/02/2020 Training SpaCy to Recognise a new Named Entity

When creating the test cases, I discovered that SpaCy does not recognise 'Ronan' as a name and instead classifies it as an 'Organisation' which got me thinking that it may not be trained to deal with Irish names. It may be useful to learn how to train SpaCy in recognising names which it currently classifies incorrectly.

I watched a tutorial on training SpaCy and reproduced it with some alterations to the structure of the code, making it more readable for a novice user and improving my understanding of how it works. In order to be able to remove code from a project, you need to know if this will affect it. The tutorial used a very small dataset and I wanted to train something slightly bigger, given that I had a bit of extra time. I wrote my own testing and training datasets and ran with 100 shuffles and a dropout of 0.3. Those concepts will be explained on my website in the NER tutorial.

What have I learnt this week?

Identified a problem while writing the code to demonstrate SpaCy NER in my tutorial: There are Irish names which SpaCy doesn't recognise. To manage this, I learned how to train SpaCy to classify a new word into an NER category.

How to host a Flask website which allows the user to enter text, then to be able to view the named entities from pre-trained SpaCy.

How to structure a tutorial to make it more interactive and digestible. This was done by re-watching existing tutorials several times, reviewing the comments left by other users, then acting upon those comments by making my tutorial meet their needs.

How far do I feel I've come this week?

Last week, I had doubts about the feasibility of the project. However, once those doubts were set aside, I felt much more motivated to produce a tutorial covering and making use of lots of the technical skills that I've gained in the process so far. Going from minimal HTML knowledge after coming out of the previous meeting to being able to produce a website that I am proud of feels quite rewarding. I found the error messages produced by JinJa to be pleasantly specific for markup syntax errors. I came across a few challenges when uploading images to the website directly from my laptop. It turns out that some IDEs automatically create the 'templates' and 'static' folders that you are certain to use when developing a website with Flask however, Visual Studio did not do this for me and it took a bit of research because the StackOverflow solution referred to 'static' which I presumed was included in the project folder. I spent around 30-45 minutes looking at fixes for 'HTML images not loading' which all suggested I was using the wrong file path. I only realised afterwards that it was a Flask problem and not a HTML problem. Given that this was a challenge that I had to overcome, I've described my approach in the troubleshooting section of the website.

06/02/2020

I felt the meeting with John was very informative today as he gave me guidance about restructuring my diary to be less formal (make it more like a brain dump) by creating two other documents that run alongside it which will cover the technical skills I am acquiring as well as a 'plan' for the project. From this point forward, this diary will serve as a brain dump for my feelings and from Monday onward, I'll have two new documents where I'll record technical progress. However, he reiterated that the end goal of the module is unrealistic (Everest) and that the creation of something of lasting value, even if that's just making a quality tutorial about an area of NLP or making something small yet useful that others can build upon. John also brought up the point that another student is creating a tool to generate movie scripts so it might be interesting to have him generate a script and me extract information from it. I'm going to improve the appearance of my How-to guide tonight because John said that unlike the diary, it needs to look attractive to retain a user's attention. I didn't make much of an effort in this regard so tonight, it will be worth adding some more visual features. He added that although the website will be assessed, more importance is placed on having a continuously updated diary that describes why you've done something, not just what you've done.

I am becoming more accustomed to working with libraries that I've never used before. When John described the course as 'sympathetic', it made me realise that it is much more important to throw myself into the areas that I'm looking at currently than to stick to what I've done before starting this module because if I developed something linear, I wouldn't run into the learning challenges that I'm currently facing and then what's the point in the module being called 'Computer Science Challenges' if there are none? Up to now, the process nearly feels like I'm training myself as if I were an NLU AI; the "training data" was all the tutorials I read about these areas of Computing that I previously knew next to nothing about, the "training process" was writing my own projects, applying what I had learnt and now, the "testing data" is being able to explain what I've learnt to a novice user. I genuinely have seen parallels between me learning about NLP and an AI learning about something new; both of us are going to make lots of mistakes at the start and the number

of mistakes made gradually decreases as we understand the material that we have taught ourselves. Going forward, I want to read even more research articles about my project brief.

I'm worried that when the other projects start up for the other modules, that I'll find myself conflicted in terms of the amount of time to spend at each project. This module is the most demanding, however, Software Design Principles is team-based so if I dedicate time that's meant for SDP to CSC, then the mark of the whole team could suffer. Thankfully, I've done lots of work with databases in the past so there are very few new skills I'll be acquiring from that module. I would say that in the past week, my work ratio has been 60:30:10; CSC:SDP:Databases. Next week, I'm going to try to make it 55:35:10.