Hyperiondev

TASK

Natural Language Processing

Visit our website

Introduction

This is an introduction to natural language processing (NLP), which is currently one of the largest fields of research in artificial intelligence (AI). First, we will get you started with some background knowledge. Then, you will learn about how to use the spaCy library to classify a piece of text into predefined categories.

AN INTRODUCTION (USING IRON MAN)

Adapted from: Iron Man, Marvel Studios

The image above is from the movie Iron Man. In this movie, the fictional main character, Tony Stark, is a billionaire genius engineer. He creates a suit of armour that is so powerful and advanced that he becomes a superhero nicknamed 'Iron Man'.

Tony has a robot AI inside his mansion called Jarvis. He talks to this robot many times during the film and it helps him perform certain tasks. The picture above is from a scene where Tony has just spoken to Jarvis.

Let's think about this a bit. What is the most advanced technology or most crazy idea in the movie Iron Man?

- Is it the fact that Tony has built a suit made out of metal that he can fly around in?
- Is it the fact that the suit makes Tony so strong that he can shoot missiles from it, reflect bullets, and fly around?
- Is it the fact that Tony is a billionaire engineer that is smart enough to do this by himself?
- Or is it the seemingly small and insignificant fact that Tony can talk to the AI, Jarvis, and Jarvis can understand exactly what he says?

If you did not know better and had no background in AI, you might think that flying around in a suit shooting missiles is more advanced than a robot understanding the things that Tony says to it, and being able to hold a conversation in the manner of a human assistant. But you would be wrong!

The fact that Jarvis understands Tony's simple words, "Wake up, daddy's home" and can reply correctly with "Welcome home, sir" is a massive technological feat for AI. While creating a superhero suit and soaring through the skies may captivate our imagination, it pales in comparison to the vast expanse of NLP, the primary focus of AI research today. This is evident in relatively recent developments, where virtual assistants like Siri, Alexa, and Google Assistant have revolutionised technology interaction. These intelligent, NLP-powered, voice-activated systems understand user commands, answer questions, and perform tasks. Building upon their success, the field of NLP has witnessed the rise of large language models (LLMs) like ChatGPT. These models utilise deep learning techniques to generate human-like text and perform various NLP tasks such as text generation, classification, question answering, and translation. Researchers eagerly anticipate the further evolution of LLMs, with future models expected to possess even more powerful capabilities.

Continuing the path of advancements in NLP, technologies like ChatGPT are at the forefront of text similarity applications. As explained previously, these systems aim to understand and provide accurate responses to human inquiries, bringing us closer to the AI companion Jarvis from Iron Man. However, the power of ChatGPT extends beyond human communication. From an enterprise perspective, ChatGPT is poised to revolutionise multiple business sectors. Currently, its applications include copywriting, customer service, compliance, and IT, offering persuasive content, personalised engagement, regulatory support, and technical assistance. As ChatGPT and similar technologies evolve, they will open new avenues for problem-solving and decision-making across industries.

Extra resource

Read more about **ChatGPT**, including the caveats and benefits of the technology. **Try it out** for yourself on the OpenAI website.

NATURAL LANGUAGE PROCESSING

In order to start thinking about creating Jarvis in real life (i.e. a robot that can understand what we say and act on it or even just reply correctly) we need many things.

We call programs like Jarvis that converse with humans in natural language conversational agents or dialogue systems. Natural languages are languages humans use to talk to each other; 'formal languages' are programming languages like Python or Java.

Jarvis must be able to recognise words from an audio signal and generate an audio signal from a sequence of words. These tasks of speech recognition and speech synthesis require knowledge about phonetics and phonology: how words are pronounced in terms of sequences of sounds and how each of these sounds is created acoustically. Pronouncing variations of words correctly (such as plurals, contractions) requires knowledge about morphology – the way words break down into component parts that carry meanings.

What if we asked Jarvis, "How many University of KwaZulu-Natal students are in the Math130 class at the end of the day?" Jarvis needs to know something about lexical semantics – the meaning of all the words (e.g. 'class' or 'students') and compositional semantics (what exactly makes a student a 'University of KwaZulu-Natal student' and not another type of student?). What does 'end' mean when combined with 'the day'? Jarvis needs to know about the relationship of the words to each other – how does Jarvis know that 'at the end of the day' refers to a time and does not refer to something like ‘the book that is written by the author JK Rowling’? Humans know this automatically, but how can computers learn this?

How does Jarvis know that when Tony says ‘Daddy’s home’, Tony is actually talking about himself? Jarvis knows this because he says ‘Welcome home, sir’, so clearly he understood that somehow. This knowledge about the kind of actions that speakers intend by their use of sentences is pragmatic or dialogue knowledge. To summarise, Jarvis needs the following knowledge of language:

- Phonetics and Phonology – knowledge about linguistic sounds
- Morphology – knowledge of the meaningful components of words
- Syntax – knowledge of the structural relationships between words
- Semantics – knowledge of meaning
- Pragmatics – knowledge of the relationship of meaning to the goals and intentions of the speaker
- Discourse – knowledge about linguistic units larger than a single utterance

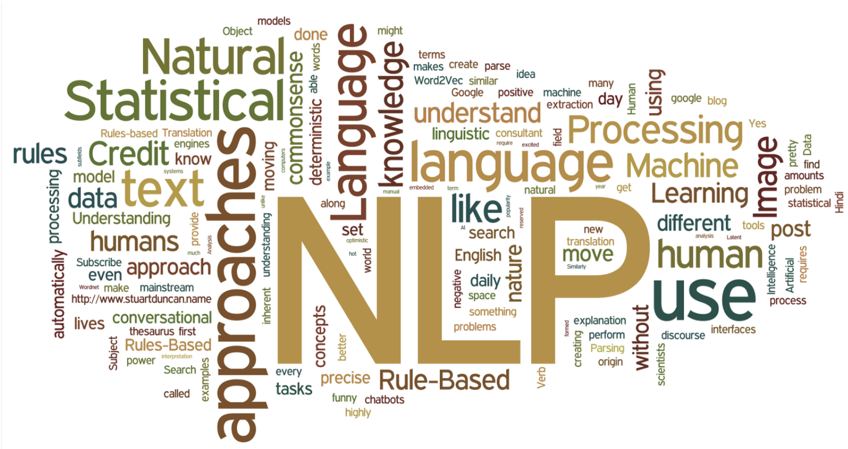

Image source: ***Deep Learning Analytics***

Ambiguity

Now, what if Tony was telling Jarvis a story about his female assistant who had annoyed him? As it happened, Tony threw a piece of paper at her and she ducked to avoid it. Describing the incident, Tony says the following sentence to Jarvis:
“I made her duck”.

This simple sentence has the following meanings. The correct meaning in Tony’s story is in bold:

- I cooked waterfowl for her
- I cooked waterfowl belonging to her
- I created the (plaster?) duck she owns
- **I caused her to quickly lower her head or body (to avoid something)**
- I waved my magic wand and turned her into a common waterfowl.

But how can we make Jarvis smart enough to know this? There are many ambiguities in this sentence because the word ‘duck’ can be a verb (move your head down) or a noun (a waterfowl), and ‘her’ can mean the woman, or can refer to the fact that the duck belongs to her.

What about hearing? Say the word ‘I’ out loud. How does Jarvis know that this word is not actually ‘eye’? What about ‘made’? It sounds just like ‘maid’! Poor Jarvis! We must use complicated models and algorithms as ways to resolve or disambiguate (remove) these ambiguities.

Siri and Alexa, as prime examples, utilise advanced techniques to effectively manage ambiguity in user queries. By analysing the conversation's context, including prior interactions and user preferences, they can deduce the most probable interpretation of ambiguous queries. Moreover, they leverage machine learning algorithms to continually enhance their comprehension and disambiguation skills by learning from vast datasets. Whenever ambiguity arises, Siri and Alexa may seek clarification through follow-up questions, allowing them to gather more details and deliver precise responses. This interactive approach facilitates their understanding refinement, ensuring users receive the desired information or assistance with greater accuracy.

Part of speech tagging

Deciding whether 'duck' is a verb or a noun is known as part of speech (POS) tagging. Verbs and nouns are different 'parts of speech' and we 'tag' a word in a sentence by assigning it one part of speech that we think is correct for the context or sentence it has been used in.

In a phrase like "The old man and the boat", the task of tagging the correct parts of speech may involve the following:

***The*:** tag as 'determiner'
***old*:** tag as 'adjective'
***man*:** tag as 'noun' (or verb?)'
***the*:** tag as 'determiner'
***boat*:** tag as 'noun'

Sometimes, probabilities are used to decide this. For example, the probability is higher that the word 'man' above means the noun 'an older male person' than the verb 'put someone there' (e.g. "The enemy is here! Man the cannons!").

As you can see, NLP is a huge field and extremely important to AI. NLP also includes the study of things like machine translation which is the same as the technology behind Google Translate – automatically translating between two languages. Google Search uses NLP techniques in order to understand your searches faster, and this is what makes Google Search more accurate, more reliable, and faster than any other search method on the Internet. NLP is also used in Gmail in order to identify spam mail and delete it.

NLP research is one of the main reasons Google is so successful. The probabilistic, machine learning techniques that Google applies to NLP can be applied to other AI tasks such as creating driverless cars.

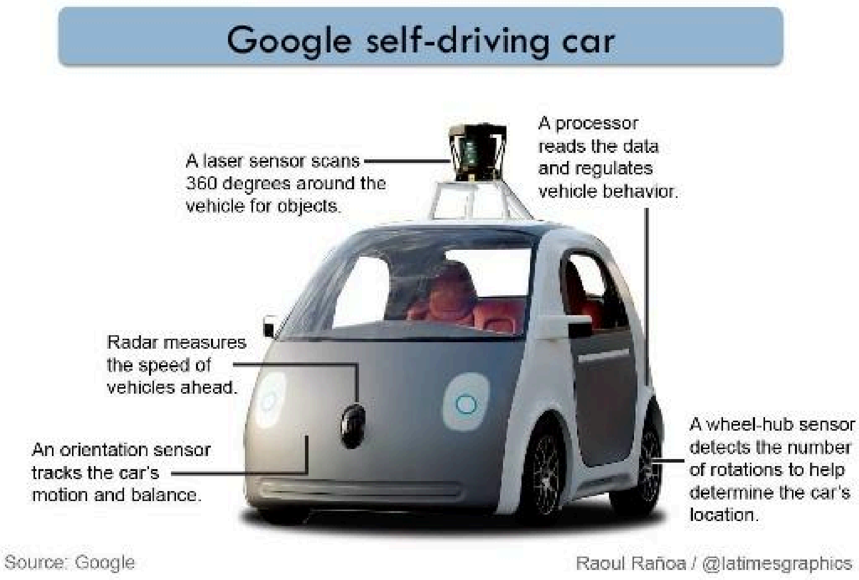

*Image source: **[(Rañoa, 2014)]***

We hope you can see how many fields in AI have to do with probability. This is because it is the only way we can deal with ambiguity, in order to enable robots/AI programs to make the best decisions!

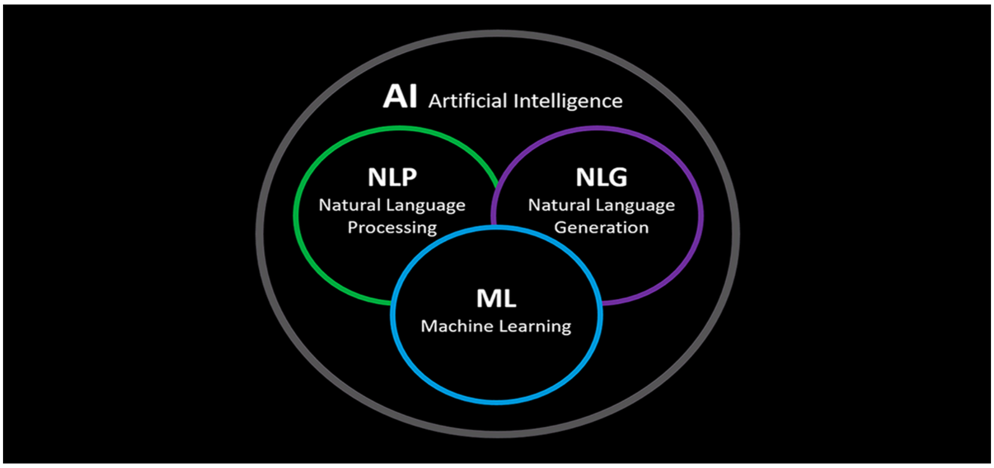

*Image source: **Conversica***

MACHINE LEARNING

NLP is a big part of AI because a large portion of NLP has to do with training machine learning models to identify patterns and use probabilities to make informed decisions.

Facebook uses machine learning to try to recommend friends to you; Amazon uses it to recommend items to buy; Google Image Search uses machine learning techniques to identify patterns in pixels to try to find similar images; and Google has designed driverless cars to act according to their environment by integrating many different aspects of machine learning. The list is endless.

Solving POS tagging

One example is to solve the problem of POS tagging as explained earlier. We can give a program a big set of tagged words (training data) and then give it a new sentence where it must try to tag words using the information learnt from the training data.

The University of Pennsylvania had the first NLP research program to ever take a large corpus (body) of words and tag each and every one by hand to create a **treebank**. A program was then activated and slowly learnt how to tag words and the probability that a certain word appeared with a certain tag in a certain context. Ever since then, AI runs on huge sets of training data. The larger the dataset, the more accurate we can train an AI program to be. Training sets utilised for ChatGPT and other LLMs generally encompass substantial volumes of text data. To illustrate, the training set for ChatGPT-3 was an extensive collection of 570 GB of text, drawn from diverse sources such as online content, books, and articles. This vast corpus facilitates the model's acquisition of patterns, syntax, and semantic relationships inherent in human language, empowering it to generate responses that are coherent and contextually appropriate.

In POS tagging we try to tag words with the correct POS tag so that we can then parse the sentence correctly. **Parsing** is the formal term for ‘putting a sentence together in the right way’ so that it can be ‘understood’. We will talk more about this later.

Text classification

We also use machine learning to try to classify texts. For example, Gmail classifies emails as either ‘spam’ or ‘not spam’. It uses machine learning by having trained an AI program on a set of already ‘classified’ emails (examples of non-spam and spam emails). Then when the AI program sees a new incoming email, it can use its prior knowledge or ‘training’ to classify the new email correctly.

This task will have an example of how we can use machine learning to get a program to identify positive or negative tweets – similar to the problem of identifying spam mail through machine learning - but first, we need to start with SpaCy.

SPACY

SpaCy is a Python NLP library specifically designed with the goal of being a useful library for implementing production-ready systems. It is particularly fast and intuitive, making it a top contender for beginners in NLP. Before doing anything, you need to have spaCy installed, as well as its English language model.

Type the following commands in the command line to install spaCy:

```
pip3 install spacy
```

Please **keep in mind** that Python 2 is now **deprecated**. If your system **does not** recognise '`python`' in the next step or if you have a macOS, please use '`python3`'.

To confirm that spaCy is installed correctly, follow these steps:

1. Open your terminal or command prompt and start the Python console by typing:

   ```
   python
   ```

2. Once inside the Python console, type the following command to check if spaCy can be imported without errors:

   ```
   import spacy
   ```

If you receive no error, this means that spaCy was installed correctly!

If you encounter an error, please use Google Colab to complete all your NLP tasks that require SpaCy. Refer to the Google Colab section in the **Installation, Sharing, and Collaboration** additional reading you used at the start of your bootcamp.

To exit the Python console, use the following command:

```
quit()
```

Now let's talk about language models.

Language models

SpaCy is not very useful without at least one language model. The model allows you to process different languages. SpaCy currently has 23 language models which include French, Dutch, Spanish, and of course English. You can install more than one model or even install a multi-language model all at once. SpaCy's models can be installed as Python packages. This means that they are a component of your application, just like any other module. Models can be installed from a download URL or a local directory, manually or via pip. Their data can be located anywhere on your file system.

To download the English model, type the command below in your terminal.

```
python -m spacy download en_core_web_sm
```

To test our newest model, we will need to try using the model. Import spaCy as below:

```
import spacy
```

Now we will load the model and assign it to a variable.

```
nlp = spacy.load('en_core_web_sm')
```

The input to NLP will be a simple stream of Unicode characters (typically UTF-8). Basic processing will be required to convert this character stream into a sequence of lexical items (words, phrases, and syntactic markers) which can then be used to better understand the content.

For spaCy, you can do this by passing the string through the language model you imported at the beginning of the script. Remember we named our model `nlp` so to process our string in preparation for spaCy manipulation, we use the code below:

```
doc = nlp("this is a test sentence")
print([(w.text, w.pos_) for w in doc])
```

Output:

```
[('this', 'PRON'), ('is', 'AUX'), ('a', 'DET'), ('test', 'NOUN'),
('sentence', 'NOUN')]
```

Like other NLP libraries, spaCy uses hash values instead of actual text strings to save space and speed up processing. To find the actual text associated with a particular attribute in spaCy, you need to add an underscore (_) to the end of the feature's name.

Now that we are able to process a string, we can do more complicated text processing such as tokenisation, lemmatisation, and named entity recognition. Let's take a closer look at these.

Tokenisation

Tokenisation is a foundational step in many NLP tasks. Tokenising text is the process of splitting a piece of text into words, symbols, punctuation, spaces, and other elements, thereby creating "tokens".

Lemmatisation

A related task to tokenisation is lemmatisation. Lemmatisation is the process of reducing a word to its base form, its mother word if you like. Different uses of a word often have the same root meaning. For example, practise, practised and practising all essentially refer to the same thing. It is often desirable to standardise words with similar meanings to their base form.

Named entity recognition

Named entity recognition is the process of classifying named entities found in a text into predefined categories, such as persons, places, organisations, dates, etc. SpaCy uses a statistical model to classify a broad range of entities, including persons, events, works-of-art, nationalities and religions. See the **spaCy documentation** for more information.

Examples of tokenisation, lemmatisation, and named entity recognition are in the **nlp_example.py** file for this task, so open it up and take a look.

Instructions

First, read **nlp_example.py**, and run it to check that your spaCy installation is working. Remember to use Google Colab if you encounter any problems. If you still have problems running **nlp_example.py**, please log a query on your dashboard.

Feel free to write and run your own example code before doing this task to become more comfortable with the topic.

Practical task

In this task, you will get to grips with using the basic functionalities of spaCy.

1. Read the introduction about **garden path sentences** and study a few of the examples provided on Wikipedia.
2. Create a new Python file called **garden.py** and save it to your folder for this task.
3. Find at least two garden path sentences from the web or think up your own.
4. Store the sentences you have identified or created in a list called `gardenpathSentences`
5. Add the following sentences to your list:
    - Mary gave the child a Band-Aid.
    - That Jill is never here hurts.
    - The cotton clothing is made of grows in Mississippi.
6. Tokenise each sentence in the list and perform **named entity recognition**.
7. Examine how spaCy has categorised each sentence. Then, use `spacy.explain` to look up and print the meaning of entities that you don't understand. For example: `print(spacy.explain("FAC"))`
8. At the bottom of your file, write a comment about **two entities** that you looked up. For each entity answer the following questions:
    - What was the entity and its explanation that you looked up?
    - Did the entity make sense in terms of the word associated with it?

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.

ACKNOWLEDGEMENTS

We worked with the University of Edinburgh, who created **NLTK** and contributed to **spaCy**, to create this task. This task uses content adapted with permission from the University of Edinburgh's Informatics department, one of the leading NLP research departments in the world. You can see their related course content at **Informatics 2A: Processing Formal and Natural Languages**.

REFERENCES

Biggs, C. (2022). 20 Grammatically-Correct Sentences You Won't Read Right The First Time. Apartment Therapy. **https://www.apartmenttherapy.com/garden-sentences-262915**

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Linguistic features · SPACY usage documentation. (n.d.). Linguistic Features. https://spacy.io/usage/linguistic-features