**Graduate Scheme Module 7 (NLP)**

**Intro to NLP**

What is it?

* A branch of AI that gives computers the ability to interpret, manipulate and comprehend human language.
* Incorporates computer science, linguistics and ML

Key challenges

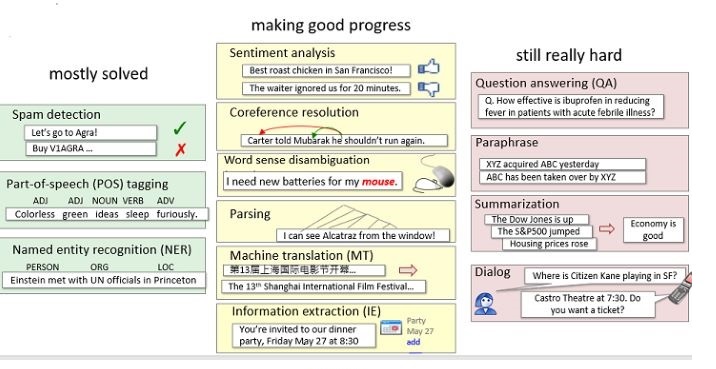
* Ambiguity – multiple possible interpretations of language.
* Synonyms and Homonyms – similarities and exactness in languages leading to different resolution.
* Irony and Sarcasm
* Slang

Common applications

* Email filters
* Smart Assistants
* Search Results
* Predictive Text
* Language Translation
* Text Analytics
* Spell check
* Voice to text
* Chatbots

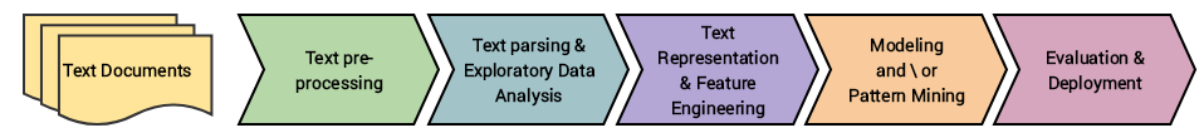
Applications of NLP (with challenges)

* Machine Translation Technologies
  + Challenge: preserve the meaning of the sentence from one language to the other
* Search Engines eg. Google
  + Challenge: recognize natural language questions, extract the meaning of the question and give an answer
* Text Classification eg. Spam Filters
  + Challenge: Overcome False Negatives and False Positives ie. sending to spam folder non-spam emails and vice-versa
* Sentiment Analysis eg. identify sentiments for a product
  + Challenge: understanding sarcasm and ironic comments
* Topic Modelling: method for discovering the abstract topics in a document collection
  + Challenge: using a robust algorithm, sacrifice speed over accuracy?
* Transcription of speech (turning spoken language into written languages)
  + Challenge: dealing with looser grammar
* Question Answering: build systems that automatically answer questions posed by humans in a natural language.
  + Challenge: understanding the infinitely varied forms of expression



Work Flow in NLP (CRISP-DM model)

1. Have a dataset/corpus
2. Text preprocessing (Data Cleaning)
3. Exploratory Analysis and Data Transformation
4. Split the dataset (Data Scientists may prefer to do the exploratory analysis after they split the Dataset)
5. Identify the technique that is most suitable for your dataset and what you may think can take out of it. This will impact the data preprocessing undertaken
6. Explore different features of the model on the Validate Dataset (Tuning)
7. Test the accuracy and the robustness of your model
8. Communicate your results
9. Make a prediction, if it is possible



Packages

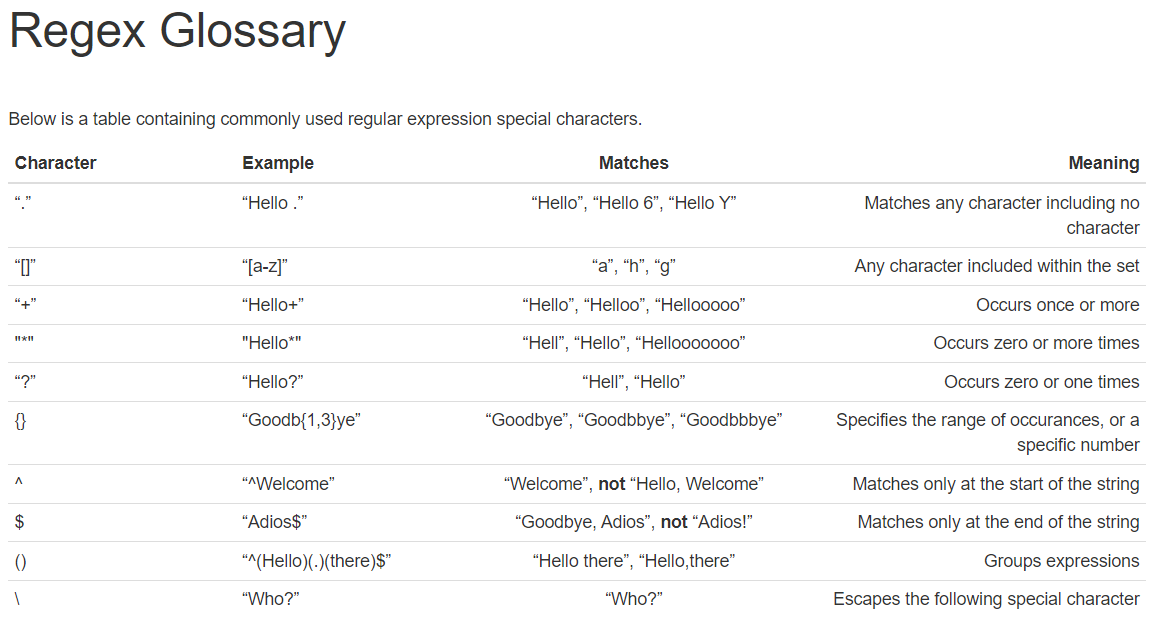
* nltk
  + nltk can be used for teaching and understanding but it is slow.
  + Good general tool on a comprehensive range of nlp tasks.
  + nltk was originally a tool for academic research.
  + A high degree of customisation is achievable.
* stanza
  + Developed by the University of Stanford.
  + A collection of tools that can be applied to many human languages.
  + Touted as having state of the art parsers.
  + Not available on ONS machines and possibly some other restricted devices
* spaCy (this is best, even though we’ve used nltk in the examples for this module)
  + Considered to be production-oriented.
  + spaCy is considered fast and robust.
  + Less potential to customise than other NLP libraries.
  + Sold as having robust tools for large scale language processing.

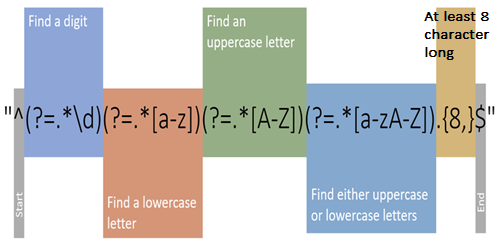
Pre-Processing

* Structured processing where order matters (creates meaning)
* Mixture of common string manipulations and unique NLP methods.
* Key steps
  + Tokenization
    - Often the first step of the cleaning pipeline.
    - Process of splitting text into individual tokens.
    - A token is a meaningful unit of text, often words or sentences.
    - Essential, as it separates linguistic elements like punctuation.
    - We can then apply further cleaning steps to address noise.
  + Remove noise
    - Noise is multi-faceted in text data.
    - Our language is so complex, with many connectives, clauses, context clues, punctuation and so on.
    - How we deal with these depends on the problem to solve.
  + Noise reduction strategies
    - Lower-casing the text
    - Remove stop words
      * A very important step when cleaning text is to remove stop words from the dataset.
      * These are common words any language such as “the”, “and” “it”, “is”.
      * There are many dictionaries of these that require careful considerations on which to use.
    - Remove punctuation/numbers/whitespace
    - Fixing mis-spelling
    - Stemming/Lemmatization
      * Stemming reduces words to their root form (known as a stem) even if this stem has no meaning.
        + Crude but requires minimal compute power
        + Appropriate to use when the meaning of words is less important
      * Lemmatization reduces words to their root dictionary form (known as a lemma) a.k.a, the root word has meaning and we removing suffixes only.

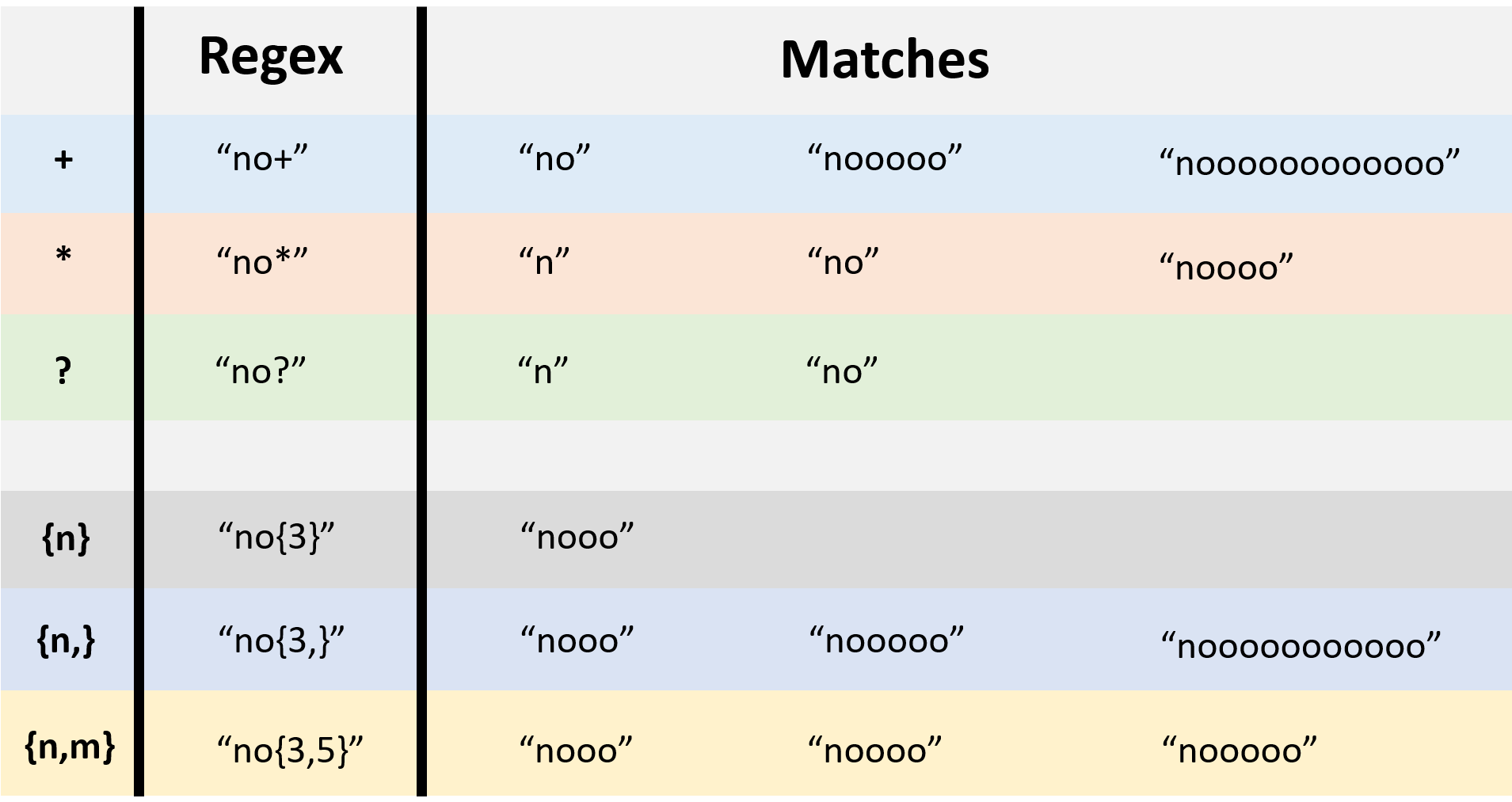
Regex (Regular Expressions) – see [regex101.com](http://www.regex101.com)

* Helps us to create patterns that can be searched within our text
* Useful to carry out general searches to help us clean data more easily and identify specific patterns
* Examples of some harder tasks to achieve are checking:
  + Whether the text is a valid email
  + If a password meets some security requirement
  + Whether a phone number follows a valid format
  + If a mistake has been made when inputting some data manually
* Eg. # Password manager that identifies whether a password is 8 digits/letters long and also contains 1 special character)
  + pattern = re.compile(r'^(?=.\*[!@#$%^&\*()\_\-+=<>?])[a-zA-Z0-9!@#$%^&\*()\_\-+=<>?]{8}$')





* Quantifiers
  + Can be greedy or lazy ([Greedy and lazy quantifiers (javascript.info)](https://javascript.info/regexp-greedy-and-lazy)
    - Greedy looks for the largest example that fits the criteria
    - Lazy looks for the smallest example that fits the criteria



* Use the Quick Reference section of regex101.com to identify possible options (or use Chat GPT to write regex expressions for you).
  + Anchors: Anchors do not match any characters themselves but instead specify positions in the text, such as the start ^ or end $ of a line.
  + Meta Sequences: Meta sequences are special expressions in regex that signify broader categories of characters, like \d for any digit or \s for any whitespace character.
  + Quantifiers: Quantifiers specify how many instances of a character, group, or character class must be present in the target sequence for a match to occur, such as \* (zero or more), + (one or more), and ? (zero or one).
  + Group Constructs: Group constructs organize patterns into subexpressions that can be captured, referenced, or applied modifiers to as a single unit, with plain () for capturing and (?:) for non-capturing groups.
  + Character Classes: Character classes match any one of a specified set of characters, such as [a-z] for any lowercase letter and [^0-9] for any character that is not a digit.
  + Flags/Modifiers: Flags or modifiers alter how the regex engine interprets the pattern, like i for case-insensitive matching or g for global search which finds all matches.
  + Substitution: Substitution involves replacing matches of the regex pattern in the text with a replacement string, often using captured groups to dynamically construct the replacement text.

Text Mining

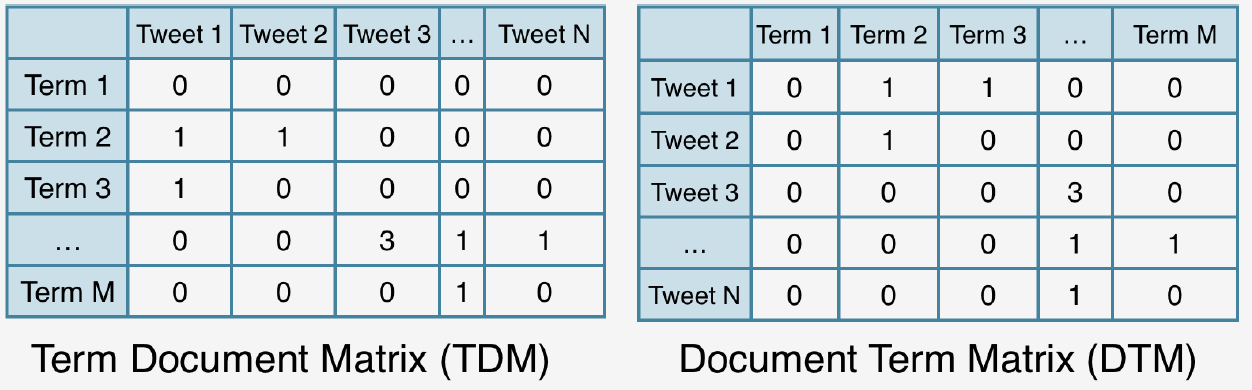
* Can be done after pre-processing
* We attempt to separate valuable keywords from a mass of other words and use them to identify meaningful patterns or make predictions.
* Following this we can apply analytical techniques such as Cluster Analysis and Classification.

Key approaches to Text Mining

* Bag of Words
* Sentiment Analysis
* Topic Modelling (Latent Dirichlet Allocation)

Bag of Words

* It is called a “bag” of words, because order and structure of words are ignored. It aims to answer whether words occur (and how often) rather than where they occur.
* This approach is usually used when we have a collection of documents, each of which are bodies of text.
* For bag of words, we create a special object known as a corpus, or a collection of documents.
* From a corpus, we create special data structures which are transposes of each other:
  + Term-Document Matrix
  + Document-Term Matrix



* Advantages
  + Simple to understand and implement using popular packages.
  + Highly flexible customisation of the corpora created.
  + Bag of Words is often the baseline model or simpler model to start a linguistic modelling problem.
* Issues
  + Vocabulary: Needs to be managed so the size doesn’t get out of hand.
  + Meaning: We lost context and sentiment behind the words, which can provide useful information.
    - Sparsity: Sparse representations are harder to model and draw information from.
    - In smaller corpora (eg. 2 documents), we get numerous binary vectors.
    - Sometimes this can lead to the issue of sparsity.
    - Since each document can have hundreds of thousands of terms, we expect the matrices to be incredibly large.
    - Inherently, we get sparse vectors, those with a high percentage of 0s.
    - These are difficult for traditional algorithms to model efficiently.

Sentiment Analysis

* Teaching a machine to understand how context can affect tone is even more difficult.
* Sentiment Analysis is the technique that attempts to do this.
* Sentiment Analysis techniques are usually employed to identify three things:
  + Polarity of the expression
    - Polarity: Is the opinion positive or negative (some include neutrality).
  + Subject of the expression
    - Subject: The thing that is being talked about.
  + Opinion Holder of the expression
    - Opinion Holder: The person, or entity that expresses the opinion.
* Considerations
  + Efficient does not mean perfect when determining sentiment.
  + We need to construct specific elements for linguistic devices for more accurate sentiment scanning, such as:
    - Context based tagging
    - N-grams (pairs, trios etc of words)
  + We end up with very flexible models, but when do we stop? No easy answer.
* Numerous approaches
  + Knowledge-Based: Categorize text based on unambiguous “affect words” like love, like, hate and so on.
    - Lexicon methods
      * This knowledge-based technique is an incredibly popular baseline approach.
      * It uses a lexicon of pre-defined positive and negative words and matches these to the text.
      * A sentiment score is calculated, averaging out the numbers of positive and negative matches.
        + This is the score that determines the sentiment classification of the text. It is calculated as:
        + sum(positives) – sum(negatives)
        + Once calculated, threshold checks are made and a classification is provided (pos, neg, neut etc).
      * Issues
        + Lexicons are largely crowd-sourced and validated by volunteers, leading to a possible lack of credible peer review.
        + However, there are so many neutral words in language that some are likely missing.
        + Lexicons don’t consider qualifiers before the word, which can invert the sentiment, for example “no good”.
  + Statistical Methods: Model detects the sentiment holder as well as the subject in the sentence.
  + Hybrid Approaches: A combination of the two with additional linguistic techniques to pick up semantics.
* Considerations
  + We may hesitate to apply these dictionaries to historical documents, since the language is so different.
  + Applying this on large chunks of text usually averages out the sentiment, so tokenisation becomes even more paramount here!

Topic Modelling

* Unsupervised NLP technique, gives us an idea of what text is about quickly.
* A topic is a label or collection of words that often occur together.
* Latent Dirichlet Allocation
  + Very popular algorithm that takes a Document Term Matrix (DTM) as it’s input.
  + Outputs matrices, one with the prevalence of topics in documents, and the other with the probabilities of words belonging to the topics.
  + As it is unsupervised, we provide a value of K topics to look for beforehand, usually informed with knowledge of the data.
  + How it works
    - Every document is a mixture of topics. For example, in 2 topic models we could say “Document 1 is 90% topic A and 10% topic B” etc.
    - Every topic is a mixture of words. For example, we could imagine a two-topic model for news, one for “politics” and another “entertainment” and so on.
    - This is essentially probabilistic clustering, but a little fuzzy as documents can be associated with more than one topic.
* Considerations
  + We must ensure stop words are dealt with beforehand, as they will likely make up most of the corpus.
  + LDA estimates two probabilities simultaneously, which leads to long runtimes.
  + When documents are often about the same subject matter, certain words become redundant as the corpus increases in size.