Topics covered:

Day 1:

* High level applications of NLP
* Theory: NLP tasks like stemming, lemmatization, stopword removal
* Theory: Text embeddings/ vectorization concept, tfidf and one-hot encoding (or Bag of Words) vector calculation.

Day 2:

* Code implementation of basic text analysis along with tokenization, stemming, lemmatization, stopword removal etc. using NLTK

Day 3:

* Theory: Intro to NLP tasks like topic modeling, text clustering, text classification, similarity comparison.
* Theory: Average workflow of ML/ DL model for NLP tasks.
* Theory: Concept of word2vec  
  <https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469>
* Code implementation: Vectorization using tfidf and spacy.  
  Steps for tfidf:
  + Fit a tfidf vectorizer using a corpus of text (the vectorizer learns the different words or tokens from the corpus)  
    *tfidfvectorizer.fit(corpus)*
  + Transform new sentences using the created vectorizer to generate embeddings.  
    *tfidf\_embedding = tfidfvectorizer.transform( [‘I have a car’] ).toarray()*

Steps for Spacy:

* Import spacy package to python and import the appropriate spacy model for English language (we used the sm model)
* Pass the sentences to spacy model and extract the vectors.  
  *sentence\_vectors=[]*

***for*** *doc* ***in*** *nlp.pipe( [‘I own a car’, ‘I am a programmer’] ):*

*sentence\_vectors.append(doc.vector)*

* Code implementation: Cosine similarity of two sentences (or paragraphs) after converting them to vectors.
* Code implementation: Using generated embeddings to train a machine learning model for text classification.

Pre-requisites:

* What is machine learning?
* What is supervised and unsupervised machine learning?
* Basic knowledge of any ML model
  + Split dataset to test and train
  + Fitting the model on the train dataset (it’d be better if one understands the math behind what happens when a mathematical model is fit- we used a Naïve Bayes model)
  + Testing the model on the test dataset using performance metrics like accuracy

Day 4:

* Introduction to Spacy
* Code: Tasks Spacy can perform
  + Tokenizer, stemming, lemmatization
  + Named Entity Recognition
  + Parts of Speech Tagging (POS) + Dependency parsing
  + Creation of embeddings using Spacy
  + Spacy pipeline
  + Custom Named Entity Recognition models
    - Patterns
    - Training existing NER models or creating new ones

Day 5

Challenge: 3 groups of text are given.

From text group 1,

1. identify entities (person, place, thing, location)
2. Categorize them into ‘living entity’, ‘person’ or ‘object’.

From text group 2,

1. Identify main words or topics mentioned in the corpus.

From text group 3,

1. Analyse the sentiment of each of the sentences
2. Use a model to extract the answer to a question from the corpus.

Hints:

* Leverage different tasks Spacy can perform along with some tasks transformers can do. Take a look at customizing some of the existing tasks too.
* Better quality vectors can produce better output for the tasks- so try experimenting with different models available under transformers

The code should be shared for evaluation along with the final output- winners will be based on the methods used, as well as the number of entities identified correctly identified.

Other useful links:

1. Topic modeling: <https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>
2. Spacy tutorial: <https://www.machinelearningplus.com/spacy-tutorial-nlp/>
3. Text classification using Spacy: <https://www.machinelearningplus.com/nlp/custom-text-classification-spacy/>
4. Training custom NER model using Spacy: <https://www.machinelearningplus.com/nlp/training-custom-ner-model-in-spacy/>