# SUBMISSION 2 LEARNING LOG

**Submission 2 Challenge**: The goal of this submission is to demonstrate that I have systematically researched the fields of Computer Vision, Linguistics and Computational Linguistics to survey what advances have been made in respect to representing the semantics behind text and also the properties of shapes within images, then using the information I’ve gathered to propose a vision for what my final submission will look like.

## Theory-based Challenges

* Research **graph-based approaches** to semantic representation.
* Consolidate understanding of **vector-based representations**.
* **Compare** and **contrast** both approaches; deduce which is preferred.

## Practical Challenges (Improve my NLP General Knowledge)

* Build **a Password classifier**, then update my how-to guide with steps for making this using Streamlit.
* Build an **Automated** **Text Summariser** using Gensim and/or Sumy, then **update my how-to guide** with steps to reproduce it.

**WEEK 6**

*Computer Vision:*

Making a computer understand a digital image or video with the intention of automating what the human visual system can do. An application of this would be supporting a driver or pilot in hazard detection. It could also be used to look for forest fires. The Mars Rover (NASA) is an example of Computer Vision being used for Space exploration – recognising and identifying pre-learned objects.

*Neural Network:*

A concept of learning from past experiences to update understanding. In AI, a neural network is trained with a range of inputs and given their expected outputs. They then compute the input, compare their output with the expected output and learn to improve the results if required.

*Morphology*:

The study of words, how they are formed and how they relate to other words in the same language.

Word-Vectors:

A word is described using a set of decimal numbers and a key of characteristics relate to all words in that word group.

*Feed-forward Neural Network:*

A unidirectional neural network; nodes do not form a cycle. It has three node types: Input, Hidden and Output.

*Syntactic Ambiguity:*

A sentence with a referent (‘it’) which also has two nouns in it where the referent could be either noun: “The ball went through the table because it was made of *metal/glass*”. An AI struggles immensely with this kind of sentence because to classify the referent as either ball or table, it needs to understand material properties.

*The Winograd Schema:*

If a machine can determine an intended referent based on clues from the context of a sentence (especially a syntactically ambiguous one), then it is using reasoning to parse the sentence. Terry Winograd proposed this concept in 1972. A team of researchers at the University of New York have devised a collection of 150 sentences which follow the Winograd Schema; more or less the Turing Test of NLP. For example, *Lily spoke to Donna, breaking her [concentration/silence]. Whose [concentration/silence]?*

<https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WS.html>

*GPT2*:

A text-generating model built by OpenAI. It can accept a prompt and then finish the passage off for you. It has 1.5 billion parameters and outperforms models trained on domain specific datasets such as Wikipedia. You can generate a whole article based on small input sentences!

*Language Modelling:*

An NLP task to probabilistically predict the next word or character of a document.

## **WEEK 7**

This week, my learning log is divided into theory-based research by reading relevant papers and practical skills acquired through learning about new Python libraries and maths. I was researching how the LexRank and TextRank algorithms work and did not understand what an Eisenvector was initially, so I decided to investigate how Eisenvectors and Eisenvalues work as well.

I’ve decided to focus on graph-based representations of semantics this week because last week, all my research that is of value to the end goal of the topic was on vector-based represenations. My plan for this week describes how I found these papers. This document showcases what I learnt from reading them.

### Streamlit

Open source ML development framework for Python which lets me make a widget-based interactive application which can be launched locally. Widgets include text entry areas, buttons, subheaders. I chose to use this framework because of easy hosting/nice aesthetics. To run a Streamlit application, you activate the virtual environment which contains Streamlit, then type ‘streamlit run <filepath>.py’.

### Automatic Text Summarisers

The ultimate goal of Automatic Text Summarisers is to generate abstracts for any document which address all its key points. There are two main types; extractive and abstractive. Extractive summarisers return a summary which only includes phrases from the source text. Abstractive summarisers may generate their own description of the source text. Each has pros and cons.

Pre-2017, abstractive summarisers coped well at summarising short documents but tended to produce repetitive and incoherent phrases. In 2017, Salesforce researchers published their deep learning reinforced model for abstractive summarisation. Its paper has been cited over 400 times, making it reliable.

### Exposure Bias, Availability Bias and Confirmation Bias – why do these have implications on representing the semantics of text?

Supervised learning runs the risk of causing exposure bias; the idea that the machine expects to be trained on the truth at each stage. Availability bias causes a decision maker to favour view A over view B because they have more information about view A, even if view B equally or more valid but the information is harder to obtain. Confirmation bias is when a decision maker favours view A over view B because view A proves a hypothesis/belief that they have held. Unless a machine is trained on an unbiased data set, the understanding of the training data may be inherently biased which will lead to the machine drawing unfair conclusions about a subject, then making a decision which is equally unfair. This is best described by Dr. Krishna Gummadi who spoke at the 2019 BCS Turing Talk in London on the importance of training unbiased AI in relation to Latent Text Representations. He identified a key issue with the existing Word2Vec; when you run “A is to X as B is to what?” tests on the vectors, sexist results like “Man is to computer as woman is to housekeeper” appeared in abundance. He also stated that the biases in these multidimensional representations are very difficult for humans to spot.

### LexRank vs TextRank

These ranking algorithms were developed by different teams at the same time and have a similar purpose. One key distinction between them is that TextRank was made for single-document summarisation whereas LexRank can also perform multi-document summarization.

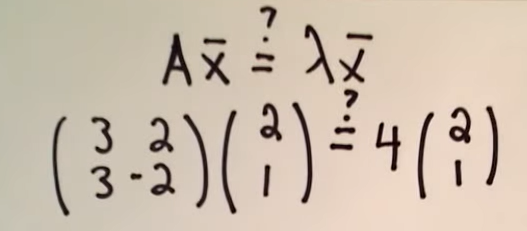
### Graph-based Survey Paper – 2013, Hansung University

This paper describes the format of a graph-based representation: nodes and edges where a node an be homogenous (represents one component of a sentence) or heterogenous (represents two or more components of a sentence). Unlike a vector-based model, the order of appearance of words can be considered. Nodes can also either be weighted or unweighted. These weights indicate importance within the graph.

### Eigenvectors and Eigenvalues

An Eigenvector is a non-zero vector which gives a scalar multiple of itself when an operation is performed on it. It does not change direction during a transformation. <http://setosa.io/ev/eigenvectors-and-eigenvalues/>

The website I’ve linked offers an interactive demonstration of the concept in terms of Adult and Child Amoeba. This fortifies my belief that interactive how-to guides are most effective. The best way of explaining Eigenvalues and Eigenvectors is through a simple Matrix Example.



In this question, A is the matrix, x bar is the Eisenvector and lambda is the Eisenvalue. The objective is to find an Eisenvalue such that Eisenvector \* Matrix = Eisenvector \* Eisenvalue.

Do Matrix multiplication:

(3\*2)+(2\*1) = 8

(3\*2)\*(-2\*1) = 4

Find a multiplier which changes the RHS into the simplified LHS. In this case, 4. I’ve only looked at the basics of Eisenvectors and intend to revisit these in the future because of how useful they are in understanding how vector-based models are created.

### Comparison between Vector-based and Graph-based representations of the meaning of text.

* According to ‘Distributed Representations of Sentences and Documents’, lots of ML algorithms require the input to be a fixed length feature vector, so word order and semantics are omitted. This paper proposes ‘Paragraph Vector’, an unsupervised ML algorithm which gets trained to predict missing words in a document. This has been shown to outperform conventional ‘bag-of-words’ vectors. The main use of ‘Paragraph Vector’ is to predict words in a paragraph which have been hidden. More precisely, we concatenate the paragraph vector with several word vectors from a paragraph and predict the following word in the given context. Both word vectors and paragraph vectors are trained by Rumelhart’s stochastic gradient descent and backpropagation.

## Generative Adversarial Networks (GAN)

* In computer vision, there are GANs which have a generator and a discriminator. The role of the discriminator is to recognise an object and state how certain it is that it is correct (1 being “THIS IS THE OBJECT” and 0 being “THIS IS NOT THE OBJECT”. The role of the generator is to make new images of that object. In a test, the discriminator can be shown a mix of images of a given object (some real, some generated) and at the beginning of the generator training, the discriminator should be able to easily identify the generated images because they’ll simply not be like the real object. However, as the generator improves, the certainties from the discriminator should move closer to 0.5 (“UNSURE IF THIS IS OR ISN’T THE OBJECT”). Once the discriminator reaches 0.5 consistently, we can discard it because we have made a generator that can make images which are very similar to the real object. When I have time, I’m going to see if anything like that has been done for text-based resources. It would be very interesting if a discrimination tool could be/has been designed to recognise automatically generated text. As far as I’m aware, this task is similar to our end-goal of formalising text in that right now, there is a computer vision version but no NLU version.