**Interpretable Word Embeddings for Explainable NLP**

**About me:**

I am Piyawat Lertivittayakumjorn, you can call me ‘Peter’, and I am from Thailand. Currently, I am a PhD student in Computing at Imperial College London under the supervision of Professor Francesca Toni. You can find out more about me on my personal page –insert\_link--.

**My Research:**

I am currently researching on interpretable word embeddings that would be used for an explainable natural language processing (NLP) program. Interpretable word embeddings are representations of words in way that a machine learning algorithm can process but is still able to be understood by a human being. These words would still be associated by values, but these values represent the attributes of the words such as its grammar and context it is used in. An NLP is a program that can process natural languages humans use and run tasks given to process the language such as translation or word comprehension. However, with current implementations of NLP we cannot retrieve any information on how the program processes its tasks due to the nature of machine learning and deep learning. Thus, with the use of interpretable NLP we can pull this information that can justify the program’s actions and fix it if the results are not desirable. This data can also be used to create rules that can be implemented for similar tasks.

**Word Embeddings:**

Word embeddings is a representation of words and their contexts. These words are represented as vectors with the differing dimensions expressing different contexts of the words. These vectors are ordered lists of numbers that can represent directions and magnitudes in space. This is needed as most Machine Learning algorithms and Deep Learning Architecture are unable to receive strings as input thus word embeddings replace these words as numbers that these programs that take as inputs and process.

These dimensions are made through the use of neural networks with methods such as Word2Vec so in general these dimensions are not known. Thus, we do not know how the words are similar or different to one another. These vectors combine to create a whole matrix that represents the word embedding of this vocabulary.

From these word embeddings we can calculate the similarities of the words using cosine similarities that takes the dot product of the two words that are being compared. The values returned are between 0 to 1 and the more related two words are the value calculated will be closer to 1. For example, the words ‘cat’ and ‘dog’ would return numbers closer to 1 whereas ‘cat’ and ‘sun’ would be closer to 0.

In the diagram above ‘dog’, ‘cat’ and ‘sun’ are represented as vectors on a 2D Cartesian plane. The cosine value of α is close to 1 while the value of β is close to 0, thus representing the similarity or difference of the words.

Further reading on the various types of word embeddings and a deeper explanation on the implementations of it can be found on this link <https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/>

**Interpretable Word Embeddings:**