**Interpretable Word Embeddings for Explainable NLP**

Interpretable Word Embeddings and Explainable NLP

This website aims to explore the topic of interpretable word embeddings and the use of it for explainable natural language processing (NLP)

* Interpretable word embeddings are representations of words that a machine can understand but can still be understood by a human.
* NLP or natural language processing is a program that can process natural languages humans use and run tasks given to process the language such as translation or word comprehension.

A branch of NLP is explainable NLP that allows for information justifying the program’s actions to be retrieved. This information would then be used to fix the program if the results are not desirable or to create rules that can be implemented for similar tasks.

This requires the use of interpretable word embeddings which are special forms of web embeddings that allow transparency of normal word embeddings by attaching context to the words such as grammar and use cases.

In the other pages, these subtopics are further explored and explained.

Piyawat Lertivittayakumjorn:

Piyawat, ‘Peter’, is a PhD student in Computing at Imperial College London under the supervision of Professor Francesca Toni. His website can be found here.

**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 can 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, it is not known how the words are similar or differ 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.

Example:

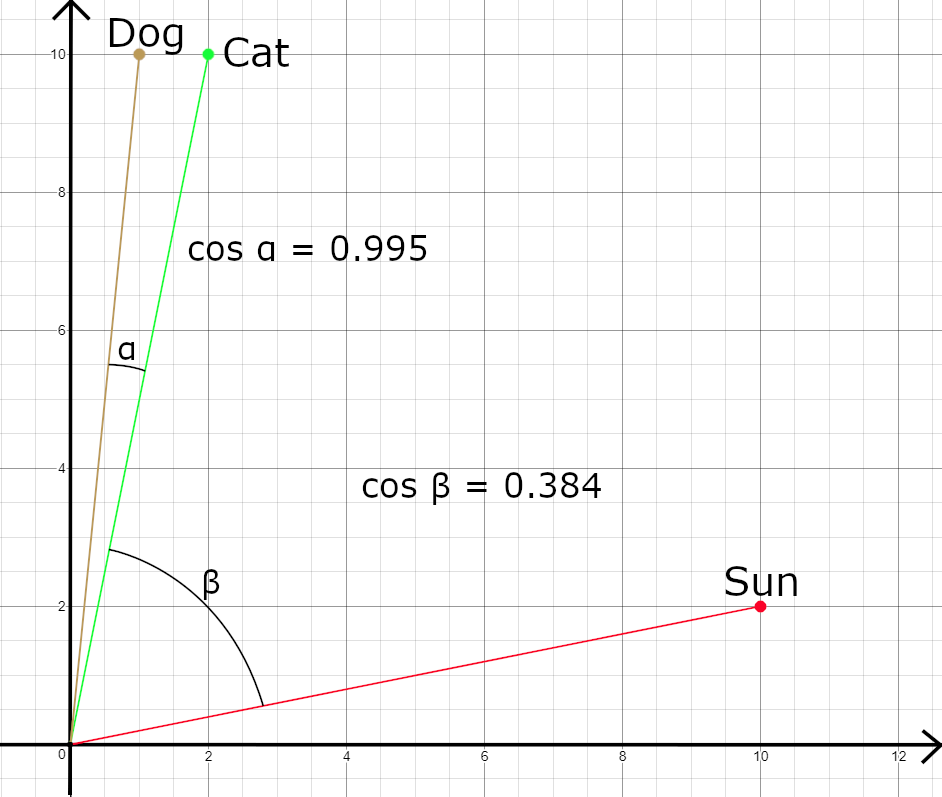
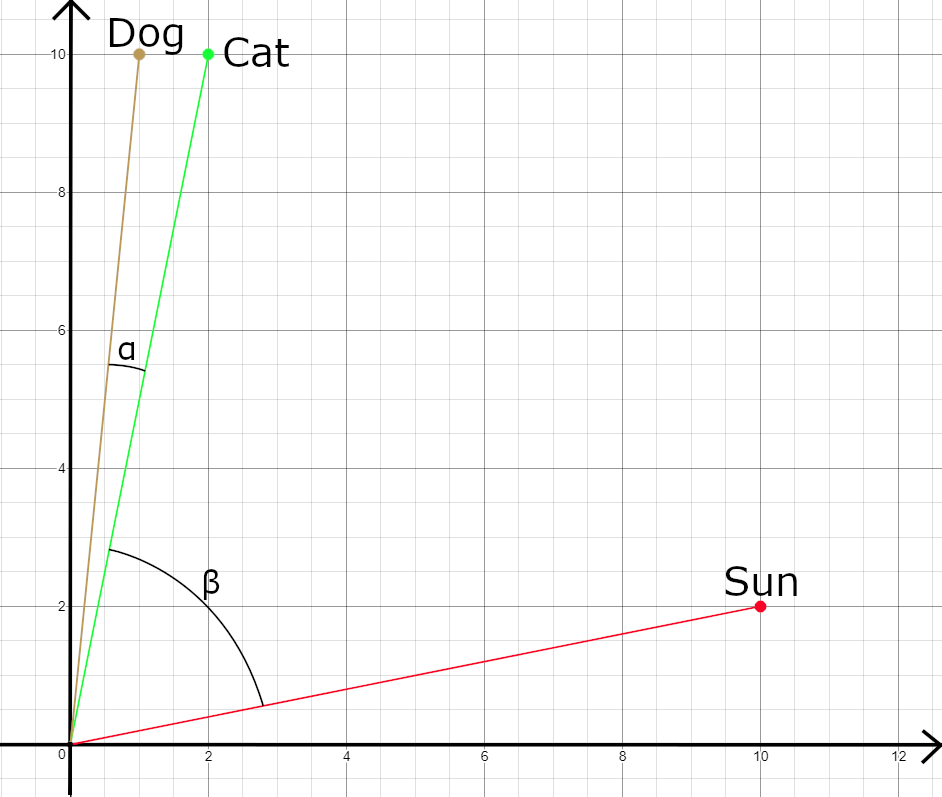
Let ‘cat’, ‘dog’ and ‘sun’ be vectors in the 2D plane.

‘cat’ will be represented as a vector

‘dog’ will be represented as a vector

‘sun’ will be represented as a vector

The dot product formula used to calculate the similarity of two vectors:



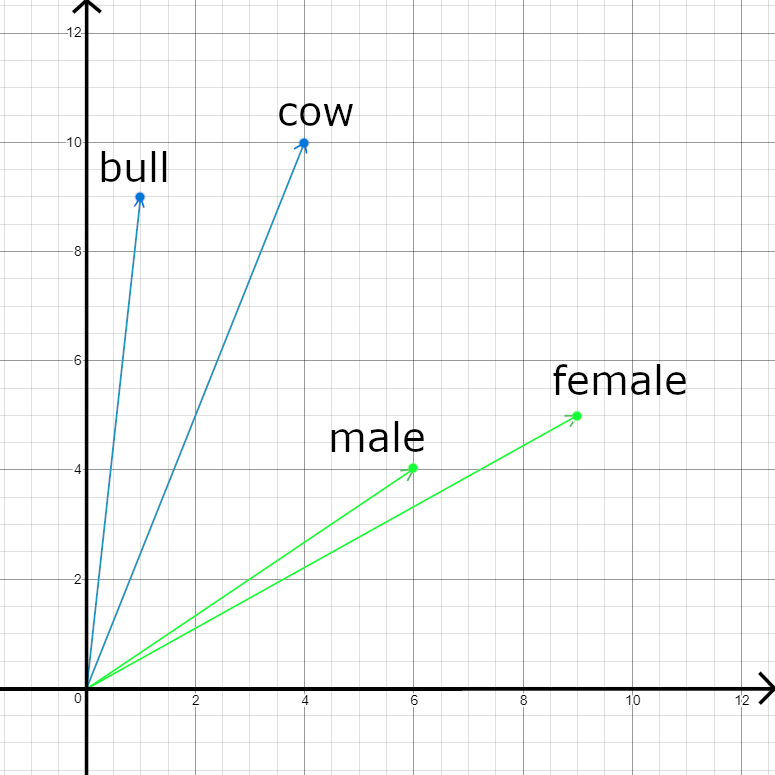
In the diagram above ‘dog’, ‘cat’ and ‘sun’ are represented as vectors on a 2D Cartesian plane. The cosine value of α is 0.995 thus close to 1 while the value of β is 0.384 which is closer to 0, thus representing the similarity or difference of the words.

Therefore, when large datasets are provided similar words will group together, thus forming different groups of words that are closely related such as a group of countries or animals.

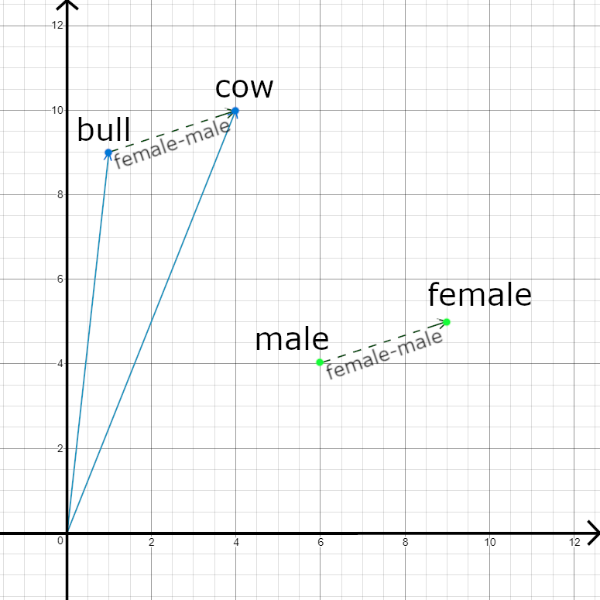
Word embeddings can also represent the relationship of words such as their tenses or male-female relationship.

For example:

The figure below has vector representations for ‘male’, ‘female’, ‘bull’ and ‘cow’



The next figure shows the vector addition of (bull + female-male) = cow. Since, the vector from ‘male’ to ‘female’ is represented as (female - male).



Further reading on the various types of word embeddings and a deeper explanation on the implementations of it can be found on these links:

<https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/>

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

<https://www.tensorflow.org/tutorials/representation/word2vec>

**Interpretable Word Embeddings:**

To enable explainable NLP an interpretable word embedding is needed to give further context to the words and be able to explain these contexts. Thus, this will result in tighter groupings between words and the reasoning behind them are known. Additionally, interpretable word embeddings allow for multiple meanings to be attached to the words, such as ‘tear’ as in liquid in the eyes when crying or ‘tear’ used when an object is ripped. This would therefore allow an NLP to explain the reason behind its result and allow for corrections to be taken when an undesirable output is given.

The creation of interpretable word embeddings is done by associating each dimension of the matrix representation of the words with a particular meaning. The embeddings should have a reasonable size so the calculations on these matrices would not be too taxing but still be very expressive in representing the contexts of the words. This should also be interpretable.

For word embeddings to be interpretable:

* It needs to be sparse
* Have few large values in a row or column
* Values are non-negative

Currently the constructions of interpretable word embeddings are split into two methods:

1. Through the enforcement of constraints during the construction
   * This is done with non-negative matrix factorization
   * And sparse neural embeddings
2. Using matrix transformations to make dense embeddings
   * With matrix rotation
   * Matrix factorization
   * K-sparse autoencoder
   * Linear mapping with the help of an external dataset

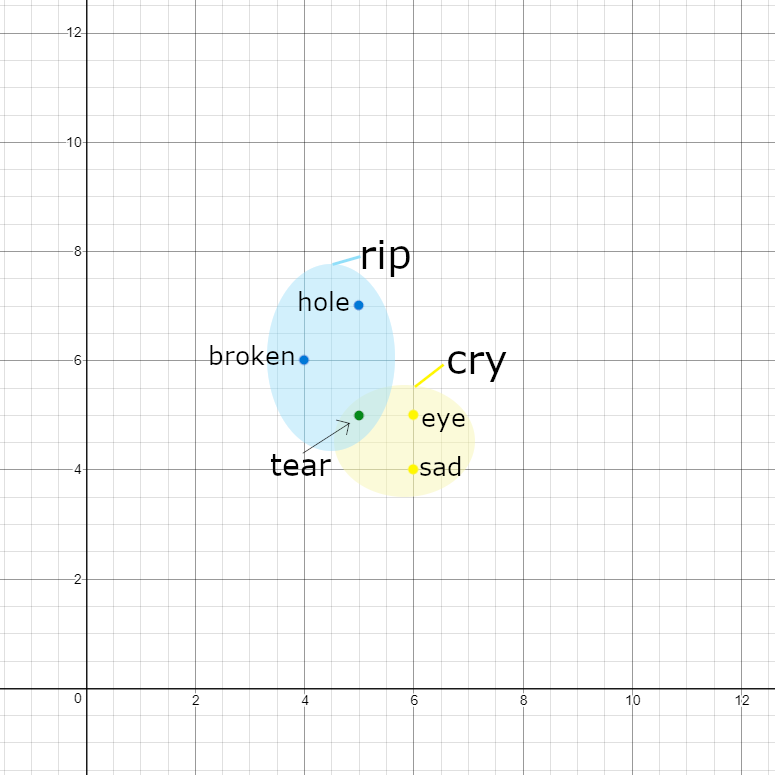
Examples:

1. Using Non-Negative Sparse Embedding <http://www.cs.cmu.edu/~bmurphy/NNSE/>

Representation for *tear*

|  |  |
| --- | --- |
| Weight | Words (per weighted dimension) |
| 0.69 | eye, sad, cry |
| 0.42 | rip, broken, hole |

1. Graph interpretation of adding additional context to ‘tear’



Further reading:

<http://aclweb.org/anthology/W18-5442>

<http://www.cs.cmu.edu/~bmurphy/NNSE/>

https://arxiv.org/pdf/1711.00331.pdf

OLD:

**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/>