Word2Vec Embedding in Natural Language Processing

Sean Jackson, Jiachen Jiang, Dr. Zhihui Zhu

What is Natural Language Processing?

- Natural Language Processing (NLP) is a field of study in Machine Learning.
- An example of NLP is ChatGPT.
- The focus of my time was the Word2Vec framework for NLP. As the name suggests, it assigns each word in a corpus (dictionary and/or collection of words) a vector value. With these values, similarities between words can be seen by examining how close two vectors are in vector space.

How do Computers Understand Words?

- Humans are very good at processing language (i.e. this conversation or reading this text), but computers cannot understand words.
- Thus, computers need words to be translated to numbers so they can understand them. This is a process called Word Embedding.
- This is typically done through representing words as vectors (https://www.turing.com/kb/guide-on-word-embeddings-in-nlp).
- For example, an analogy is representing places as 2D vectors of longitude and latitude.
  - **Washington DC** is at [38.9, 77]
  - **New York** is at [40.7, 74]
  - **London** is at [51.5, 0.1]
  - **Paris** is at [48.9, -2.4]
- Actual word embeddings/vectors will typically range anywhere from 100 to 300 dimensions (https://www.nature.com/articles/s41467-021-23795-5).

Image from: https://www.understandingai.org/p/large-language-models-explained-with

Neural Networks (NN)

- Words are turned to vectors using Neural Networks.
- Neural networks are large frameworks that can take in inputs and give outputs. Through a training method, the network tweaks its parameters so that it more accurately gives output for a given set of input.

How are Neural Networks Used in NLP?

- In the context of Natural Language Processing, neural networks are used in predicting a word given the words around it (context words) or predicting context words given a word.
- For example, if you gave a NN the words “The red truck” it may predict the next word as “stopped,” “slowed,” or “crashed.”

Basic Overview of Neural Networks

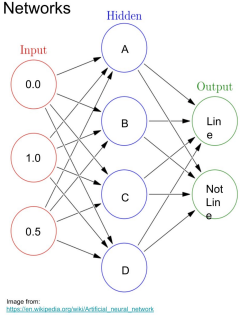

Image from: https://en.wikipedia.org/wiki/Artificial_neural_network

Neural networks consist of nodes. (Their name comes from how they look similar to neurons in our brain.) Here, there are three layers of nodes: input, hidden (there can be as many or few of these as the designer desires), and output.

Let us imagine a simple neural network that is trying to classify a 3-pixel black-and-white image. It will either classify it as a line (all three pixels are white) or not a line.

For a given training input, each node is given a value between 0 and 1. For our example, these values correspond to the intensity of the pixel (1 for pure white, 0 for pure black). In the picture, each input node has an arrow pointing to a node in the next layer. This represents the formula that is used to determine the value of the nodes in the next layer. Each input layer node has parameters a weight (w) and bias (b) associated with it. To find the value of A, you would add $((0.0)w_1 + b_1) + ((1.0)w_2 + b_2) + ((0.5)w_3 + b_3)$. This is technically not the final value of A, as you must take this value and plug it into a “squishing” function, or a function that gives it a value between 0 and 1.

This same process is used to determine the values of all the nodes in the hidden layer, and then the nodes in the output layer from the ones in the hidden layer. If the “Line” node has a larger final value, the NN has determined that the image is a line, and if the “Not Line” node has a larger final value, it has determined the opposite.

Cont.: Loss Function and Backpropagation

The result for each image from the NN will be compared to what the expected result should be (i.e. [1, 0] for “Line” and [0, 1] for “Not Line”). There is a loss function calculated between the expected result and the calculated resulted. By taking the partial derivatives of this loss function in terms of each weight and bias (something called a gradient in multivariable calculus), the NN can determine how to tweak its parameters in a process called backpropagation.

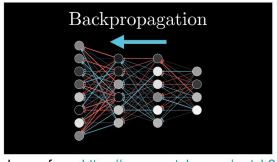

Image from: https://www.youtube.com/watch?v=Ilg3gGewQ5U

What is Word2Vec?

- Word2Vec is a popular way of word embedding.
- It is a way of associating words with vectors using neural networks.
- There are two main methods within Word2Vec: Continuous Bag of Words (CBOW) and SkipGram.
- CBOW uses the context words to predict likely next words.
- SkipGram uses a given word to predict context words.

Methods to Create Simple Word Embedding Framework

- PyTorch (Python Framework)
- Small Corpus (dictionary of all words)
- Window size of 5, batch size of 3 (hyperparameters)
- Stochastic Gradient descent

Parting Thoughts

- NLP will no doubt revolutionize the information sector going forth
- Word Embeddings are the basis of immense networks like ChatGPT
- It is important to be cognizant of the data we use and the biases they present. NLP, like all AI, will be biased because of its data. For example, stereotypes present in the writings of humans will show up in ChatGPT because that is the data it’s based on.

SkipGram in More Depth

- I focused on the SkipGram algorithm.
- SkipGram utilizes a neural network.
- Think of each input as a word in the dictionary, and each word/input is represented as a “hot vector,” where the place corresponding to the word is 1 and every other place is 0. These are the input nodes, and after going through the hidden layers, the output nodes with the highest values correspond to the words that the NN predicts are most likely to surround the given input.

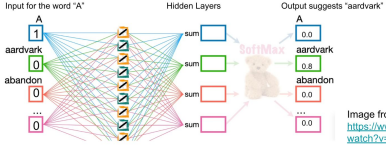

Image from: https://www.youtube.com/watch?v=viZrOnJclY0