Words in order

Description

Neural network researchers invented several methods that store and make inferences about the order of words in a sentence. The main method I will present here provides one of the components that undergirds the recent impressive performance of natural language processing (NLP) models known as *transformer* models.

The method also resonates with my prior investigations into the calibration of periodic cycles as a kind of urban rhythmanalysis. Trigonometric periodicities also feature in discrete cosine transforms (DCTs) for data analysis, image compression and steganography.

The periodic approach to temporal encoding of word sequences supercedes the recurrent neural network (RNN) approach, which Iâ??ll describe first.

Recurrent neural networks

A neural network (NN) takes patterns of activation as input and delivers patterns as output. You could train the NN to take one word as input and deliver another particular word as output, e.g. input = â??apple,â?• output = â??tree.â?• You train the NN on many such pairings and it adjusts its parameters to recall each of these pairings. Throughout the training session the NN algorithms makes fine adjustments to its parameters to preserve previous input-output pairings (or at least their influence on the network as a whole). Thatâ??s a serious optimisation task involving incremental error correction (back propagation, gradient descent, etc).

There are different method for feeding the input words to the NN. One common method is to use an indexed vocabulary. Here the NN is designed to have as many input and output nodes as there are words in the vocabulary (several thousand).

But contemporary NNs use a method that exploits the way words are embedded in the vocabulary.

Semantic embedding

As I showed in a previous post, itâ??s possible to use a neural network to define the position of any word in a multidimensional vector space. So each word in the vocabulary is associated with a list of decimal numbers (a â??vectorâ?•). A two dimensional space would have two coordinates per word, e.g. (12.5, 0.68). In three dimensional space the coordinates might be something like (12.5, 0.68, 56.0). But NN word embeddings may have several hundred parameters that make up the coordinates of a word. Typically, the number of dimensions for these word embeddings corresponds to the number of nodes in a hidden layer of a neural network tasked with calculating the relationships between words in a particular corpus of texts â?? several million occurrences of words in different contexts.

The coding of word embeddings is a representation that is fairly opaque to human scrutiny. NN network scholars claim that such embeddings capture the semantic relationships between words. The word encodings in the language model will all be of the same length and stored in the vocabulary list along with the word and its index number. The NN that uses them will have the same number of input nodes as there are dimensions in the multidimensional vector space (which is 2,048 in ChatGPT-3). The number of output nodes will equal the number of words (and tokens) in the vocabulary.

As the semantic embeddings contain information about relationships between words, that gives the NN a head start in matching inputs to outputs in ways that take account of word meanings in input-output word pairings. For example, if a network has been trained to associate the semantic embedding for â??appleâ?• with the semantic embedding for â??tree,â?• it will more readily team â??appleâ?• with â??orchardâ?• than â??appleâ?• and â??yacht,â?• as both â??treeâ?• and â??orchardâ?• will be relatively close in the multidimensional semantic space. Semantic embedding enables a NN to simulate a kind of analogical inference.

The sequencing challenge

Typically, the RNN method of representing information about sequences of words involves training a neural network on words two at a time in a loop. During training, each word of the sentence being â??learnedâ?• constitutes an input to the network, with the word that follows it in the sentence as the output. That second word then becomes the input for a training round with the third word in the sequence as output. The process continues as a looped operation until the end of the sentence.

By then the neural network contains weights and biases that have captured the ordering of the words in the sentence. The NLP saves the parameters for the state of the network as an attribute of that particular word sequence, ready for a full-on round of training for the whole corpus.

My summary here of RNNs seems to accord with the explanation in Jurafsky and Martinâ??s chapter in their book *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition,* about RNNs and LSTMs (long short training models), though I canâ??t always get ChatGPT (in conversation) to agree on that. Jurafsky and Martin write:

> In simple Recurrent Neural Networks sequences are processed one element at a time, with the output of each neural unit at time t based both on the current input at t and the hidden layer from time t -1. (208)

The main shortcomings of these RNN methods is that they easily â??forgetâ?• earlier parts of long word sequences. As with most NLPs, they also take a lot of computation. With this method the process canâ??t easily be speeded up via the parallel processing capabilities of the current generation of powerful GPUs (graphics processing units). Transformer models address these and other NLP shortcomings.

The transformer model

Currently, the most powerful method for storing sequence information is that based on the *transformer* NLP model, as outlined in the seminal paper â??Attention is all you needâ?• by Google researchers in 2017. They outline three components of the transformer model. Iâ??ll focus on the sequencing component here. This is the method used in ChatGPT and that most NLP researchers think provides a significant change in NLP capability, leading to the convincing performance seen in platforms such as ChatGPT.

I have found a series of videos by Richard Walker at *Luci Date* very helpful in explaining the *transformer* method, and how it handles sequencing. Walker introduces the theme by providing a compelling example for how the ordering of words really matters, and not just to get the grammar right.

The only change in each of the following sentences is the position of the word â??only.â?• Yet the meanings of the sentences differ, in some cases substantially. A language model needs to be able to handle word order to account for such shifts in meaning.

Only Luci told Claude that she loved him.
Luci **only** told Claude that she loved him.
Luci told **only** Claude that she loved him.
Luci told Claude **only** that she loved him.
Luci told Claude that **only** she loved him.
Luci told Claude that she **only** loved him.
Luci told Claude that she loved **only** him.
Luci told Claude that she loved him **only**.

As described above, words are represented to an NLP neural network as coordinates within a multidimensional semantic space, i.e. a list of numbers. If the vector space is 2,048 dimensions, then the positional vector of any word will be a list of 2,048 numbers. Thatâ??s unwieldy to illustrate, so Walker considers a space of just 5 dimensions. That means that each word in a sentence will be represented by a list of 5 numbers. Iâ??ll change his example sentence, but use the method he describes. Hereâ??s my sentence to illustrate positional embedding.

Graffiti is a social good.

We ignore capitalisation and punctuation for simplicity. The word â??socialâ?• has proximity relationships with other words in the multidimensional space of the whole training corpus. â??Socialâ?• is close in terms of meanings to â??community,â?• and far from words like â??sunset,â?• â??chainsawâ?• etc. That â??semanticâ?• information is already captured by its coordinates in the

multidimensional space. Imagining a semantic space confined to just 5 dimensions (instead of several hundred) might produce a series of 5 parameters for each word as follows. (These numbers are made up and copied from Walkerâ??s example.)

Semantic encoding

Graffiti	-36.65	59.47	35.25	-21.78	-33.44
is	19.66	-62.65	34.55	88.36	-50.35
a	89.78	33.45	-80.82	-11.63	77.30
social	39.49	-34.71	46.79	-55.97	-83.68
good	-68.02	-31.98	-6.96	-46.72	63.36

The transformer method involves combining this semantic embedding with positional embedding for each word as it is fed into the neural network during training.

The seminal Google article by Vaswani et al (â??Attention is all you needâ?•) presents the method for positional encoding. It is not entirely intuitive, or easy to follow, but the authors claim that it seems to work. Iâ??ll explain it as best I can here.

In this method the positional information for the *i*th parameter at position *k* in the word sequence (i.e. sentence) for a multidimensional semantic space of dimension *d* is defined differently for the odd and even values of *i*. Vaswani et al also introduce a scaling factor of *n* adjusted by the algorithm designers to vary the influence of the positional embedding when it is combined with the semantic embedding. They use a value of 10,000. Here I use the smaller value of 1,000 so that the effects of the positional embedding are more obvious.

Here are the formulas they use. For even values of *i*, calculate the embedding vector component as

$$\sin\left(\frac{k}{n^{\frac{2i}{d}}}\right)$$

For odd values of i

$$\cos\left(\frac{k}{n^{\frac{2i}{d}}}\right)$$

Assisted by a helpful video by Mehreen Saeed, I reproduced the method on an Excel spreadsheet to produce this table.

Position in sequence	Positional encoding				
0	0.00	1.00	0.00	1.00	0.00
1	0.84	0.54	0.16	0.99	0.03
2	0.91	-0.42	0.31	0.95	0.05
3	0.14	-0.99	0.46	0.89	0.08
4	-0.76	-0.65	0.59	0.81	0.10

Iâ??ve left out the actual words as they donâ??t matter here. The values in the encoding would be the same for any five word sentence with the same dimension number for the semantic encoding and the same *n* factor.

The introduction of trigonometric functions to indicate position is at first esoteric and arbitrary. As I stated above, the authors of â??Attention is all you needâ?• seem to suggest simply that this is something they tried and it works.

Combining semantic and positional embeddings

These values are added to the semantic encoding to provide for slight discriminations between words so as to reflect their positioning. The combined numbers are unique representations of the absolute and relative positions of the words in the sentence.

Position in sequence	Words in sequence	Semantic encoding						Positional encoding						Sum of semantic and positional encoding			
0	**Graffiti**	-36.65	59.47	35.25	-21.78	-33.44	+	0.00	1.00	0.00	1.00	0.00	=	-36.65	60.47	35.25	-2
1	**is**	19.66	-62.65	34.55	88.36	-50.35		0.84	0.54	0.16	0.99	0.03		20.50	-62.11	34.71	8
2	**a**	89.78	33.45	-80.82	-11.63	77.30		0.91	-0.42	0.31	0.95	0.05		90.69	33.03	-80.51	-1
3	**social**	39.49	-34.71	46.79	-55.97	-83.68		0.14	-0.99	0.46	0.89	0.08		39.63	-35.70	47.25	-5
4	**good**	-68.02	-31.98	-6.96	-46.72	63.36		-0.76	-0.65	0.59	0.81	0.10		-68.78	-32.63	-6.37	-4

If the word order in the sentence is changed by swapping the positions of â??graffitiâ?• and â??isâ?•, then the order of the semantic encodings changes, and the sum of the semantic encoding and the positional encoding will also change.

Position in sequence	Words in sequence	Semantic encoding						Positional encoding						Sum of semantic and positional encoding			
0	**Is**	19.66	-62.65	34.55	88.36	-50.35	+	0.00	1.00	0.00	1.00	0.00	=	19.66	-61.65	34.55	8
1	**graffiti**	-36.65	59.47	35.25	-21.78	-33.44		0.84	0.54	0.16	0.99	0.03		-35.81	60.01	35.41	-2
2	**a**	89.78	33.45	-80.82	-11.63	77.30		0.91	-0.42	0.31	0.95	0.05		90.69	33.03	-80.51	-1
3	**social**	39.49	-34.71	46.79	-55.97	-83.68		0.14	-0.99	0.46	0.89	0.08		39.63	-35.70	47.25	-5
4	**good**	-68.02	-31.98	-6.96	-46.72	63.36		-0.76	-0.65	0.59	0.81	0.10		-68.78	-32.63	-6.37	-4

In his video on the method, Richard Walker points out that the trigonometrical positional encodings will lie between 0 and 1. The values are small, but sufficient for a neural network trained on a corpus of word sequences to take account of the positioning of words as it â??learns,â?• and as it makes inferences to generate new sentences. As the training and inference algorithms deal with the positional embedding for each word independently they can be more easily processed in parallel, thereby speeding the calculations.

Walker brings the periodic method home to something more human about how we experience time and order.

> Consider how we represent time. We donâ??t use a single number. We use a vector, and the elements in those vectors are periodic: hours, minutes, seconds â?? all of them at different frequencies, as do days, months and years. Of course, our motivation to represent time as a multi-element vector is to fit in with our lifestyles. The elements have significance. We start work on a Monday. We meet for dinner at 7:30 pm. We plant our crops in March. The elements in our time vector mean something to us on a human scale. This is not so different from positional encodings. We are working here at AI scale, giving our artificial neurones something to latch on to, to be able to look at the context of a word relative to its neighbours.

He concludes with a quote from linguist John Firth: â??You shall know a word by the company it keeps.â??

Bibliography

- Buduma, Nithin, Nikhil Buduma, and Joe Papa. *Fundamentals of deep learning designing next-generation machine intelligence algorithms*. Beijing: Oâ??Reilly, 2022.
- Jurafsky, Daniel, and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Third Edition*. Redwood City, CA: Stanford University Press, 2023. (draft available as PDF)
- Saeed, Mehreen. â??A Gentle Introduction to Positional Encoding in Transformer Models, Part 1.â?• *Machine Learning Mastery*, 20 September, 2022. Accessed 13 March 2023.

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. â??Attention Is All You Need.â?• In *31st Conference on Neural Information Processing Systems*, 1-15. Long Beach, CA, USA, 2017.
- Walker, Richard. â??ChatGPT Position and Positional embeddings: Transformers & NLP 3.â?• Luci Date, 26 February, 2023. Accessed 12 January 2023. https://www.youtube.com/watch?v=DINUVMojNwU
- Walker, Richard. â??Attention â?? the beating heart of ChatGPT: Transformers & NLP 4.â?• Luci Date, 26 February, 2023. Accessed 12 March 2023. https://www.youtube.com/watch?v=sznZ78HquPc
- Walker, Richard. â??ChatGPT â?? Semantics: Transformers & NLP 2.â?• Luci Date, 12 December, 2022. Accessed 12 March 2023. https://www.youtube.com/watch?v=6XLJ7TZXSPg

Notes

- Feature image is from MidJourney, prompted with â??you will know a word by the company it keepsâ?•
- For a further explanation of the use of trig curves in defining time sequences see the video at 3:50 Anon. â??Attention is all you need || Transformers Explained || Quick Explained.â?• *Developers Hutt*, 2022. Accessed 2 April 2023. https://www.youtube.com/watch?v=66seIToeguE

Category

1. Artificial Intelligence

Tags

1. language
2. NLP
3. positional embedding
4. sequencing
5. transformer

Date Created
May 6, 2023
Author
rcoyne99