AI learns ABC

Description

Neural networks (NNs) as deployed in automated language processing (NLP) are good at identifying and reconstructing patterns in data. So a NN trained on a corpus of texts can identify words that are commonly grouped according to their proximity within sentences, e.g., we wouldnâ??t be surprised to find words (tokens) such as â??building,â?• â??services,â?• â??constructionâ?• in each otherâ??s company in a text about architecture.

Language is however more than bundles of words. Phrases, sentences and paragraphs are typically arranged following an order, which affects comprehension, meaning and grammar.

It turns out that training a neural network to detect and predict patterns in the ordering of tokens, as in a sentence, is difficult, especially if the process is to be handled efficiently as is required for conversational AI. In conversations, the AI model needs to process the order in which tokens appear in the communicators data stream, and do so one token at a time reaching back to the start of a defined context window. This window may be several hundred tokens long.

The method deployed by AI models built on the generative pre-trained transformer (GPT) model is known as â??**positional encoding**.â?• Iâ??ve investigated this before (see post: Words in order), but now Iâ??ve implemented a demonstration in the Python programming language.

The construction of a home-made NLP program trained to produce arbitrary but reasonably grammatical sentences is currently beyond my desktop capabilities , even with expert assistance from ChatGPT4. Evidence my latest attempt from a model trained on 14k of my own text and attempting positional encoding:

> distance points. Motivation. Giant bernard equilibrium change language and positioning common as support also site parks consists response were meaning communications floresiensis layer other marshalled plausible neural particular win i.e. Online converse notion.

Itâ??s difficult to recognise if the word order here is any better than random. To test the effectiveness of positional encoding I decided to revert to a well-known sequence, the English alphabet. It would be easy to detect if my AI model is able to reproduce that sequence from fragments.

I adopted the character string â??abcdefghijklmnopqrstuvwxyzabcde,â?• and wrote a program to fragment it randomly into shorter segments, e.g., cdefghi, tuvwx, onpqr, opqsrut, ijklnmop, lmnpoq, efghi. I created 500 such segments between 2 and 8 characters in length. The fragments overlap and I introduced some noise by having the program occasionally swap the order of the letters.

The objective was to train a neural network model so that I could then prompt it with a letter sequence, such as â??def,â?• and it would continue indefinitely with the complete alphabet, even looping after â??zâ?• with â??abcâ?• etc.

Semantic and positional encoding

I had to scale down the specifications for full NLP. I ran these 500 segments through a neural network program that derives the strength of relationships between each of the alphabetical characters, according to their frequencies and adjacencies in the various segments. The network assigned each character (a, b, c, etc) to a 4 dimensional vector. For example, the program assigned c to a vector (1.630, -0.090, 1.303, 4.088). Thatâ??s the semantic encoding of that character. See post: Architecture in multidimensional feature space.

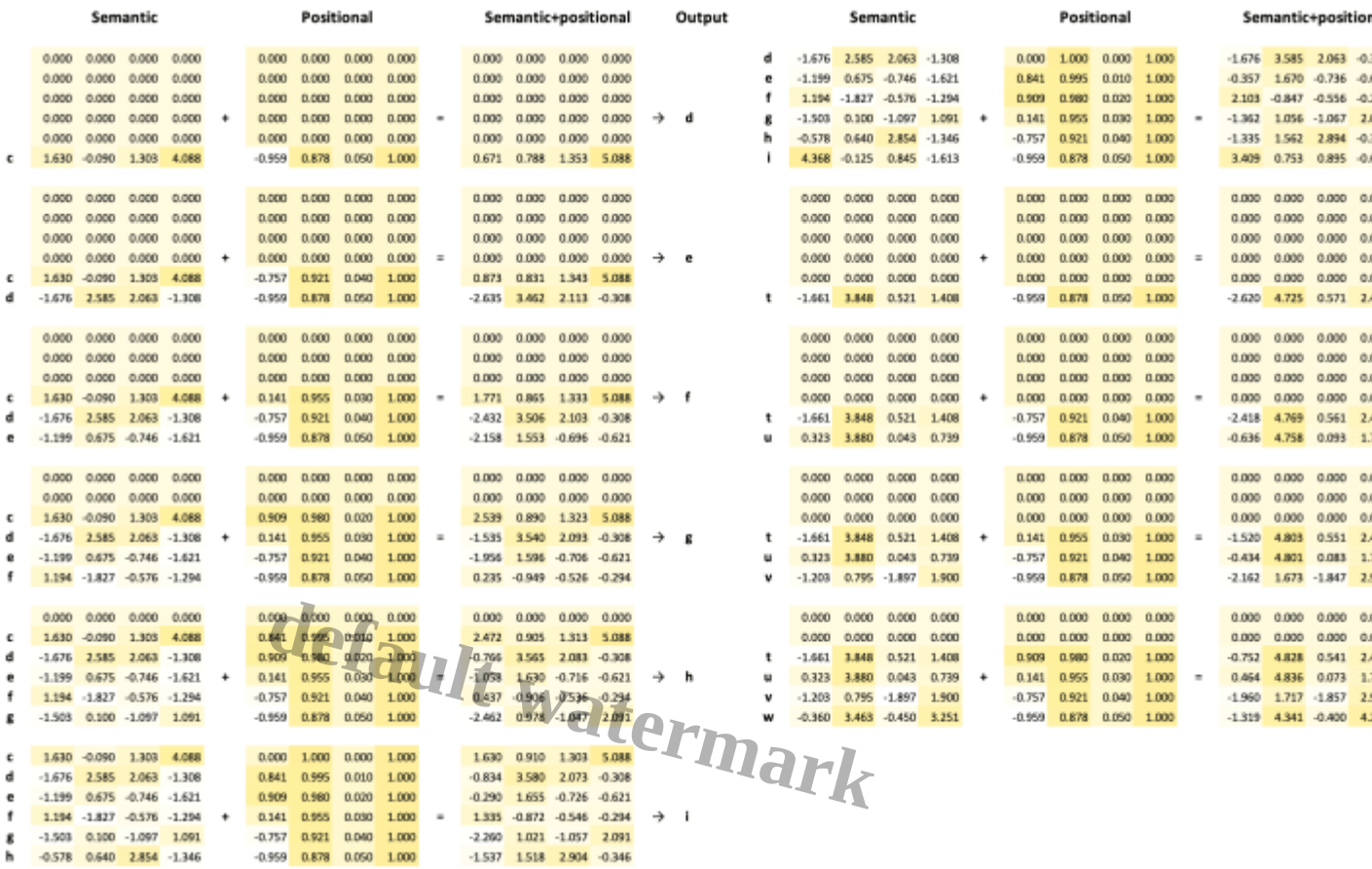
Sliding the context window

I defined the context window as just 6 characters. The semantic vector for each character slides into the context window one character at a time, with the rest of the characters in the sequence following in its wake, until the last one enters and another sequence starts. Each step in this procession through the context window constitutes a training event where the NN is trained on inputs that correspond to outputs.

The input to the neural network during training is not just the semantic vector for each letter in turn, but the vectors for the entire context window. So a neural network for training with 4d semantic vectors and a context window of 6, has 6Ã?4 = 24 input nodes. Itâ??s a â??flattenedâ?• 6Ã?4 matrix. On the output sides there are as many output nodes as there are characters, 26 in this case.

Information about the position of an input character in its sequence is captured via a positional encoding matrix. It also has 6Ã?4 dimensions and moves through the context window with the semantic embeddings. The values within each of those matrices are added, and its the sum of the semantic and positional matrices that is supplied as input to the neural network. As the network is being trained to predict, the output is the next character in the sequence.

The procedure is captured in this chart. (Click to enlarge.)



Prediction

In the prediction phase of the NN model we provide some new input and ask the network to work out what comes next, and to do this in a loop to create an output sequence. The prediction phase of the network operation has to adopt the same procedure as the training phase: adding semantic and encoding vectors.

The mysterious element in this procedure is the formulation and role of the positional encoding, and why adding positional vectors to the semantic vectors captures sequential information. Explaining that is for another time, but for now â?? here are some outputs to demonstrate that positional encoding does make a difference.

Testing the model

I can use my alphabetically trained model to make predictions, such as predict what follows a b c. I can also suppress certain features, such as eliminating the positional or semantic encoding from the training and prediction algorithms to test what differences they make.

With just semantic encoding the model generates a sequence that at best looks like fragments joined randomly.

a b c b d e f f h j k m l y z z y z y v w x y z y z b d c f f e f h i j n p q r s t u v w x y z a b c d f f g h i j k l m n y a z y z n s a v x y y z a b b d c t z z b c d e f i h j n o p p q r s t v v x y

The positional encoding on its own looks even more random.

a b c d x v g s d n v k o e s i p d i k k o q b n z r y p r b f g d f s d f v l b s v g r e p e y g h o d s r b j j n g p q t x s i b x t k e l d i s p a i a f g m l l n c j x s f o j d o p p c v z i o f y

Here is the output when the semantic and positional encoding are taken into account as proposed.

a b c d e f g h i j l m n o p q r s t u v w x z y a z b c d e f g h i j k l m n o p q r s t u v w x y z a b c d e f g h i j k l m n o p q p r s t u v w x y z a c b d e f h g i h j k l m n o p q r s t u v w

There are some random glitches, reflecting the imperfections in the training data. Each of these examples is sampled from a number of trials, using the same global parameters, such as the number of training epochs, and the â??temperatureâ?• in the prediction algorithm that influences the degree of randomness in the output.

Alphabetical ordering is a deterministic domain with a single identifiable outcome. I could also have used the Periodic Table, an ordered list of the most popular films about AI, or an ordered list of shops in my local shopping mall.

Predicting a full sequence from fragments where the final sequence is less well defined poses particular challenges, and would arguably be more useful. DNA sequencing from DNA fragments comes to mind.

Reproducing the alphabet in this way increases my confidence that positional encoding works and can be applied in different contexts, such as way finding and journey planning.

Note

- The feature graphic is from Bing, prompted with â??a simple post-apocalyptic stretched horizontal banner that clearly says â??a b c d e f h k g a i.'â?• See https://www.bing.com/images/create?
- Note that the semantic and positional encodings are calculated to 9 or more significant figures, rounded down to just 4 here to fit on the screen.
- This process crunches a lot of numbers to approximate something that could be achieved by a simple algorithm, but demonstrates the process of training a neural network on the order within a text string, as well as the nature of the challenge.

Category

1. Artificial Intelligence

Tags

1. alphabet
2. journey
3. path
4. positional encoding

Date Created

October 21, 2023
Author
rcoyne99