

ECE 1786 Lecture #4

Work-in-Flight: Assignment 2 - Classification of Language, revised deadline, Tuesday Oct 10 at 9pm

Last Day: Classification of Language Using Word Embeddings

Today: 1. Introduction to Language Models & Transformers
2. Course Project Structure, Scope and Deliverables

Assignment 2 has you training two versions of a neural network classifier that takes in sentences and produces a classification - that the sentence is either objective or subjective. The same method can do other types of classification.

- Sentences are encoded with pre-trained word embeddings
- A2 asks you to let the embeddings also be trained/tuned

Introduction to Transformers

- the reason I'm teaching this course is that, in my research on building therapeutic chatbots, we found that pre-trained transformers produced remarkable results, over 4 years ago, with GPT-2 then 3. Subsequent improvements have continued to be stunning.

Transformers are the state-of-the-art method for:

1. Classification of language - not MLP, or CNN as in A2, and not recurrent neural networks (RNNs) as found in the pre-2018 literature.
 2. Generation of language, which we haven't discussed yet outside of lecture 0.
 - Generation is qualitatively different than classification (in the same way reading and writing are different, perhaps), but neural networks that can do one can be re-purposed to do the other.
- That said it is all just prediction!

We will focus on classification for lectures 4 and 5, and then generation in lectures 6,7, and 9.

Let's begin by describing a core topic in NLP: Language Models

- GPT-1/2/3/4 are called "Large Language Models" [see Jurafsky 3.0/3.1]

"Language Models are models that assign probabilities to words." Which means what???

- It is the probability that the next word in a sequence of words is X
- .e.g $P(\text{"frog"}) = 0.2$
- But what does it mean to have a probability of a word?

If that word is the next word in the partial sentence, then the sentence will be

1. Grammatically correct
2. Makes sense

- i.e. that the sentence or the partial sentence is likely to found in the use of the language

e.g. Partial sentence: "I believe clean running water is important for ..."

High probability: Good health; success; everyone;

Low probability: lights; computers; desks

The task of a language model is to do this:

Given: One or more words in a sequence

Compute: probability of every word in the vocabulary being the next word

- Perhaps according to 1 & 2 above, but maybe more

i.e. assume that the size of the vocabulary (# words) $|V| = M$

(So we have W_0, W_1, \dots, W_{M-1} as the words)

Now, given a sequence of n words - X_0, X_1, \dots, X_{n-1}

Compute: $P(W_0 \text{ is } X_n)$

$P(W_1 \text{ is } X_n)$

⋮

$P(W_{M-1} \text{ is } X_n)$

} i.e. across the whole vocabulary.

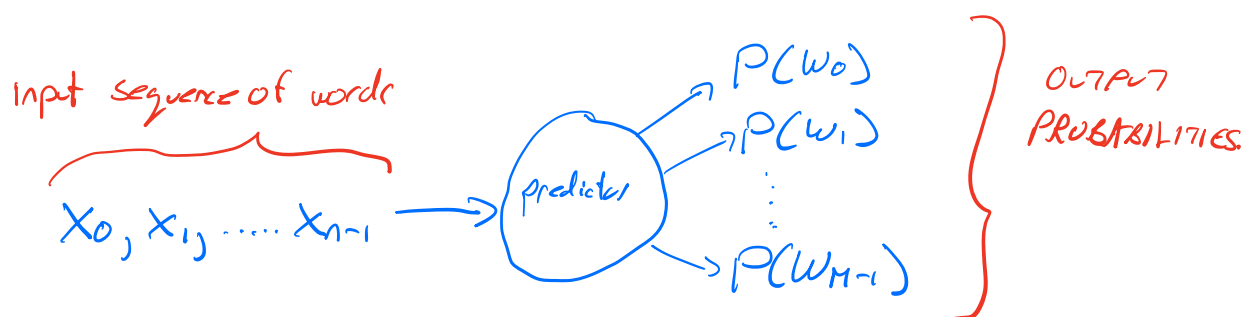
- Given that we can do this, we can use these probs to compute the 'likelihood' of an entire sequence of words. (Again, the likelihood that the sequence is grammatical/makes sense; or that this sequence would be found in the use of the language)
 - Do by computing $P(X_0) P(X_1) \dots P(X_{n-1})$ for an n-word sequence. (Multiply)
 - We can judge a language model by computing this probability on a fixed sequence of words that are known to be good. Must always use the same sequence to compare different models
 - The **Perplexity** of a model is a function of the above product, but it is both inverted (so that lower numbers are "better") and normalized by taking the nth root.

Perplexity of a model can only be measured on a specific test set of n words, where you compute $P(x_0)$ being the 1st word, $P(x_1) = P(x_1 | x_0)$,
 $P(x_2) = P(x_2 | x_1, x_0) \dots P(x_{n-1}) = P(x_{n-1} | x_0, x_1, \dots, x_{n-2})$

this means the probability that x_2 would be the word that follows the words x_0, x_1 according to the model.

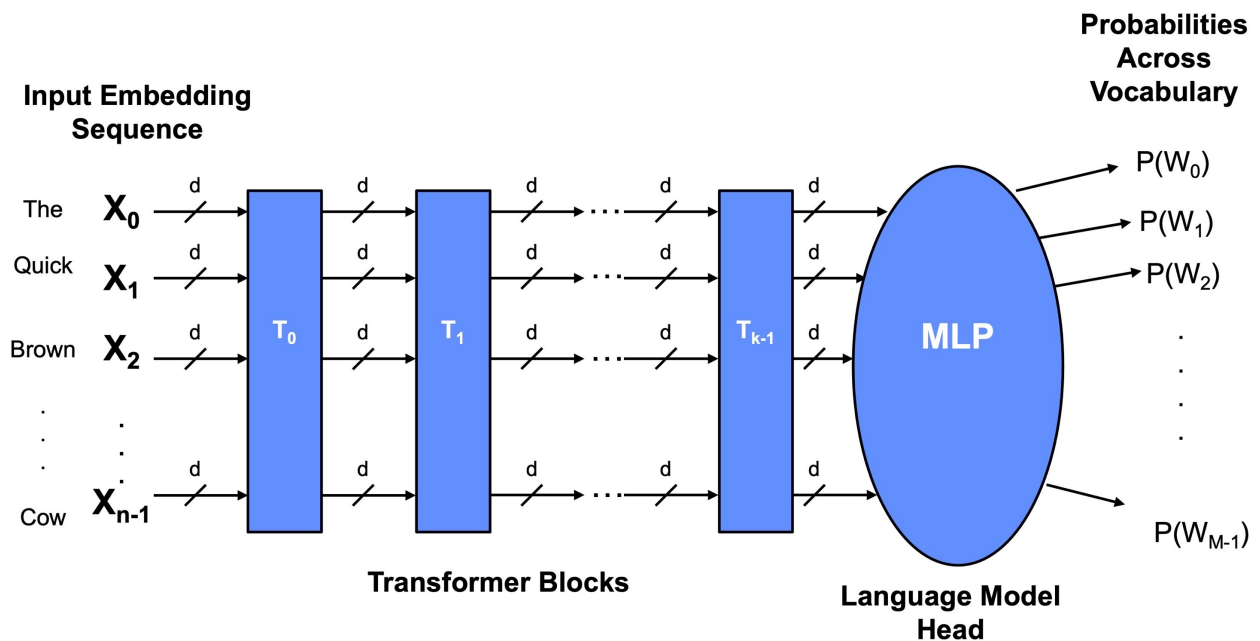
- In the literature, you'll see perplexity being used as a metric to judge models, and sometimes the output of models themselves. Lower is better.

So, the "language model" that we want is a predictor that looks like this:



- Does this look familiar? (Assignment 1, Section 3)
- Except that there is more than 1 word input. (There are n words)

- We want to train a neural network to make this prediction
 - The X_i would be input as word embeddings, (of course?)
 - Could use pre-trained embeddings, like GloVe **OR** could have the embeddings themselves be learned as part of the process (i.e. randomly initialized like the rest of the network).
- Where can we find the training data/examples?
 - **Everything ever written, again!**
 - Except it turns out, that this time, we are unlocking something very very powerful -> chatGPT, GPT-4
- The transformer architecture that we will use (first, as a classifier) is actually trained to do exactly the above prediction - predict the probability that each word in the vocabulary is the next word after a sequence.
- Here is a one view of the picture of a Transformer:



We will (but not yet) dive into the details of the Transformer block in Lecture 5

- Suffice it to say that it is a neural network of some kind

Three important comments/insights:

1. One reason this is called a Transformer is that, for each T_i block, the number of inputs = number of outputs. So, the information coming in is 'transformed' but not increased or decreased in size. There are $d \times n$ numbers coming in and $d \times n$ going out, where d is the embedding size.
 - This allows an number (K) of T_i blocks to be stacked
 - Using that, one way that the big transformers are made big is by adding more blocks this way, i.e. just making K bigger
2. The number of embeddings coming in - n - is often called the **context size**, and the input itself is called the **context**; it is very important
 - Context is very important in all human communication!
 - Original Vaswani Transformer, $n = 512$
 - GPT-2 $n = 1024$ (1K)
 - GPT-3 $n = 2048$ (2K)
 - GPT-4 $n = 8K \rightarrow 32K$
 - While bigger seems to always be better, the attention block inside the transformer is n^2 in the computational complexity; we'll cover that next, but some of the newer models are finding ways around it.
3. Observe the "Language head" which, to repeat, looks just like the output in Assignment 1, Section 3.
 - A. This is used when training the network to be a language model; **key note: when we want to use this network to do classification, we chop off this language head, and put a classifier head - an MLP just to do the classification on it, and train it as a classifier, with the parameters of the pre-trained transformer blocks left intact.**
4. The big models are most useful when pretrained on lots of words - e.g. GPT-2 was trained on billions of words, GPT-4 trillions

Consider this sentence being used to create training examples:

The smooth blue lake became choppy in the wind.

Training example 1:

input:

output label: The

Training example 2:

input: The

output label: smooth

Example 3:

input: The smooth

output label: blue

etc.

For example 3, the model computes $P(\text{'blue'})$ (& all the other probabilities), but using the log loss, since **blue** is the right answer, the loss that is computed for this example is

$$\text{loss} = -\log(P(\text{'blue'})) \quad [\text{just as you did with cross entropy loss in A1}]$$

- For a batch of examples of size b , compute

$$\text{Average loss} = \frac{\sum_{i=1}^b \text{loss}(\text{training example } i)}{b}$$

- sometimes the term 'teacher forcing' is used to say that you don't use the predicted answer in the next example, you use the correct example
- There are trillions of examples of writing, unlike most other ML problems; that writing contains most of human knowledge!

Note 1: the sequence of words input - $X_0, X_1, X_2 \dots$ does not contain any information about the order of the words (despite it looking like it does).

- Yet that ordering must certainly matter! (RNNs don't have this issue)
- In the original Vaswani paper, they add a positional embedding to the word embedding, to provide that information.
- The number is both initialized in various ways (absolute, relative, sinusoidal) and is itself learned.
- However, I have found this very confusing, as some Transformers appear to learn without these embeddings; there is a recent paper that suggests that the training itself does give a hint as to what the word order is.

Note 2: The word embeddings themselves are learned as part of the same training process, as discussed above.

Note 3: In lecture 5 we will talk about the immense power the above training puts into a model. Consider chatGPT & what it can do - it can answer any question. Even moreso: the big models appear to do almost anything we ask them to do, including classification, but many more things.

Next lecture: the internal structure of the Transformer block

Now: The project structure.