ECE 1786 Lecture #4

Last Day: Classification of Language Using Word Embeddings

Work-in-flight: Assignment #2 — modification (A2 V2) released SATURDAY, making 5.1 to be 4.8

Today: ① Introduction to Transformers
② Course Project Structure, Scope, Deliverables

Where are we? Working on A2, making/training a neural-net classifier that takes in Sentences and produces a classification. ⇒ objective / subjective but could be many others.
⇒ sentences are encoded with pre-trained word embedding
⇒ side note: A2 also asks you to try letting the word embeddings themselves be trained/tuned. as part of ~~grad~~ grad. descent
⇒ optimizing the embeddings with respect to the specific application of classifying objective/subjective

---

Introduction to Transformers

- the reason I'm teaching this course is because, in my research on making therapeutic chatbots, we found a pre-trained transformer produced remarkable results - over 3 years ago; + subsequent improvements & capabilities are both stunning & fundamental in many ways.

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

① Classification of language - not MLP or CNN as in A2 ☺
-reading/understanding 'NLU'

② Generation of language - which we haven't yet discussed outside of Lecture Ø; I believe generation (writing) is qualitatively different than classification, even though the NN structures overlap; [well reading & writing are related] aren't they

- will focus on 1 for L4/L5 ↗A3 2 L6/7. ↗A4.

Let's begin by describing a core topic in NLP: Language Models – GPT-1/2/3/4 are called "Large Language Models"; Jurafsky Chap 3/3.0 3.1

"Language models are models that assign probabilities to words" ??
- probability that the next word in a sequence of words is word x: e.g. P("frog") = .2
- meaning what? that if that word is added next, that the sentence, that to that point: ne
  1. Is grammatically correct
  2. Makes sense

e.g: I believe clean running water is important for ... ?

high prob
- good health
- success
- everyone
-

| lights × computers × desks × low prob

The task of a language model is to ~~Once we can~~ do this:

Given: One or more words in a sequence

Compute: The probability ~~that~~ across every word in the vocabulary that it is the next word.

i.e. assume the size (# words) in vocabulary $|V| = M$ → $W_0, W_1, \vdots, W_{M-1}$

Given a sequence of n words $X_0, X_1, \dots X_{n-1}$

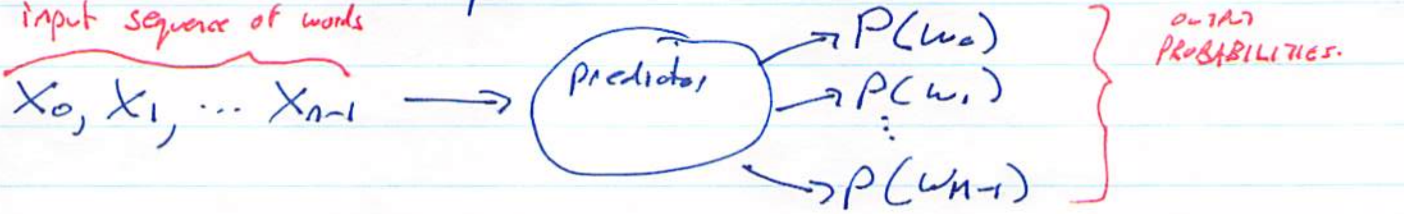
Compute:

$$\left.\begin{aligned} &P(W_0 \text{ is } X_n) \\ &P(W_1 \text{ is } X_n) \\ &\vdots \\ &P(W_{M-1} \text{ is } X_n) \end{aligned}\right\} \text{i.e. across whole vocab}$$

- aside: given that we can do this, we can use these probabilities to compute the "likliehood" of an entire sequence of words. ↳ likliehood that it is grammatical/makes sense

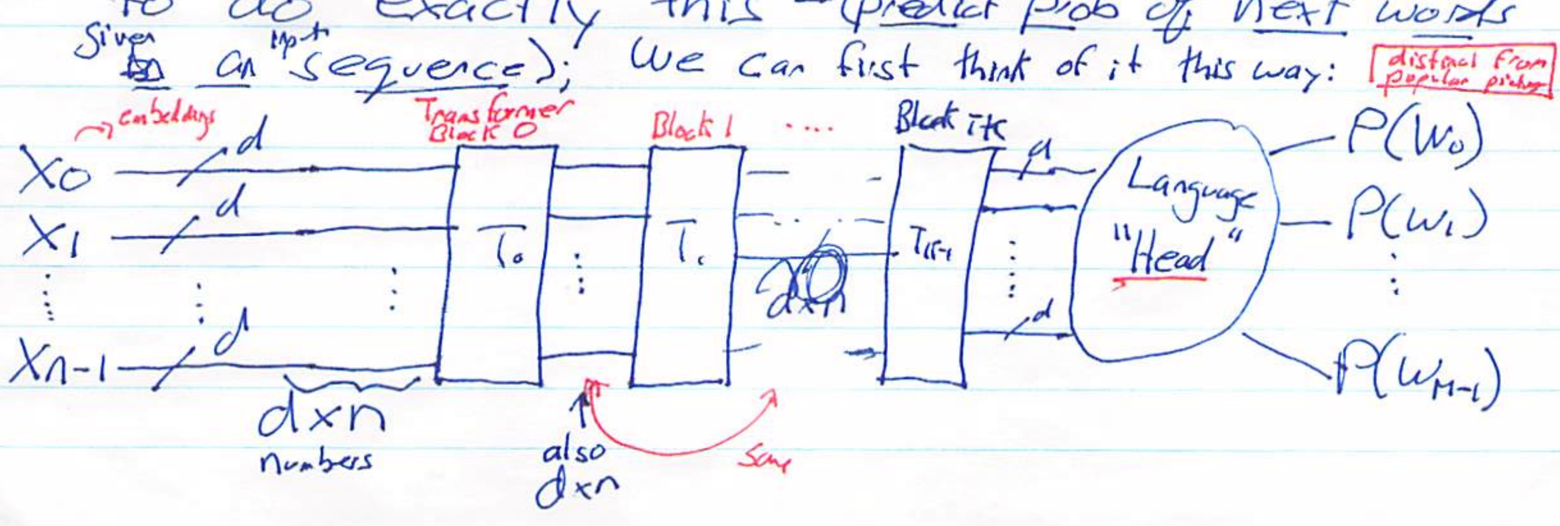
⋮

- i.e. want a predictor that looks like this:



- does this look familiar? (A1 S3)
  ↳ except > 1 word input) – order didn't matter
- we want to train a neural network to do this ↑
  - the X_i are word embeddings, of course
  - either pretrained (e.g. GloVe) OR trained as part of the process [L12, A2]
- where can we find the training data/examples?
  → everything ever written, again!
  → except now we might be unlocking something very powerful.
  amount of data is unprecedented

- The Transformer architecture that we will first use as a classifier is actually trained to do exactly this ↑ (predict prob of all possible next words given an input sequence); We can first think of it this way: [distinct from popular picture]



- we will (but not yet) dive into Transformer Blocks $\boxed{T_i}$ (L5)

Three important comments / insights:

① Perhaps one reason this is called a Transformer is that, for each Tr block, the #inputs = #outputs and so the information is "transformed" ('changed') but not increased or decreased. d×n in d×n out d = embedding dim
- this allows any number of Tr blocks to be stacked on top
→ that is one way the big Transformers are made big (smaller) → more blocks.
  → evidence has been that more = better, if sufficient data

② That size, d×n, or just n assuming fixed d. is often called the context size; very important.
  - Context is very important in all communication → re llc.
  - for original Vashwani Transformer $n = 512$
    GPT-2 $n = 1024$
    GPT-3 $n = 2048$
  - is limited by the n^2 complexity of attention (coming soon)

③ Observe the "Language Head" which, to repeat, looks just like output in Assignment 1, Section 3.
  - this is used when training the network to be a language model
  - Consider this sentence being used to create training examples:

The smooth blue lake became choppy in the wind

→ Key: when we want to classify, we chop off this head an put on an MLP classifier head & train with fewer examples
- Just like transfer learning in CNNs the transformer keeps its features

embedding

$\underline{X_0}$ $\underline{X_1}$ $\underline{X_2}$ $\underline{X_3}$ $\underline{X_4}$ ... X_{n-1}

Training Example 1: input The smooth blue [mask] [mask] [mask].
label
output: lake.

as well as many other probs

- model computes P(lake) (& lots of other probs as output)
- (normalized with softmax across all probs) →
- → use "log loss" aka cross-entropy

$$loss = -\log(P(lake))$$

- for a batch of examples of size b, compute the Average loss $\sum_{i=1}^{b}$ loss (training example i)
- side note → "teacher forcing" means always use the {correct / right} answer to compute loss. [which is sort of obvious, rather than using the predicted output]

- there are trillions of examples of writing, unlike most other ML problems
- that writing contains much of human knowledge!

- Note1: the sequence of words input - x_0, x_1, x_2 ... does not contain any information about which word comes where, in order.
  - yet that ordering very much matters.
- so, in addition to the word embeddings, a positional embedding is added to the word embedding [RNNs don't have this problem]

- positional embedding can be a number absolute or relative that is added to the embedding

- Note 2: both the word embeddings themselves and the positional embedding ~~can be~~ are learned (treated as parameters changed through gradient descent back propagation).
  - i.e. don't start with word embeddings

Next Lecture: the details of the Transformer Block beginning with attention

If time, two rants: show Vashwani picture

1. This structure (just one stack of T-blocks), I believe, is the essence of Transformers. The literature is full of Encoder → Decoder structures or encoder-only or decoder-only. This Encoder → decoder structure is based on how RNNs were used to process language and is not necessary for transformers. However, it is the source of lots of confusion. See (*) next page
   [Both BERT & GPT only use one of encoder/decoder]

2. We will dive into what is happening/thought to happen inside the Neural Network of a transformer:
   - it is speculation.
   - the proof of success is always empirical.

That said, it is remarkable what learning to predict the next word can acheive!

4.2.3 TRANSFORMER DECODER (T-D)

We introduce a simple but effective modification to T-ED for long sequences that drops the encoder module (almost reducing model parameters by half for a given hyper-parameter set), combines the input and output sequences into a single "sentence" and is trained as a standard language model.

That is, we convert a sequence-transduction example $(m^1, ..., m^n) \mapsto (y^1, ..., y^\eta)$ into the sentence $(w^1, ..., w^{n+\eta+1}) = (m^1, ..., m^n, \delta, y^1, ..., y^\eta)$, where δ is a special separator token and train a model to predict the next word given the previous ones:

$$p(w^1, ..., w^{n+\eta}) = \prod_{j=1}^{n+\eta} p(w^i | w^1, ..., w^{j-1})$$

Since the model is forced to predict the next token in the input, m, as well as y, error signals are propagated from both input and output time-steps during training. We also suspect that for monolingual text-to-text tasks redundant information is re-learned about language in the encoder and decoder. * We believe this allows for easier optimization and empirically observe this with longer sequences (see Section 5.3). Note that because of the self-attention of the Transformer, when generating the next token, attention from both m and y are considered. At inference we provide the input sequence, m_i, initially, and auto-regressively generate the output, y_i, as normal.

From "Generating Wikipedia by Summarizing Long Sequences"
Liu, Saleh, Pot, Goodrich, Kaiser & Shazeer, ICLR 2018 + Arxiv.