FROM EMBEDDINGS TO LANGUAGE

MARCH 19, 2024
MIKE CAFARELLA

THANKS TO MIKEY SHULMAN
FOR SOME SLIDE IDEAS
The Core Thing

• We need a model that can compute $P(\text{someSentence})$

• We can use it to build a sequence model
  – Input: a sequence of tokens
  – Output: a sequence of tokens

• Lots of applications can be built this way
Translation

INPUT: “J’ai vu un chat noir”

OUTPUT A: “I saw a cat black” vs
OUTPUT B: “I saw a black cat”

Can we compute that \( P(A) < P(B) \)?
Speech Recognition

INPUT: <some waveform>

OUTPUT A: “Yesterday I met Prince Harry”
vs
OUTPUT B: “Yesterday I met prints hairy”

Can we compute that $P(A) > P(B)$?

*The GPT image for “prints hairy” is too weird and alarming to put onscreen*
Naive Text Generation

• If you can compute 
  \( P(\text{nextWord} \mid \text{precedingWords}) \)... 

• \( P(\text{nextWord} \mid "\text{My favorite food is"}) \): 
  – nextWord = “pizza” 
  – nextWord = “love” 
  – nextWord = “antagonist”

• Just try every possible word, pick the best
Computing Sentence Probability

• For a sequence with n words
  \[ w_{n_1} = w_1, w_2, \ldots, w_n \]

• Every word \( w \) is drawn from a fixed vocabulary

• \( P(w_{n_1}) = P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|w_{n-1}) \)

\[ = \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \]
The Simplest Model

- Where do we get evidence for $P(w_k | w_{k-1})$?

- If you want to be really basic, just ignore the context (that is, a k-gram model where $k=1$)

  Or, put another way, assume that
  $P(w_k | w_{k-1}) = P(w_k)$

- Compute as
  $P(w_k) = \frac{\text{Count}(w_k)}{\text{TotalTrainingWords}}$
Expanding Context

• We can do better with more context

• For \( k=2 \) ("bigrams"), we model
  \[
P(w_k|w_{k-1}) = P(w_k|w_{k-1})
  \]

• Count the bigram \( P(w_k|w_{k-1}) \), divide by count of all bigrams starting with \( w_{k-1} \)
  \[
P(w_k|w_{k-1}) = \frac{C(w_{k-1}w_k)}{\sum_{w'} C(w_{k-1}w')}
  \]

• Use special tokens for start/end sentence
Discussion

• Why does a larger corpus help?

• How big does your corpus have to be?

• How could we tell if corpus was too small?

• Why word-grams? Why not characters?
Evaluation

• How can we tell if the model is good?
  – Maybe it helps with a downstream task
  – A general-purpose metric would be nice

• “Perplexity” measures the inverse probability of an unseen test corpus with a particular language model
  – If text is real, then its probability should be high
  – Lower is better

$$PP(W) = P(w_1 \ldots w_N)^{-\frac{1}{N}}$$
Perplexity

• What’s nice about Perplexity?
  – It’s easy to compute
  – You don’t need a concrete task
  – You don’t need to understand the language!

Measures of Perplexity on different language models’ predictions of single-nucleotide sequences from prokaryotic genomes

From Nguyen et al, “Sequence modeling and design from molecular to genome scale with Evo”, 2024
Perplexity

• What’s bad about Perplexity?
  – It only works if your model gives a real probability (so: no rule-based methods)
  – Can’t compare language models with different vocabularies
  – What is a “good” Perplexity number?

• Discussion: is a bigram language model an example of supervised or unsupervised learning?

• Discussion: when might you see overfitting in this setting?
Data, Models, Features

• More context is better
• ...but we will run out of data for statistics when k-grams get big enough

• We need some combination of:
  – More informative features
  – Constrained models to avoid overfitting
• ...but feature engineering for language is extremely hard
• ...and expressivity of the model is hard to engineer
Neural Methods

• Neural methods like CBOW let us avoid feature engineering
• Managing overfitting is poorly-understood but works in practice through model architecture, dropout, and other methods
Encoder-Decoders

• Most sequence models use Encoder-Decoder architecture
  – The encoder converts the input into a compressed embedding-style representation
  – The decoder converts an encoded representation back into the target language

• Nice qualities:
  – You can train them separately
  – You can mix/match them for different input and output types

• word2vec has encoder/decoder architecture
Beyond w2v

• For a chatbot, what’s bad about the w2v encoder/decoder architecture?
Beyond w2v

- (Focus on CBOW for now)
- What’s bad about this for chat?
Weaknesses of w2v for chat

• Input architecture “looks into the future” (this is easy to fix)

• Each word has a single embedding, regardless of usage
  – “I am going to **stick** to it”
  – “I am going to throw the **stick**”
  – The w2v embedding for stick will reflect both senses, even though in some contexts the correct sense is obvious to a human

• Can’t handle truly huge vocabularies

• Sentence modeling is very primitive
BERT, ELMO, and the Transformer

• Ideas in these papers led to incredible improvements in the last 7 years
• We’ll cover these after break