CS 182 Lecture 13: NLP

Unsupervised Pretraining: We have a lot of text data, but mostly unlabeled — can still learn "language representation"

Wide spectrum of "representations"

local, non-contextual ⟵ ⟶ global, contextual

word embeddings | sentence embedding | pretrained LMs.

↳ "meaningful" approach (s.t. similar words are closer in rep): word2vec (low-dim projections)
essentially, "meaning" of word - what words are close to it?

↳ then, can we predict word's neighbors from embedding value? $p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$

u_o: "context word", v_c: "center word", log. res., vocabulary

Train: $\arg\max_{u_1,\dots,u_n, v_1,\dots,v_n} \sum_{c,o} \log p(o|c)$ ← all c,o combinations

Issue: huge vocabulary + really costly to compute all combs

↳ can reframe as binary classification problem. (right/wrong word?)

$p(o \text{ is the right word} | c) = \sigma(u_o^T v_c)$ ← sigmoid, now output is one of 2 classes.

↳ but we also need to push different words apart!

$p(o \text{ is the wrong word} | c) = \sigma(-u_o^T v_c)$ ← randomly chosen "negatives"

word2vec in a nutshell:

$$\arg\max_{u_1,\dots,u_n, v_1,\dots,v_n} \left[\sum_{c,o} \left(\log p(o \text{ is right word} | c) + \sum_w \log p(w \text{ is wrong} | c) \right) \right]$$

$$\left[\sum_{c,o} \left(\log \sigma(u_o^T v_c) + \sum_w \log \sigma(-u_w^T v_c) \right) \right]$$

"push v_c toward u_o AND away from other u_w."

→ can then determine, in theory, real algebraic relations!

→ this idealized word-rep. >>> basic one-hot encoding scheme

Issue w/ word embeddings: vectors do not change given context. "local context"

↳ use pretrained LM's hidden state after training on sentence

"Sesame Street" models: ELMo: bidirectional LSTM model

BERT: huge transformer LM → context-dependent embeddings

ELMo: trains forward + backward LM on same data. "Add" all hidden states.

↳ "Embeddings from Language Models"

↳ simple: $ELMo_t = [h^{fwd}_{t,last}, h^{bwd}_{t,last}]$

large corpus of unlabeled text.

IN PAPER: last = 2 (lol)

↳ complex: $ELMo_t = \gamma \sum_{i=1}^{L} w_i [h^{fwd}_{t,i}, h^{bwd}_{t,i}]$ (learned)

→ for a downstream app: feed existing rep + ELMo rep to model (after running ELMo).

pros: context-specific, semantically meaningful (very)

cons: computationally expensive, storage issue.

really good at generation (GPT et al.)

BERT (industry-leading) → "Bidirectional Encoder Representations from Transformers"

↳ basically, ELMo that is purely attention-based.

Base model for BERT: decoder-only transformer (specified direction)

"We can do better": full self-attention layers (no mask), no input-output shift.

but randomly mask out some tokens → so some words are 1:1, but model cannot directly overfit.

↳ keeps model actively "on its toes"

fill in the blanks.

TLDR: encoder-only transformer w/ input mask.

Training BERT: pass in pairs of sentences (Q/A, machine translation, etc.).

↳ apply 15% mask

↳ coin flip A/B or B/A sentence order

→ very first token: trained on binary classifier (does A follow B or B follow A)

Using BERT (quite flexible): for classification, 1. replace A/B classifier w/ desired cross-entropy loss

2. finetune end-to-end

for per-position, - SQuAD: highlight span of paragraph - NER: output class for each token } change loss for each token

→ can also be used to extract representations (like ELMo).

↳ test config: concat last k hidden state layers.