CS 378 Lecture 15: Language Modeling
n-grams, RNNs

Announcements
- Midterm back soon (Mon?)
- A4 released Tues
- Vicente Ordóñez talk Fri 11am 6.302
- Custom FP proposals due in one week (email to me)

Today
- Intro to language modeling
- N-gram LMs
- Neural LMs
- RNNs (brief!)
- Transformers on Tuesday
Recap (course so far)

Input

\[ \text{text} \xrightarrow{\text{classify (sentiment)}} \text{POS} \xrightarrow{\text{syntactic parsing}} \text{text} \xrightarrow{\text{label/structure}} \]

Next few weeks:

\[ \text{text} \rightarrow \text{text (seq2seq)} \]

machine translation, dialogue systems, summarization, QA, ...

Today

Language Modeling

"autocomplete" / predictive text

RNNs / Transformers
LM: a distribution $P(\overline{w})$ over sequences of words

Assign high prob. to real, grammatical, natural sentences

Grammatical error correction:

$\overline{w}$, try changing some function words, find $\overline{w'}$

$P(\overline{w'}) > P(\overline{w}) \Rightarrow$ we fixed some errors?

N-gram language modeling

$\overline{w} = (w_1, \ldots, w_m)$ m words
By the chain rule of probability:

\[
P(\overline{w}) = P(w_1) P(w_2 | w_1) P(w_3 | w_1 w_2) \\
\text{no assumptions - } P(w_n | w_1 \ldots w_{n-1})
\]

\[
P(\overline{w}) = P(w_1) P(w_2 | w_1) P(w_3 | w_2)
\]

2-gram LM

n-gram LM: depend on past \(n-1\) words

\[
P(\overline{w}) = \prod_{i=1}^{n} P(w_i | w_{i-n+1}, \ldots, w_{i-1})
\]

\(n=4: w_{i-3} w_{i-2} w_{i-1}\)
the cat ran

2-gram LM: start of sequence

\[ P_2(w) = P(\text{the } | <s>) \cdot P(\text{cat} | \text{the}) \cdot P(\text{ran} | \text{cat}) \cdot P(\text{STOP} | \text{ran}) \]

3-gram

\[ P_3(w) = P(\text{the } | <s> <s>) \cdot P(\text{cat} | <s> \text{ the}) \cdot P(\text{ran} | \text{the cat}) \]

Ex

I saw the dog — lots of completions

\[ \text{yesterday} \quad \text{bark} \quad \text{owner} \]

\[ \text{sleep} \quad \text{store} \]

\[ \text{in} \quad \text{at} \]
The capital of Texas is ___.

fewer completions

trickier!

24 words

Austin

far

hot

LM is a very fundamental task

= N-gram parameterization ∨ vocab

Like HMM transitions: lookup table

1v1 wi

2-gram

the

P(dog|the)

P(cat|the)

wi-1

P(-1 cup of TX is)
Sparse: not all 3-grams exist
actual size $< |V|^3$

$n \sim 7$ is about as high as you can go
Parameters: count + normalize over a big corpus

the cat  \[ P(\text{cat} \mid \text{the}) = \frac{1}{2} \]
the cat  \[ P(\text{snake} \mid \text{the}) = \frac{1}{4} \]

Ex

\( N = 5 \), I hate to go to Maui

\[ P(\text{Maui} \mid \text{hate to go to}) = 0 \]

\[ \frac{\text{count}(\text{hate to go to Maui})}{\text{count}(\text{hate to go to})} \]

\[ n = 5 \]
Smoothing

\[ P_s(w_i | w_{i-4} w_{i-3} w_{i-2} w_{i-1}) = \lambda \cdot P_{raw}^s(w_i | w_{i-4} w_{i-3} w_{i-2} w_{i-1}) \]

\[ + (1-\lambda) P_y(w_i | w_{i-3} w_{i-2} w_{i-1}) \]

\[ \lambda = 0.9 \]

\[ P_y = \lambda \cdot P_{raw}^y + (1-\lambda) P_3 \]  recursive

\[ P_i = P_{raw}(w_i) > 0 \text{ any } w_i \in \mathcal{V} \text{ is seen 21 time} \]
What do we store:

\[ P_{\text{raw}} \sim \mathcal{V}^{15} \quad p_{\text{raw}}, P_{\text{raw}}, P_{\text{raw}}, P_{\text{raw}} \]

Counts are sparse
Distribution is not sparse

Many smoothing schemes

Absolute discounting: sets the \( \lambda \) dynamically

Kneser-Ney smoothing
Neural Language Modeling

\[ P(w_i|w_{i-1}, \ldots, w_1) \] model w/ a NN

\[ P(w_6|\ldots): \quad \text{DAN} \xrightarrow{\text{embedding}} \quad \text{FFNN} \rightarrow P(w_6|\ldots) \]

Con: ignore ordering

\[ \text{FFNN} \rightarrow \text{concat} \]
Cons: require weights for every word in every position

The movie was —

My favorite movie

May be hard to scale/different sizes of input

Two solutions:

① RNN: process sequences "uniformly"

② Transformer: halfway between DAN + concat
RNN will maintain a hidden state \( \overline{h} \) after processing \( n \) words.

\[ \overline{h}_n = \text{neural net}(w_1, \ldots, w_n) \]

\[ P(w_{n+1} | w_1, \ldots, w_n) = \text{softmax}(W \overline{h}_n) \]

\( W \): \( |V| \times d \) matrix

\( \overline{h} \): \( d \)-dimensional vector

\[ \overline{h}_{n+1} = \text{neural net} \left( \overline{h}_n, w_{n+1} \right) \]

easy to update
RNN encodes a sequence of vecs. into a single vector

$$\begin{align*}
\overline{h}_i &= \tanh (W \overline{w}_i + V \overline{h}_{i-1}) \\
\overline{y}_i &= \tanh (U \overline{h}_i)
\end{align*}$$

$$P(w_4 | w_{i-3}) = \text{softmax}(X \overline{h}_3)$$

Elman network:
Training:

$$P(w_t|w_{t-3})$$

loss

backprop

updates $$V, W, U$$

update term from every cell

Elman network

LSTM