Machine Learning for Natural Language Processing

Language Modeling

Lecture 4

Benjamin Muller

INRIA Paris - ALMANACH
benjamin.muller@inria.fr

Lecture Outline

- Language Model
- Estimation of Language Model

Language Model

Language Modeling

- What is a Language Model ?
- Modeling language with n-grams

Language modeling

What is language modeling?

- **Language modeling** corresponds to assigning a probability to a text
- A text is a sequence of tokens, or characters
- Tokens can be words, sub-words,
- For example:

$$\{\text{a cat}\} = \{\text{a}, \text{cat}\},$$

What is language modeling?

- **Language modeling** corresponds to assigning a probability to a text
- A text is a sequence of tokens, or characters
- Tokens can be words, sub-words,
- For example:

$$\begin{aligned} \{\text{a cat}\} &= \{\text{a}, \text{cat}\}, \\ &= \{\text{a}, \ , \text{c}, \text{a}, \text{t}\}, \end{aligned}$$

What is language modeling?

- **Language modeling** corresponds to assigning a probability to a text
- A text is a sequence of tokens, or characters
- Tokens can be words, sub-words,
- For example:

$$
\begin{aligned}
\{\text{a cat}\} &= \{\text{a}, \text{cat}\}, \\
&= \{\text{a}, \ , \text{c}, \text{a}, \text{t}\}, \\
&= \{\text{a}, \ , \text{ca}, \text{t}\}.
\end{aligned}
$$

What is language modeling?

- Given a sequence $\{w_1, \ldots, w_T\}$ of tokens, a language model estimates its probability:

$$P(w_1, \ldots, w_T)$$

- P depends on a **vocabulary**, i.e., the set of unique tokens.

- Question: How to estimate P ?

What is language modeling?

- Given a sequence $\{w_1, \ldots, w_T\}$ of tokens, a language model estimates its probability:

$$P(w_1, \ldots, w_T)$$

- P depends on a **vocabulary**, i.e., the set of unique tokens.

- Question: How to estimate P ?

Language Models

- Causal Language Model
- Mask Language Model

Applications of language modeling

Language models are applied in several fields:

- Speech recognition:

$$P(\text{"Vanilla, I scream"}) < P(\text{"Vanilla ice cream"}).$$

- Machine translation:

$$\begin{aligned} P(\text{"Déçu en bien"} \mid \text{"Pleasantly surprised"}) &< \\ P(\text{"Agréablement surpris"} \mid \text{"Pleasantly surprised"}) & \end{aligned}$$

- Optical Character Recognition:

$$P(\text{"m0ve fast"}) < P(\text{"move fast"})$$

Probabilistic *Causal* language model

- Sequence probability as a product of token probabilities:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, \ldots, w_1)$$

Probabilistic *Causal* language model

- Sequence probability as a product of token probabilities:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, \ldots, w_1)$$

- Indeed we have:

$$P(a, b) = P(a)P(b \mid a)$$

Probabilistic *Causal* language model

- Sequence probability as a product of token probabilities:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, \ldots, w_1)$$

- Indeed we have:

$$P(a, b) = P(a)P(b \mid a)$$

- Recursively applied to a sequence:

$$\begin{aligned} P(w_1, w_2, w_3) &= P(w_1)P(w_2, w_3 \mid w_1) \\ &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2, w_1). \end{aligned}$$

Probabilistic *Causal* language model

- Sequence probability as a product of token probabilities:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, \ldots, w_1)$$

- Indeed we have:

$$P(a, b) = P(a)P(b \mid a)$$

- Recursively applied to a sequence:

$$\begin{aligned} P(w_1, w_2, w_3) &= P(w_1)P(w_2, w_3 \mid w_1) \\ &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2, w_1). \end{aligned}$$

- Causal Language models estimate probability of upcoming token given past:

$$P(w_t \mid w_{t-1}, \ldots, w_1).$$

Estimating Language Models

- Causal Language Model
- Mask Language Model

Mask Language Model [1,2]

Sentence The cat is drinking milk in the kitchen

[1] Devlin et al. (2018)
[2] also referred as Cloze Task

Mask Language Model [1,2]

Sentence	The cat is drinking milk in the kitchen
input	The cat <MASK> drinking <MASK> in the kitchen

- Randomly replace 15% of words in sentence with a <MASK> token

[1] Devlin et al. (2018)
[2] also referred as Cloze Task

Mask Language Model [1][2]

Sentence The cat is drinking milk in the kitchen

input The cat <MASK> drinking <MASK> in the kitchen

targets {"is", "milk"}

- Randomly replace 15% of words in sentence with a <MASK> token
- Take the masked words as targets for the model to predict

[1] Devlin et al. (2018)

[2] also referred as Cloze Task

Mask Language Model [1,2]

Sentence	The cat is drinking milk in the kitchen
input	The cat mushroom drinking shoes in the kitchen
targets	{"is", "milk"}

- Randomly replace 15% of words in sentence with a <MASK> token
- Take the masked words as targets for the model to predict
- Extension: use random words from vocabulary instead of <MASK>

[1] Devlin et al. (2018)
[2] also referred as Cloze Task

Mask language model

Masked Language Modeling estimates the probability of sequence tokens of length T with:

$$P(w_i|w_1, .., w_{i-1}, w_i, .., w_T)$$

Language Models in a nutshell

- a Language Model is a model that predicts a **token** based on its surrounding linguistic **context**
- Tokens can be words, sub-words or characters
- Context can be the *left sequence*, *left and right sequence*, the sentence, a window around the words, the paragraph...
- We saw two way of defining langvge models: Causal Language Model and Mask Language Model

Estimating language models

Estimating language models

- Statistical approach: N-Gram model
- Neural Language Models
-
  - Recurrent Neural Networks (LSTM)
  - The Transformer Architecture

Count based language model

- Example:

$$
\begin{aligned}
P(\text{English} \mid \text{The moment one learns}) &= \frac{c\,(\text{The moment one learns English})}{c\,(\text{The moment one learns})} \\
&= \frac{35}{73} = 0.48
\end{aligned}
$$

Sentence “The moment one learns English” appears 35 in dataset
Sentence “The moment one learns” appears 75 in dataset

Limitiations of count based language model

- Number of unique sentences increases with dataset size,
- Long sentences are rare: no good statistics for them

$\rightarrow$ Too many sentences with not enough statistics
(Sparsity due to combinatorial structure of language)

Count based language model

- **Solution** truncate past to a fixed size window
- For example:

$$P(\text{English} \mid \text{The moment one learns}) \approx P(\text{English} \mid \text{one learns})$$

- Implicit assumption: most important information about a word is in its recent history
- **Beware!** In general:

$$P(w_1, \ldots, w_T) \neq \prod_{t=1}^{T} P(w_t \mid w_{t-1}, \ldots, w_{t-n+1})$$

Count based language model

- **Truncated count based models = *n*-gram models**
- "n" refers to the size of past
- Examples:
  - Unigram:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t)$$

  - Bigram:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1})$$

Count based language model: unigram

- Probability of a sentence with a unigram model:

$$P_U(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t) = \prod_{t=1}^{T} \frac{c(w_t)}{N}$$

N = total number of tokens in dataset
$c(w_t)$ = number of occurences of w_t in dataset

- Unigram only uses word frequency

- Example of text generation with this model:

the or is ball then car

Count based language model: bigram

- Probability of a sentence with a bigram model:

$$P_U(w_1, \dots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}) = \prod_{t=1}^{T} \frac{c(w_{t-1} w_t)}{c(w_{t-1})}$$

$c(w_{t-1} w_t) =$ number of occurences of sequence $w_{t-1} w_t$

- Predict a word just with the previous word

Count based language model: bigram

- Example of text generation with bigram model:

new car parking lot of the

- “car” is generated from “new”, “parking” from “car”...
- But “new” has no influence on “parking”

Count based language model

- Simple to extend to longer dependencies: trigrams, 4-grams...
- *n*-grams can be "good enough" in some cases
- But *n*-grams cannot capture long term dependencies required to truely model language

Estimating *n*-gram probabilites: an example

- bigram:

$$P(w_t \mid w_{t-1}) = \frac{c(w_{t-1} w_t)}{c(w_{t-1})}$$

- Dataset:

$<s>$*we sat in the house*
$<s>$*we sat here we two and we said*
$<s>$*how we wish we had something to do*

- Extract some probabilities:

$$P(sat \mid we) = 0.33,\; P(wish \mid we) = 0.17,\; P(in \mid sat) = 0.5$$

- $<$s$>=$ token for beginning of sentence; $P(<\text{s}>) = 1$.
- Compute sentence probability with them

Estimating n-gram probabilites: an example

- Extract count from Berkeley Restaurant dataset (9222 sentences)
- Unigram counts:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Bigram counts:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Estimating n-gram probabilites: an example

- The bigram probabilities are obtained by dividing the bigram counts with the unigram counts:

$$P(w_2 \mid w_1) = \frac{c(w_1 w_2)}{c(w_1)}$$

- Resulting bigram probabilities:

	i	want	to	eat	chinese	food	lunch	spend
i	0.022	0.33	0	0.036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Estimating n-gram probabilites: an example

- Example:

$$P(\text{<s> i want chinese food})?$$

$\text{<s>} =$ token for beginning of sentence; $P(\text{<s>}) = 1$.

- Result:

$$\begin{aligned} P(\text{<s> i want chinese food}) =& P(\text{<s>})P(\text{i}|\text{<s>})P(\text{want}|\text{i})P(\text{chinese}|\text{want})P(\text{food}|\text{chinese}) \\ =& 1 \times .25 \times 0.33 \times 0.0065 \times 0.52 \\ =& 0.00027885 \end{aligned}$$

Estimating *n*-gram probabilites: an example

	i	want	to	eat	chinese	food	lunch	spend
i	0.022	0.33	0	0.036	0	0	0	0.00079
...								
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- Example:

$$P(\text{<s> i bring my lunch to work})?$$

- Result:

$$\begin{aligned} P(\text{<s> i bring my lunch to work}) &= P(\text{<s>}) \dots P(\text{to}|\text{lunch}) \dots \\ &= 1 \times \cdots \times 0 \times \dots \\ &= \mathbf{0} \end{aligned}$$

- **Does not generalize well!**

Good-Turing estimation

- **Idea** reallocate probability mass of n-grams that occur exactly $c + 1$ times to n-grams that occur exactly c times

- reallocate mass of n-grams appearing once to unseen n-grams

$\rightarrow$ **alternative to Add-1**

Good-Turing estimation

- **Idea** reallocate probability mass of n-grams that occur exactly $c+1$ times to n-grams that occur exactly c times
- reallocate mass of n-grams appearing once to unseen n-grams

→ **alternative to Add-1**

- the adjusted count:

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

where N_c is the number of n-grams that appears exactly c times

Good-Turing estimation

- **Idea** reallocate probability mass of *n*-grams that occur exactly $c+1$ times to *n*-grams that occur exactly c times
- reallocate mass of *n*-grams appearing once to unseen *n*-grams

→ **alternative to Add-1**

- the adjusted count:

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

  where N_c is the number of *n*-grams that appears exactly c times

- *n*-gram probability depends on c^* instead of c

Good-Turing estimation

- **Idea** reallocate probability mass of n-grams that occur exactly $c+1$ times to n-grams that occur exactly c times

- reallocate mass of n-grams appearing once to unseen n-grams

$\rightarrow$ **alternative to Add-1**

- the adjusted count:

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

  where N_c is the number of n-grams that appears exactly c times

- n-gram probability depends on c^* instead of c

- **Problem** What if $N_{c+1} = 0$ (but $N_c > 0$)?

Backoff and Interpolation

- If no good statistics on long context: use shorter context
- **Backoff**: use trigram if enough data, else backoff to bigram.
- **Interpolation**: mix statistics of trigram, bigram and unigram.

Pros and Cons of N-Gram Language Models

Pros

- Fast at training and inference
- Can reach good accuracy if lots of data

Cons

- Impossible to model very long dependencies (simplistic assumptions done)
- Generalization limited
- Not Deep Learning compatible

References I

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.