

# Language Modeling in Automatic Speech Recognition

## ZRE lecture

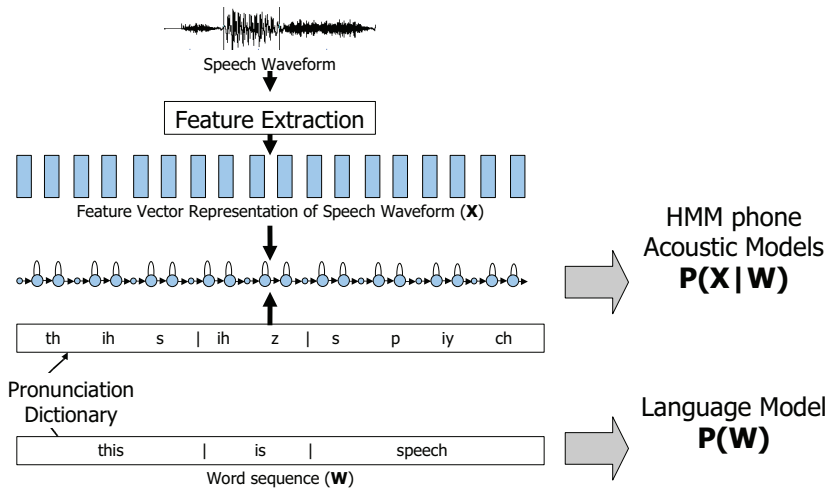
Karel Beneš

03. 04. 2023

# Roadmap of the Lecture

- 1 ASR Intro
- 2 Count-based Language Modeling
- 3 Neural-network Language Modeling

# Automatic speech recognition



- Determine the most probable word sequence  $\tilde{W}$  given the observed acoustic signal  $Y$

$$\tilde{W} = \operatorname{argmax} P(W|Y) = \frac{\operatorname{argmax} P(W)P(Y|W)}{P(Y)}$$

- Search for word sequence  $\tilde{W}$  that maximizes  $P(W)$  and  $P(Y|W)$ 
  - $P(W)$  = language model (likelihood of word sequence)
  - $P(Y|W)$  = acoustic model  
(likelihood of observed acoustic signal given word sequence)

# Roadmap of the Lecture

- 1 ASR Intro
- 2 Count-based Language Modeling
- 3 Neural-network Language Modeling

Estimate the probability of a word sequence  $W$ :

$$P(W) = P(w_1, w_2, \dots, w_N) \quad (1)$$

$$P(W) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdot \dots \cdot P(w_N|w_1, \dots, w_{N-1}) \quad (2)$$

# Markov assumption, n-grams counting

Approximate by only considering 1-word history, e.g.:

$$P(W) \approx P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdot \dots \cdot P(w_N|w_{N-1}) \quad (3)$$

Gives rise to *2-gram / bigram* language model. Probabilities  $P(w_a|w_b)$  are estimated as:

$$P(w_a|w_b) = \frac{C(w_b, w_a)}{C(w_b)} \quad (4)$$

# Markov assumption, n-grams counting

Approximate by only considering 2-word history, e.g.:

$$P(W) \approx P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdot \dots \cdot P(w_N|w_{N-2}, w_{N-1}) \quad (5)$$

Gives raise to 3-gram / trigram language model. Probabilities  $P(w_a|w_b, w_c)$  are estimated as:

$$P(w_a|w_b, w_c) = \frac{C(w_b, w_c, w_a)}{C(w_b, w_c)} \quad (6)$$



- 1 limited context, e.g.

“... to Paris, we couldn't wait to see the ??? ”

Unfortunate, but acceptable.

- 2 unseen n-grams, e.g. 2M words from Wikipedia -> 2.6 % of unseen words in other wiki texts. Cca 1/3 of trigrams not seen since 1970s. So called *Curse of Dimensionality*, severe.

If we have enough data,  $C(w_{i-n+1}, \dots, w_{i-1}, w_i) > k$ :

$$P(w_i | w_{i-n+1}, \dots, w_{i-1}) = d_{C(w_{i-n+1}, \dots, w_{i-1}, w_i)} \frac{C(w_{i-n+1}, \dots, w_{i-1}, w_i)}{C(w_{i-n+1}, \dots, w_{i-1})} \quad (7)$$

The discount factor  $d_{C(\cdot)}$

Determines how much do we take away from n-grams appearing  $C(\cdot)$  times; estimated typically with Good-Turing algorithm.

If  $C(w_{i-n+1}, \dots, w_{i-1}, w_i) \leq k$ :

$$P(w_i | w_{i-n+1}, \dots, w_{i-1}) = \alpha_{w_{i-n+1}, \dots, w_{i-1}} P(w_i | w_{i-n+2}, \dots, w_{i-1}) \quad (8)$$

The redistribution factor  $\alpha$

Determines how much of discounted probability is assigned to the (n-1)-gram  $w_{i-n+1}, \dots, w_{i-1}$ .

The threshold  $k$  for “enough” can in practice be set to 0.

For bigrams:

$$P(w_i | w_{i-1}) = \frac{\max(C(w_{i-1} w_i) - \delta, 0)}{C(w_{i-1})} + \lambda \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)} \quad (9)$$

The mysterious  $N_{1+}(\cdot)$  just counts how many unique bigrams have we seen:

$$N_{1+}(\bullet w_i) = |\{w_{i-1} | C(w_{i-1}, w_i)\}| \quad (10)$$

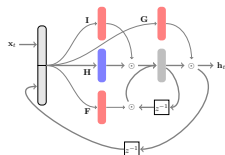
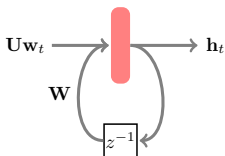
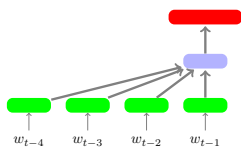
$$N_{1+}(\bullet\bullet) = |\{w_{i-1}, w_i | C(w_{i-1}, w_i)\}| \quad (11)$$

So effectively, we only believe the unigram  $w_i$ , if we have seen it in many different contexts.

# Roadmap of the Lecture

- 1 ASR Intro
- 2 Count-based Language Modeling
- 3 Neural-network Language Modeling**

# Battling the Curse of Dimensionality using NNs



- **Embeddings**
- The actual architecture
- Softmax output

Only the neural magic in the middle changes!

- LMs help recognition
- Effective LM can be based on simple counts . . .
- . . . but smoothing is needed. When in doubt, pick Kneser-Ney.
- Neural networks pay with compute to reduce curse of dimensionality
- Commonly, LSTMs are used; in many variants (incl. GRUs)
- Transformers taking over, but more expensive
  - Linearly increasing size of hidden state
  - Linearly increasing cost of each step (=> quadratic overall)