Guest Lecture

Representation Learning for Grounded Special Reasoning – Michael Jenner

Lecture 2 – Topic Models

Topic for today: Language Models (LM)

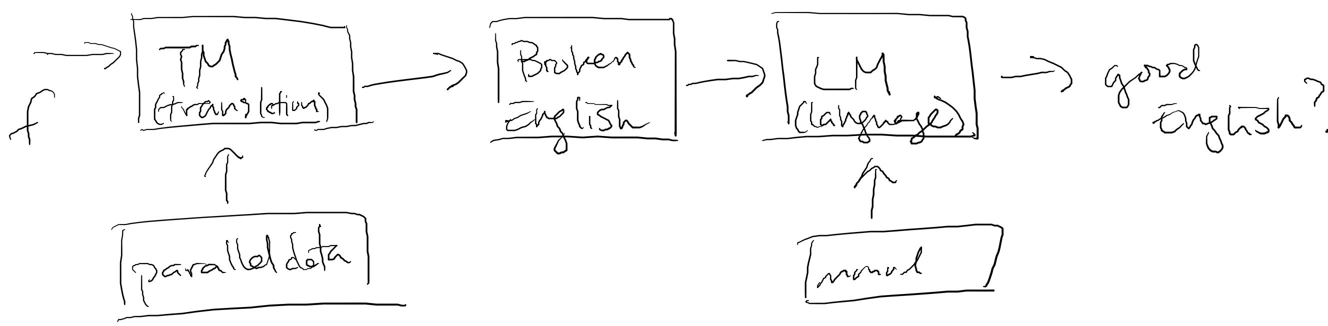
1. Applications of LM
    a. Noisy-channel models
2. Definition of N-gram models (particular type of LM)
    a. Decompositionality
3. Evaluation: Perplexity
4. Smoothing

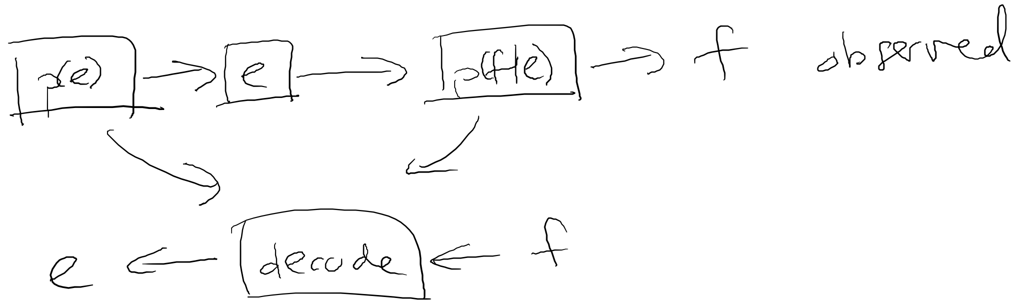
Activity: Guess the Author (refer to slides)

- Someone guesses machine correctly
- Output was from model trained with Shakespearean text

Definition of LM

- Assume we are given a finite vocabulary $w_L \in V$
- Ask $p(w_1, w_2, \ldots., w_M) = ?$
- Statistical Machine Translation (Stat. MT):





- Mathematical Description: $argmax_e P(e|f) = argmax_e \frac{P(f|e)P(e)}{P(f)} = argmax_e P(f|e)P(e)$

Definition of N-gram models (particular type of LM)

- Given $w_L \in V$
- Want to find $p(w_1, w_2, \dots., w_M)$
  - Chain Rule: $p(w_1, w_2, \dots., w_M) = \prod p(w_i|w_1, \dots, w_{i-1})$
  - Markov Assumption: $\prod p(w_i|w_1, \dots, w_{i-1}) \sim \prod p(w_L|w_{L-1}, \dots, w_{L-k})$
    - (current word depends on only last k words, not all past words)
- Example: $p(I\ love\ Arya)$
  - Unigram Model: $p(I)p(love)p(Arya)$
  - Bigram Model: $p(love|I)p(arya|love)$
- Usually have START and END tokens
- Parameterization (θ)
  - Bigram Model: $p(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})} \rightarrow |V|^2$ parameters
- Steps for Training
  - Collect training set X
  - Estimate parameters θ
  - Test on unseen X'
- N-gram model; large or small N?
  - Small N – may not be enough information
  - Large N – hard to train, data is sparse
  - Turns out that for many applications 3 or 4 is good enough
    - Guess the Author was 3-gram model
  - Will see methods in the future for increasing N without cons
- How much entropy/ambiguity in English/languages?

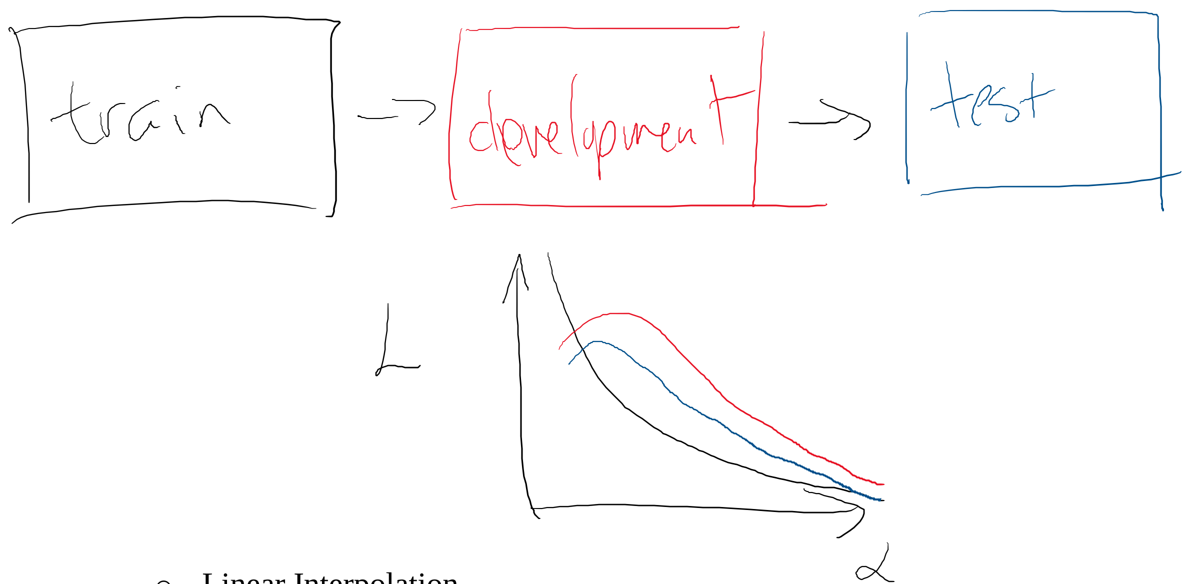
Evaluation

- Test set $X' = x_1, \dots, x_n$
- Look at $p(X'|\theta) = \prod_i p(x_i|\theta)$
- In machine learning, often look at average log likelihood: $l = \frac{1}{|X|} \sum \log p(x_i|\theta)$
- But in NLP, we look at perplexity 2^{-l} which satisfies $2^{-l} \in [1, \infty)$
  - Suppose $\forall w, p(w) = \frac{1}{|V|}$. Then perplexity must be $|V|$
  - Lower perplexity does not necessarily mean better performance in Z application!

Smoothing

- Bi-gram model results in sparse data – how to deal with this? Smoothing.
- Zipf's Law – observation about frequency of words (refer to slides)
- Pseudoscience on Indus Script – many things satisfy Zipf's law, not just languages

- Smooth sparse data distribution by 'redistributing the wealth'
    - Add α (Laplacian)
        - Assigns a non-zero prior to all words
        - $p(w) = \frac{count(w)+\alpha}{\sum_i(count(w_i)+\alpha)}$, $p(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i)+p(w_i)}{\sum_i(count(w_{i-1},w)+p(w_i))}$
        - How to determine hyperparameter α?



    - Linear Interpolation
        - Combines information from unigrams, bigrams, …, and N-grams
        - $p(w_i|w_{i-2},w_{i-1}) = \lambda_1 p(w_i|w_{i-2},w_{i-1}) + \lambda_2 p(w_i|w_{i-1}) + \lambda_3 p(w_i)$
        - $\sum \lambda_i = 1, \lambda_i \geq 0$