CS 371R: IR and Web Search: Language Models

Raymond J. Mooney
University of Texas at Austin

Language Models

- Formal grammars (e.g. regular, context free) give a hard "binary" model of the legal sentences in a language.
- For NLP, a ***probabilistic*** model of a language that gives a probability that a string is a member of a language is more useful.
- To specify a correct probability distribution, the probability of all sentences in a language must sum to 1.

Uses of Language Models

- Speech recognition
  - “I ate a cherry” is a more likely sentence than “Eye eight uh Jerry”
- OCR & Handwriting recognition
  - More probable sentences are more likely correct readings.
- Machine translation
  - More likely sentences are probably better translations.
- Generation
  - More likely sentences are probably better NL generations.
- Context sensitive spelling correction
  - “Their are problems wit this sentence.”

Completion Prediction

- A language model also supports predicting the completion of a sentence.
  - Please turn off your cell _____
  - Your program does not ______
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

N-Gram Models

- Estimate probability of each word given prior context.
  - P(phone | Please turn off your cell)
- Number of parameters required grows exponentially with the number of words of prior context.
- An N-gram model uses only N–1 words of prior context.
  - Unigram: P(phone)
  - Bigram: P(phone | cell)
  - Trigram: P(phone | your cell)
- The ***Markov assumption*** is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a ***kth-order Markov model***, the next state only depends on the *k* most recent states, therefore an N-gram model is a (N–1)-order Markov model.

N-Gram Model Formulas

- Word sequences

$$w_1^n = w_1 \ldots w_n$$

- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2) \ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- Bigram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

- N-gram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

Estimating Probabilities

- N-gram conditional probabilities can be estimated from raw text based on the ***relative frequency*** of word sequences.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

- To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words.

Generative Model & MLE

- An N-gram model can be seen as a probabilistic automata for generating sentences.

  Initialize sentence with N–1 <s> symbols
  Until </s> is generated do:
  Stochastically pick the next word based on the conditional probability of each word given the previous N –1 words.

- Relative frequency estimates can be proven to be ***maximum likelihood estimates*** (MLE) since they maximize the probability that the model *M* will generate the training corpus *T*.

$$\hat{\lambda} = \operatorname*{argmax}_{\lambda} P(T \mid M(\lambda))$$

Example from NLP Textbook

- P(<s> i want english food </s>)
  = P(i | <s>) P(want | i) P(english | want)
  P(food | english) P(</s> | food)
  = .25 x .33 x .0011 x .5 x .68 = .000031
- P(<s> i want chinese food </s>)
  = P(i | <s>) P(want | i) P(chinese | want)
  P(food | chinese) P(</s> | food)
  = .25 x .33 x .0065 x .52 x .68 = .00019

Laplace (Add-One) Smoothing

- "Hallucinate" additional training data in which each possible N-gram occurs exactly once and adjust estimates accordingly.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)+1}{C(w_{n-N+1}^{n-1})+V}$$

where V is the total number of possible (N−1)-grams (i.e. the vocabulary size for a bigram model).

- Tends to reassign too much mass to unseen events, so can be adjusted to add $0<\delta<1$ (normalized by δV instead of V).

Advanced Smoothing

- Many advanced techniques have been developed to improve smoothing for language models.
  - Good-Turing
  - Interpolation
  - Backoff
  - Kneser-Ney
  - Class-based (cluster) N-grams

A Problem for N-Grams: Long Distance Dependencies

- Many times local context does not provide the most useful predictive clues, which instead are provided by ***long-distance dependencies***.
  - Syntactic dependencies
    - "The ***man*** next to the large oak tree near the grocery store on the corner **is** tall."
    - "The ***men*** next to the large oak tree near the grocery store on the corner **are** tall."
  - Semantic dependencies
    - "The ***bird*** next to the large oak tree near the grocery store on the corner **flies** rapidly."
    - "The ***man*** next to the large oak tree near the grocery store on the corner **talks** rapidly."
- More complex models of language are needed to handle such dependencies.

Summary

- Language models assign a probability that a sentence is a legal string in a language.
- They are useful as a component of many NLP systems, such as ASR, OCR, and MT.
- Simple N-gram models are easy to train on unsupervised corpora and can provide useful estimates of sentence likelihood.
- MLE gives inaccurate parameters for models trained on sparse data.
- Smoothing techniques adjust parameter estimates to account for unseen (but not impossible) events.