Natural Language Processing CSCI 4152/6509 — Lecture 14 N-gram Model and Smoothing

Instructors: Vlado Keselj
Time and date: 16:05 – 17:25, 19-Oct-2023
Location: Rowe 1011

Previous Lecture

- Naïve Bayes classification model (continued)
  - Spam detection example
  - Computational tasks
  - Number of parameters
  - pros and cons, additional notes
  - Bernoulli and Multinomial Naïve Bayes
- N-gram model
  - Language modeling
  - N-gram model assumption

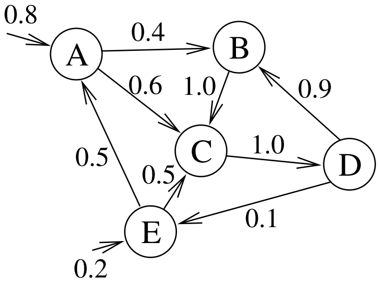
N-gram Model as a Markov Chain

- N-gram Model is very similar to Markov Chain Model
- Markov Chain consists of
  - sequence of variables V_1, V_2, ...
  - probability of V_1 is independent
  - each next variable is dependent only on the previous variable: V_2 on V_1, V_3 on V_2, etc.
  - Conditional Probability Tables: $P(V_1)$, $P(V_2|V_1)$, ...
- Markov Chain is identical to bi-gram model, but higher-order n-gram models are very similar as well

Markov Chain: Formal Definition

- *Stochastic process* is a family of variables $\{V_i\}$ $i \in I$, $\{V_i, i \in I\}$, or $\{V_t, t \in T\}$
- *Markov process:* for any t, and given V_t, the values of V_s, where $s > t$, do not depend on values of V_u, where $u < t$.
- If I is finite or countably infinite: V_i depends only on V_{i-1}
- In this case Markov process is called *Markov chain*
- Markov chain over a finite domain can be represented using a DFA (Deterministic Finite Automaton)

Markov Chain: Example



This model could generate the sequence $\{A, C, D, B, C\}$ of length 5 with probability:

$$0.8 \cdot 0.6 \cdot 1.0 \cdot 0.9 \cdot 1.0 = 0.432$$

assuming that we are modelling sequences of this length.

Evaluating Language Models: Perplexity

- Evaluation of language model: extrinsic and intrinsic
- Extrinsic: model embedded in application
- Intrinsic: direct evaluation using a measure
- Perplexity, W — text, $L = |W|$,

$$\mathrm{PP}(W) = \sqrt[L]{\frac{1}{P(W)}} = \sqrt[L]{\prod_i \frac{1}{P(w_i|w_{i-n+1}\ldots w_{i-1})}}$$

- Weighted average branching factor

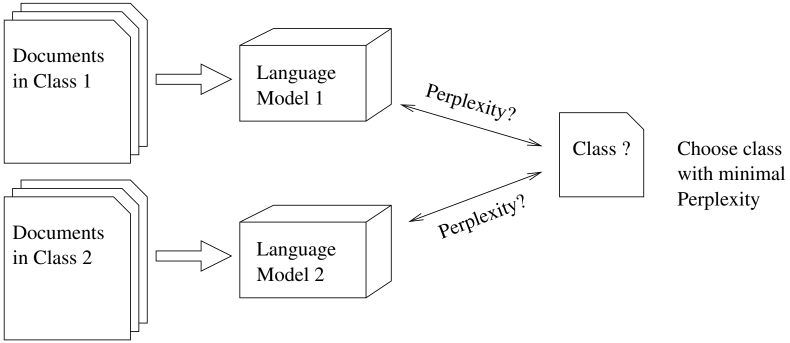
Use of Language Modeling in Classification

- Perplexity, W — text, $L = |W|$,

$$\mathrm{PP}(W) = \sqrt[L]{\frac{1}{P(W)}} = \sqrt[L]{\prod_i \frac{1}{P(w_i|w_{i-n+1}\ldots w_{i-1})}}$$

- Text classification using language models

Classification using Language Modeling



Unigram Model and Multinomial Naïve Bayes

- It is interesting that classification using Unigram Language Model is same as Multinomial Naïve Bayes with all words

N-gram Model Smoothing

- Smoothing is used to avoid probability 0 due to sparse data
- Some smoothing methods:
  - Add-one smoothing (Laplace smoothing)
  - Witten-Bell smoothing
  - Good-Turing smoothing
  - Kneser-Ney smoothing (new edition of [JM])

Example: Character Unigram Probabilities

- Training example: `mississippi`
- What are letter unigram probabilities?
- What would be probability of the word '`river`' based on this model?

Unigram Probabilities: `mississippi`

Add-one Smoothing (Laplace Smoothing)

- Idea: Start with count 1 for all events
- $|V|$ = vocabulary size (unique tokens)
- n = length of text in tokens
- Smoothed unigram probabilities:

$$P(w) = \frac{\#(w) + 1}{n + |V|}$$

- Smoothed bi-gram probabilities

$$P(a|b) = \frac{\#(ba) + 1}{\#(b) + |V|}$$

Mississippi Example: Add-one Smoothing

- Let us again consider the example trained on the word: `mississippi`
- What are letter unigram probabilities with add-one smoothing?
- What is the probability of: `river`

Mississippi Example: Add-one Smoothing

Witten-Bell Discounting

- Idea from data compression (Witten and Bell 1991)
- Encode tokens as numbers as they are read
- Use special (escape) code to introduce new token
- Frequency of ‘escape’ is probability of unseen events
- Consider again example: `mississippi`
- What is the probability of: `river`

Mississippi Ex.: Witten-Bell Discounting

Witten-Bell Discounting: Formulae

- Modified unigram probability

$$P(w) = \frac{\#(w)}{n + r}$$

- Probability of unseen tokens:

$$P(w) = \frac{r}{(n + r)(|V| - r)}$$

Higher-order N-grams

- Modified probability for seen bigrams

$$P(a|b) = \frac{\#(ba)}{\#(b) + r_b}$$

- Remaining probability mass for unseen events

$$\frac{r_b}{\#(b) + r_b}$$

- Estimate for unseen bigrams starting with b (N_b is the set of tokens that never follow b in training text):

$$P(a|b) = \frac{r_b}{\#(b) + r_b} \cdot P(a)/\Sigma_{x \in N_b} P(x)$$

The Next Model: HMM

- HMM — Hidden Markov Model
- Typically used to annotate sequences of tokens
- Most common annotation: Part-of-Speech Tags (POS Tags)
- First, we will make a review of parts of speech in English