CSCI 4152/6509 — Natural Language Processing *19-Oct-2009*

Lecture 16: N-gram Model

Room: FASS 2176
Time: 11:35 – 12:25

Previous Lecture

- Naïve Bayes model (continued):
  - assumption,
  - computational tasks,
  - example,
  - number of parameters,
  - pros and cons;
- N-gram model,
- Language modeling in speech recognition

---

10 N-gram Models

An important task in probabilistic NLP is *language modelling:* Estimating the probability of arbitrary NL (natural language) sentence: P(sentence)
One application of this problem is in speech recognition. In speech recognition, we are interested in

$$\operatorname*{arg\,max}_{\text{sentence}} \mathrm{P}(\text{sentence}|\text{sound})$$

This is equal to:

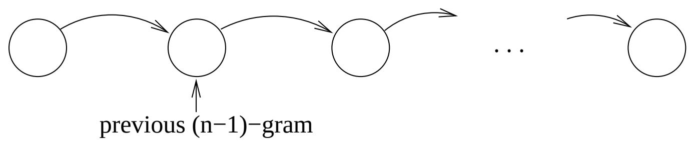
$$\begin{aligned}
\operatorname*{arg\,max}_{\text{sentence}} \mathrm{P}(\text{sentence}|\text{sound}) &= \operatorname*{arg\,max}_{\text{sentence}} \frac{\mathrm{P}(\text{sentence}, \text{sound})}{\mathrm{P}(\text{sound})} \\
&= \operatorname*{arg\,max}_{\text{sentence}} \mathrm{P}(\text{sentence}, \text{sound}) \\
&= \operatorname*{arg\,max}_{\text{sentence}} \mathrm{P}(\text{sound}|\text{sentence})\mathrm{P}(\text{sentence})
\end{aligned}$$

It is easier to estimate $\mathrm{P}(\text{sound}|\text{sentence})$ than $\mathrm{P}(\text{sentence}, \text{sound})$, and it is done by an *acoustic model,* while $\mathrm{P}(\text{sentence})$ is estimated by a *language model.*
N-gram model is very simple and it is among the most successful models for language modelling; trigram ($n = 3$) word model in particular. In an n-gram model, we calculate joint distribution for all n-tuples of consecutive words (or characters). For example, in the trigram model, we count the number of occurrences of each triple of consecutive words from a corpus. Using this statistics, we can estimate the probability of arbitrary word w_3 following two given words w_1 and w_2: $\mathrm{P}(w_3|w_1w_2)$. It is useful to assign some small probability to unseen triples as well (using a technique called *smoothing*). If we use two "dummy" words '$\cdot$' at the beginning of each sentence, then the probability of arbitrary sentence can be calculated as:

$$\mathrm{P}(w_1w_2\dots w_n) = \mathrm{P}(w_1|\cdot\cdot)\mathrm{P}(w_2|w_1\cdot)\mathrm{P}(w_3|w_2w_1)\dots\mathrm{P}(w_n|w_{n-1}w_{n-2})$$

- Reading: Chapter 4 of [JM]
- Graphical representation
- Use of log probabilities

Graphical Representation

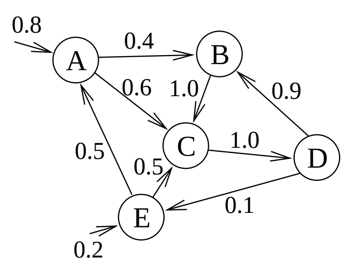


Use of log probabilities

Multiplying a large number of probabilities gives a very small result (close to zero), so in order to avoid floating-point underflow, we should use logarithms of the probabilities in the model.

Markov Chain

Ngram model is a Markov chain.
A *stochastic process* in general is a family of random variables $\{V_i\}$, where i is an index from a set I. A stochastic process is also denoted as $\{V_i, i \in I\}$, or $\{V_t, t \in T\}$, with intuition coming from time index. The index set I can be an arbitrary ordered set, but we will usually assume they it is either finite or countably infinite (i.e., enumerable), and then process can be denoted as $\{V_i\}_{i=1}^{\infty}$. A process is called a *Markov process* if given the value of V_t, for some index t, the values of V_s, where $s > t$, do not depend on values of V_u, where $u < t$. In case of a finite or countably infinite index set, this means that the value of V_i depends only on the value of the previous variable V_{i-1}. In this case, the Markov process is called a *Markov chain.*
A Markov Chain can be described similarly to a deterministic finite automaton, but instead of reading input, we assume that we start in a random state based on a probability distribution, and change states in sequence based on a probability distribution of the next state given the previous state. For example, a Markov chain could be illustrated in the following way.



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 length 4. If we want to model sequences of arbitrary length, we would also need a stopping probability.

Perplexity

- extrinsic and intrinsic evaluation
  In extrinsic evaluation, the language model is embedded in a wider application, and the performance of the model is measured through the performance of the application. For example, we can evaluate performance of a language model by measuring improvement in a speech recognition application in which it is embedded.
  In intrinsic evaluation, we directly evaluate the language model using some measure, such as perplexity.
- Perplexity, W — text, $L = |W|$,

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

- weighted average branching factor
- Text classification using language models