Week 1 Introduction and language model

一、Introduction to NLP

what is NLP?

- computers using natural language as input (understanding) and/or output (generation)
- key applications: machine translation, information extraction, text summarzation, dialogue systems

basic NLP problems

- tagging (part-of-speech tagging, named entity recognition)
- parsing

why is NLP hard?

- Ambiguity (acoustic level 声学 、 semantic level语义、syntactic level句法、discourse level 语境)

what will this course be about?

- NLP sub-problems: part-speech tagging, parsing, word-sense disambiguation, etc.
- machine learning techniques: probabilistic context-free grammars, hidden markov models, estimation / smoothing techniques, the EM algorithm, log-liner models, etc.
- Applications: information extraction, machine translation, natural language interfaces.

a syllabus教学大纲

- language modeling, smoothed estimation
- Tagging, hidden Markov models
- Statistical parsing
- machine learning
- Log-linear model, discriminative methods
- Semi-supervised and unsuprtvised learning for NLP

books

- Comprehensive notes for course: http://www.cs.columbia.edu/~mcollins
- Jurafsky and Martin: Speech and Language Processing (2nd edition)

二、The language modeling problem

The language modeling problem

- **traing set**

  $\mathcal{V}$: finite set of vocabulary

  $\mathcal{V}^{\dagger}$: an infinite set of strings (quite large, may have hundreds of billions of words nowdays)

- **task**

  to learn a probability distribution p that satisfies

$$\sum_{x \in \mathcal{V}^{\dagger}} p(x) = 1, p(x) \geq 0 \ for\ all\ x \in \mathcal{V}^{\dagger}$$

- Why do we need to do this:
  - Speech recognition was the original motivation. (Other applications: optical character recognition, handwriting recognition, machine translations)
  - The estimation techniques developed for this problem is VERY useful for other problems in NLP.
- A naive method

  We have N training sentences, for any sentence $x_1 \ldots x_n$, $c(x_1 \ldots x_n)$ is the count of the sentence in training data, then a naive estimate:

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

  Deficiencies:

  probability of sentences that not seen in training data will be 0.

  has no ability to generate the probability of new sentences.

Markov process

- Definition

  A sequence of random variables $X_1, X_2, \ldots, X_n$, each random variables can take any value in a finite set $\mathcal{V}$, then to model

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

  we can get $|\mathcal{V}|^n$ different sequences in this model.

- First-Order Markov process

$$\begin{aligned} & P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) \\ &= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots X_{i-1} = x_{i-1}) \end{aligned}$$

  the first-order Markov assumption: for any $i \in \{2 ... n\}$, and for any $x_1 \ldots x_n$,

$$P(X_i = x_i | X_1 = x_1, \ldots X_{i-1} = x_{i-1}) = P(X_i = x_i | X_{i-1} = x_{i-1})$$

  then

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$
$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | Xi - 1 = x_{i-1})$$

- Second-Order Markov process

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$
$$= P(X_1 = x_1) \times P(X_2 = x_2 | X_1 = x_1)$$
$$\times \prod_{i=3}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$
$$= \prod_{i=1}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

  Assume $x_0 = x_{-1} = *$, where $*$ is a special "start" symbol. And define $X_n =$STOP where STOP is a special symbol.

Trigram models

- A trigram language model consists of:
  - A finite set $\mathcal{V}$
  - A parameter $q(w|u, v)$ for each trigram u, v, w such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, and $u, v \in \mathcal{V} \cup \{*\}$
- For any sentence $x_1 \cdots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \cdots (n-1)$, and $x_n = \text{STOP}$, the probability of the sentence under the trigram language model is

$$p(x_1 \cdots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-2}, x_{i-1})$$

  where $x_0 = x_{-1} = *$.

Evaluating language models: perplexity

- For test data $s_1, s_2, \cdots, s_m$, define perplexity as

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

where M is the total number of words in the test data, and the log base is 2.

The lower quantity of perplexity is, the better the model is.

- Intuition about perplexity

  Vocabulary is $\mathcal{V}$, and $N = |\mathcal{V}| + 1$, and model predicts

$$q(w|u, v) = \frac{1}{N}$$

  for all $w \in \mathcal{V} \cup \{\text{STOP}\}$, and all $u, v \in \mathcal{V} \cup \{*\}$, we can get perplexity equals to N.

Estimation techniques:

- Maximum likelihood estimate

$$q(w_i|w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

Deficiencies:

- Huge number of parameters: vocabulary size $|\mathcal{V}| = N$,then there are N^3 parameters in the model.
- Numerator and denominator may be 0, which will lead to estimates being unrealistically low or ill defined.

- Liner interpolation
  - Trigram maximum-likelihood estimate

$$q_{\text{ML}}(w_i|w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

  - Bigram maximum-likelihood estimate

$$q_{\text{ML}}(w_i|w_{i-1}) = \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}$$

  - Unigram maximun-likelihood estimate

$$q_{\text{ML}}(w_i) = \frac{\text{Count}(w_i)}{\text{Count}()}$$

  - Then,

$$\begin{aligned} q(w_i|w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\text{ML}}(w_i|w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times q_{\text{ML}}(w_i|w_{i-1}) \\ & + \lambda_3 \times q_{\text{ML}}(w_i) \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i .

  - Estimate the value of λ
    - Hold out part of training data set as validation data
    - Define $c'(w_1, w_2, w_3)$ to be the number of times the trigram (w_1, w_2, w_3) is seen in validation set
    - Choose to maximize:
- discounting methods

Further reading:

C. Shannon. Prediction and entropy of printed English. Bell Systems Technical Journal, 30:50–64, 1951.