[Day-2] Unsupervised Learning for Natural Language Processing

(Part III: Typical Unsupervised Learning Algorithms)

Keh-Yih Su and Jing-Shin Chang
kysu@bdc.com.tw, shin@nlp.csie.ncnu.edu.tw

(2002/8/18)

Behavior Design Corporation (BDC)
No. 5, 2F, Industrial East Road IV, Science-Based Industrial Park
Hsinchu, Taiwan 30077, R.O.C.

Department of Computer Science and Information Engineering
National Chi-Nan University
Puli, Nantou, Taiwan 545, R.O.C.

Day-2: Unsupervised Learning for Natural Language Processing

- Part I: Introduction
  - What and When for Unsupervised Learning, Why it is getting popular
- Part II: Basic Concepts and Background (using EM as an example)
  - Incomplete Data Space
  - Learnability
- **Part III: Typical Unsupervised Learning Algorithms: Viterbi & EM**
  - **Procedures, Characteristics**
- Part IV: Potential Traps & Source of Problems
  - Various Mismatches, Model Deficiencies, Local Maximum, and Over-fitting
- Part V: Suggested Strategies for Better Performance
  - Lessons Learned from Past Experience
  - Recommended Procedures for Unsupervised Learning
- Part VI: Co-Training
  - Basic Principles
  - Example: Chinese New Word Extraction
- References

Part III: Typical Unsupervised Learning Methods: EM and Viterbi Algorithms

- Basic Unsupervised Learning Methods: EM & Viterbi
- Characteristics of the Unsupervised Methods
- An Example: Part-of-speech Tagging
- More Details and Differences Between EM & Viterbi
- Problems Frequently Observed

Basic Unsupervised Learning: EM/Viterbi Algorithm

1. Develop a Model:

- Select Potentially Useful Features
- Build a statistical Language Model with those adopted features

2. Prepare a Training Corpus

3. Set up Initial Conditions:

- EM: Guessing Initial Model Parameter (uniformly, or heuristically), and then calculating the initial expectation
- Viterbi: Guessing Initial Labels (uniformly, or heuristically)

Basic Unsupervised Learning (cont.):

4. Re-Estimate Model Parameters via MLE
   - EM: Using the expectation (which implies weighting every possibility)
   - Viterbi: Using the guessed labels (which implies using only one possibility: a simplified case of EM)
5. Re-Generating Prediction according to new Model Parameters
   - EM: Re-estimate the Expectation of Sufficient Statistic
   - Viterbi: Re-Labeling
6. Repeat the Prediction and Estimation Steps until the joint likelihood value of the training corpus converge

Characteristics of the Unsupervised Learning Algorithms

- Joint likelihood values of the training corpus monotonically increases with iterations

$$\text{EM: } P(w_1^n|\Lambda_0) \le P(w_1^n|\Lambda_1) \le P(w_1^n|\Lambda_2) \cdots$$

$$\text{Viterbi: } P\left(\left[c_1^n\right]_0, w_1^n \mid \Lambda_0\right) \le P\left(\left[c_1^n\right]_0, w_1^n \mid \Lambda_1\right) \le P\left(\left[c_1^n\right]_1, w_1^n \mid \Lambda_1\right) \le P\left(\left[c_1^n\right]_1, w_1^n \mid \Lambda_2\right) \le \cdots$$

- Likelihood values will converge to a local maximum given sufficiently large iterations

Viterbi Training (1):

- Example Task: Tagging a Corpus w_1^n (=w_1, w_2,…, w_n) with the appropriate tag sequence c_1^n (=c_1, c_2,…, c_n)
- 1. Decide Statistical Language Model:

$$\hat{c}_1^n = \arg\max_{c_1^n} P(c_1,\cdots,c_n|w_1,\cdots,w_n,\Lambda)$$
$$= \arg\max_{c_1^n} P(w_1^n|c_1^n,\Lambda)\times P(c_1^n|\Lambda)$$

  - Bi-gram Language Model: $\hat{c}_1^n \equiv \arg\max_{c_1^n} \prod_{i=1}^{n} P(w_i|c_i)\times P(c_i|c_{i-1})$

Viterbi Training (2):

- 2. Get Untagged Corpora:
  - The current design of …
  - det adj/n v/n p …
- 3. Make Initial Guess (based on initial parameter set Λ_0):
  - i.e., prior distribution of unigram, P(c_{ki} | w_i)
  - The Current Design of ...
  - $[c_1^n]_0$: [det n v p ...]

Viterbi Training (3):

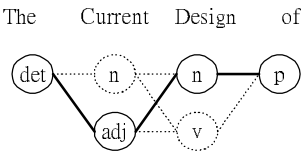
- 4. Maximum Likelihood Estimation

$$\Lambda_1 = \arg\max_{\Lambda} P\left(\left[c_1^n\right]_0, w_1^n \mid \Lambda\right)$$

$$P\left(c_i = \mathrm{n} \mid c_{i-1} = \mathrm{det}\right) = \frac{\#\left[\mathrm{det}, \mathrm{n}\right]}{\#\left[\mathrm{det}\right]}$$

$$P\left(w_i = \mathrm{design} \mid c_i = \mathrm{v}\right) = \frac{\#\left[\mathrm{design}, \mathrm{v}\right]}{\#\left[\mathrm{v}\right]}$$

- 5. Re-tagging: Select the path with maximum probability



$$\left[c_1^n\right]_1 = \arg\max_{\{c_1^n\}} P\left(c_1^n, w_1^n \mid \Lambda_1\right)$$

Viterbi Training (4):

- 6. Re-estimation: Estimate parameters that maximize the likelihood value

$$\Lambda_2 = \arg\max_{\Lambda}\left[P\left(w_1^n \middle| \left[c_1^n\right]_1, \Lambda\right) \times P\left(\left[c_1^n\right]_1 \middle| \Lambda\right)\right]$$

- 7. Repeat: $\Lambda_1 \Rightarrow \Lambda_2 \Rightarrow \Lambda_3 \Rightarrow \Rightarrow \Lambda^*$ (optimal parameters)

- Likelihood Value is Monotonically Increasing

$$P\left(\left[c_1^n\right]_0, w_1^n \mid \Lambda_0\right) \le P\left(\left[c_1^n\right]_0, w_1^n \mid \Lambda_1\right)$$
$$\le P\left(\left[c_1^n\right]_1, w_1^n \mid \Lambda_1\right) \le P\left(\left[c_1^n\right]_1, w_1^n \mid \Lambda_2\right) \le \cdots$$

EM Training (1)

- Example Task: Tagging a Corpus w_1^n (=w_1, w_2,…, w_n) with the appropriate tag sequence c_1^n (=c_1, c_2,…, c_n)

1: Set up Language Model, which is the same as that in the Viterbi Training

- Bi-gram Language Model: $\hat{c}_1^n \equiv \arg\max_{c_1^n} \prod_{i=1}^{n} P(c_i|c_{i-1}) \times P(w_i|c_i)$

2: Prepare Training Corpus, which is the same as that in the Viterbi Training

EM Training (2)

3: Guessing Initial Model Parameter (uniformly, or heuristically), and then calculating the initial expectation of the sufficient statistics ([$N_{c(i)}$, $N_{c(i),\,c(j)}$, $N_{w(k)}$]; for every possible combination)

- $E[N_{c(i)}]$: Expected Number of transitions from a specific POS c(i) (position independent: the position index is ignored)

$$E\left[N_{c(i)}\right] = \sum_{c_1^n}\left[\text{Number of times that c(i) occurs under } c_1^n\right] \times P\left(c_1^n \middle| w_1^n, \Lambda\right)$$

- $E[N_{c(i),\,c(j)}]$: Expected Number of transitions from a specific POS c(i) to another POS c(j) (ignoring the position indexes)

$$E\left[N_{c(i),c(j)}\right] = \sum_{c_1^n}\left[\#[c(i),c(j)] \text{ pairs occurs under } c_1^n\right] \times P\left(c_1^n \middle| w_1^n, \Lambda\right)$$

EM Training (3)

3: Calculate the expectation of the sufficient statistics (cont.)

- $E[N_{w(k)}]$: Expected Number of word w(k) appears in the corpus (position independent: the position index is ignored)

$$E\left[N_{w(k)}\right] = \sum_{c_1^n}\left[\text{Number of times that } w(k) \text{ occurs under } c_1^n\right] \times P\left(c_1^n \middle| w_1^n, \Lambda\right)$$
$$= \left[\text{Number of times that } w(k) \text{ occurs in the corpus}\right]$$

4: Using the expectation to do Maximum Likelihood Estimation

$$P\left(c_i = c(j) | c_{i-1} = c(i)\right) = \frac{E\left[N_{c(i),c(j)}\right]}{E\left[N_{c(i)}\right]}; \quad P\left(w_i = w(k) | c_i = c(j)\right) = \frac{E\left[N_{w(k)}\right]}{E\left[N_{c(j)}\right]}$$

EM Training (4)

5: Re-calculate the expectation of the sufficient statistics

- Repeat the calculations as in Step (3).

6: Re-estimate the parameters

- Repeat the calculations as in Step (4).

7: Repeat the above procedures.

- Likelihood Value Monotonically Increases

$$P\left(w_1^n \mid \Lambda_0\right) \le P\left(w_1^n \mid \Lambda_1\right) \le P\left(w_1^n \mid \Lambda_2\right) \le \cdots$$

EM versus Viterbi (1)

- EM is a kind of soft-labeling (that is, an observation can belong to several different classes simultaneously with various associated probabilities).
  - Example: every possible tag are assigned to a given word-position (i.e., every possible tag sequence are associated with the given word sequence) .
- Viterbi is a kind of hard-labeling (that is, an observation can only belong to one class).
  - Example: only one tag can be assigned to a given word-position (I.e., only one tag sequence is associated with the given word sequence).

EM versus Viterbi (2)

- Soft-labeling versus Hard-labeling during supervised learning
  - Hard-Labeling is a special case of Soft-Labeling
  - Soft-Labeling carries more information than Hard-Labeling does, given the same amount of training data (i.e., more efficient).
  - The advantage diminishes when the corpus size goes larger
  - Soft-Labeling annotation is very difficult to be performed by human

EM versus Viterbi (3)

- Optimization Criteria are Different
  - EM: optimize

$$\hat{\Lambda} = \arg\max_{\Lambda} P\left(w_1^n|\Lambda\right) = \arg\max_{\Lambda} \sum_{c_1^n} P\left(w_1^n, c_1^n|\Lambda\right)$$

  - Viterbi: optimize

$$\hat{\Lambda} = \arg\max_{\Lambda} \left\{\max_{c_1^n} P\left(w_1^n, c_1^n|\Lambda\right)\right\} = \arg\max_{\Lambda} \left\{\max_{c_1^n} \left[P\left(w_1^n|c_1^n, \Lambda\right) \times P\left(c_1^n|\Lambda\right)\right]\right\}$$

  - In general, they would converge to different parameter points (i.e., obtain different parameter sets)

EM versus Viterbi (4)

- Characteristics Comparison:
  - EM can deliver better performance, but requires heavier computation
  - Viterbi is simple and quick, but with inferior performance
  - The performance difference is usually small and tolerable for most NLP and Speech Recognition tasks conducted in the community
    - In many cases, the parameters will be adjusted again by using an adaptive learning algorithm any way
    - Initial difference might not have effect after adaptive learning

More on EM Training [Dempster 77]

- EM (Expectation and Maximization) algorithm: an unsupervised training process which consists of an expectation step followed by a maximization step.
- There is a many-to-one mapping **x**→**y** from ***X*** to ***Y***.
  - **x**: is the *complete data* with density **x** ~ $f(\mathbf{x}|\Phi)$ depending on the parameter set Φ.
  - **y**: the *incomplete data* with the sampling density $g(\mathbf{y}|\Phi)$

$$g(y|\Phi) = \int_{X(y)} f(x|\Phi)\,dx$$

More on EM Training (cont.)

- The EM training procedure in the *p*-th iteration:
  - E-step: Estimate the complete-data sufficient statistics **t(x)** by finding

$$t^{(p)} = E\left[t(X)\middle|y,\Phi^{(p)}\right]$$

  - M-step: Determine $\Phi^{(p+1)}$ which maximizes.

$$f\left(t^{(p)}\middle|\Phi^{(p+1)}\right)$$

E-Step in above Generic Model:

- To find $\pi^{(p+1)}$ from $\pi^{(p)}$ ($\pi^{(p)}$ denotes the value of π after p iterations):
- E-Step:

$$x_1^{(p)} = E\left[X_1 | X_1 + X_2 = 125, \pi^{(p)}\right] = 125 \times \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}\pi^{(p)}}$$

$$x_2^{(p)} = E\left[X_2 | X_1 + X_2 = 125, \pi^{(p)}\right] = 125 \times \frac{\frac{1}{4}\pi^{(p)}}{\frac{1}{2} + \frac{1}{4}\pi^{(p)}}$$

M-Step in above Generic Model:

- M-Step:

$$Let \ \frac{\partial f(x|\pi)}{\partial \pi} = 0 \Rightarrow \pi = \frac{x_2 + x_5}{x_2 + x_3 + x_4 + x_5}$$

$$\pi^{(p+1)} = \frac{x_2^{(p)} + 34}{x_2^{(p)} + 18 + 20 + 34}$$

  - x_1^p and x_2^p are usually not integers.
  - The re-estimation process converge to π^* when p>5, where $\pi^* \approx$ 0.6268214980 is the MLE of π.

Examples of Applying EM to Text Classification (1)

- Classify Documents with EM [Nigam 99]
  - Naïve Bayes's Model

$$P\left(t_a \mid d_i;\hat{\theta}\right)=\sum_{c_j} P\left(t_a \mid c_j;\hat{\theta}\right)\frac{P\left(c_j \mid \hat{\theta}\right)\prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}} \mid c_j;\hat{\theta}\right)}{\sum_{r=1}^{|C|} P\left(c_j \mid \hat{\theta}\right)\prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}} \mid c_j;\hat{\theta}\right)}$$

- Experiment Set Up [Nigam 99]
  - Task 1: 20,017 news articles to be classified into 20 different news groups; test set of 4,000 documents
  - Task 2: 4,199 computer science department web pages to be clustered into four categories; one quarter is used as test data
  - Task 3: 12,902 Reuters articles with 90 topic categories; test set of 3,299 documents

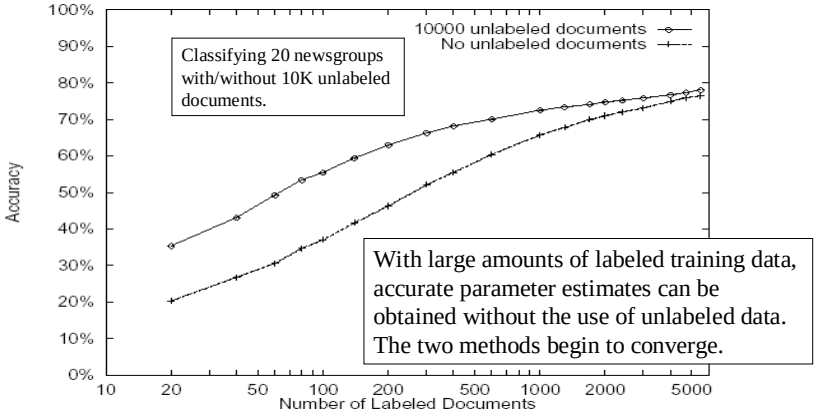
Examples of Applying EM to Text Classification (2)

- Tasks to be conducted
  - Effect of various sizes of labeled data
  - Effect of various sizes of unlabeled data
  - Effect of model mismatch
  - Effect of weighting unlabeled data
    - Likelihood function with modulated unlabeled data

$$l_c\left(\theta \mid D;\mathbf{z}\right)=\log\left(P\left(\theta\right)\right)+\sum_{d_i\in D^L}\sum_{j=1,|C|} z_{ij}\log\left(P\left(c_j \mid \theta\right)P\left(d_i \mid c_j;\theta\right)\right)$$
$$+\lambda\left(\sum_{d_i\in D^U}\sum_{j=1,|C|} z_{ij}\log\left(P\left(c_j \mid \theta\right)P\left(d_i \mid c_j;\theta\right)\right)\right)$$

Examples of Applying EM to Text Classification (3)

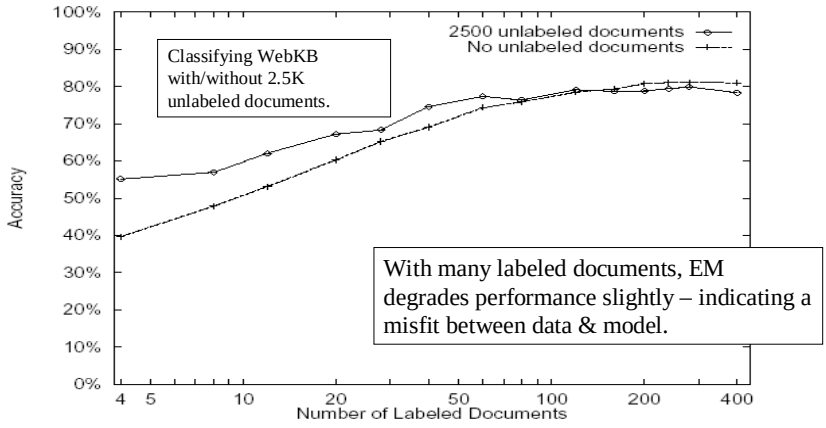
- Effect of various sizes of labeled data [Nigam 99]
  - Benefit of unlabeled data shrinks when increasing the size of labeled data



With large amounts of labeled training data, accurate parameter estimates can be obtained without the use of unlabeled data. The two methods begin to converge.

Examples of Applying EM to Text Classification (4)

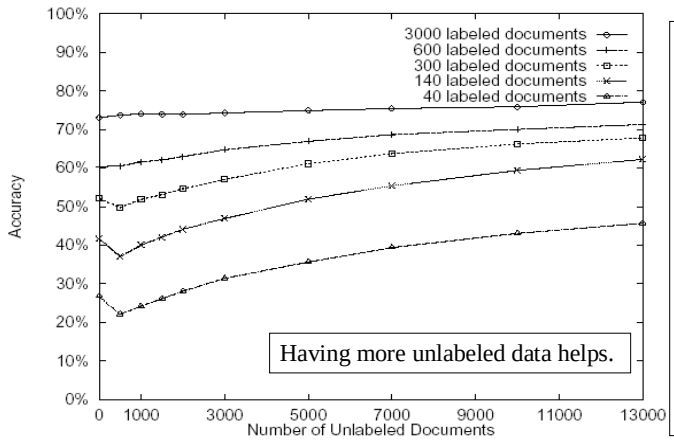
- Effect of various sizes of labeled data (Cont.)
  - Unlabeled data even possible to bring in in adverse effect



With many labeled documents, EM degrades performance slightly – indicating a misfit between data & model.

Examples of Applying EM to Text Classification (5)

- Effect of various sizes of unlabeled data [Nigam 99]

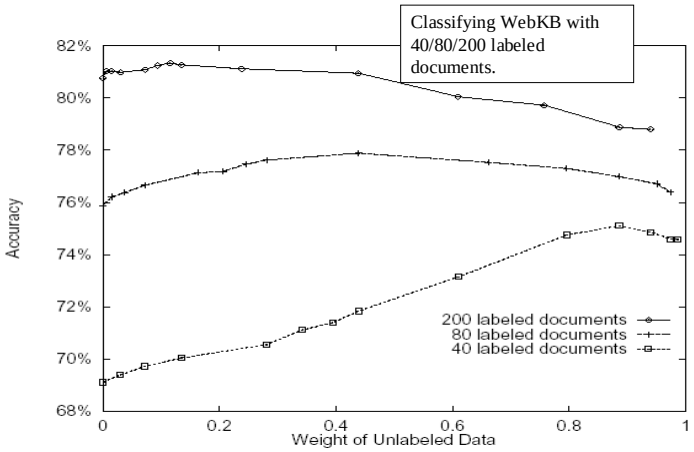


A small unlabeled data set added to a small labeled data set may result in dip in accuracy due to extreme estimates (0, or 1) for unlabeled documents.

Examples of Applying EM to Text Classification (6)

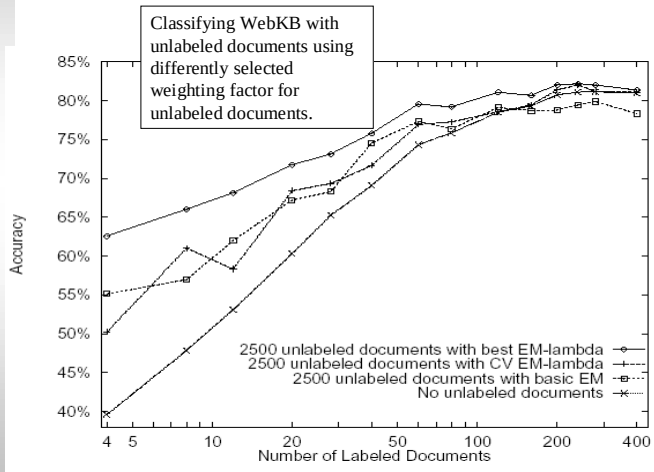
- Effect of weighting unlabeled data [Nigam 99]
  - When there is less labeled data, more weight is given to unlabeled data

When the labeled set is large, accurate parameter estimates are attainable from the labeled data alone, and the unlabeled data should receive less weight.



Examples of Applying EM to Text Classification (7)

- Effect of weighting unlabeled data (Cont.)
  - 240 documents for Cross-Validation set



With large labeled set, CV EM-lambda is more accurate than basic EM. Thanks to the weighting factor, large amounts of unlabeled data no longer degrades accuracy, and yet the algorithm retains the large improvements with small amounts of labeled data.

Examples of Applying EM to Text Classification (8)

- Effect of model mismatch [Nigam 99]
  - Both Viterbi and EM have no mechanism to prevent overfitting

Table 5. Performance of EM using different numbers of mixture components for the negative class and 7000 unlabeled documents. Precision-recall breakeven points are shown for experiments using between one and forty mixture components. Note that using too few or too many mixture components results in poor performance.

Category	EM1	EM3	EM5	EM10	EM20	EM40
acq	70.7	75.0	72.5	77.1	68.7	57.5
corn	44.6	45.3	45.3	46.7	41.8	19.1
crude	68.2	72.1	70.9	71.6	64.2	44.0
earn	89.2	88.3	88.5	86.5	87.4	87.2
grain	67.0	68.8	70.3	68.0	58.5	41.3
interest	36.8	43.5	47.1	49.9	34.8	25.8
money-fx	40.3	48.4	53.4	54.3	51.4	40.1
ship	34.1	41.5	42.3	36.1	21.0	5.4
trade	56.1	54.4	55.8	53.4	35.8	27.5
wheat	52.9	56.0	55.5	60.8	60.8	43.4

Examples of Applying EM to Text Classification (9)

- Effect of model mismatch (Cont.)

Table 6. Performance of EM using different numbers of mixture components for the **negative** class, but with no unlabeled data. Precision-recall breakeven points are shown for experiments using between one and forty mixture components.

Category	NB1	NB3	NB5	NB10	NB20	NB40
acq	69.4	69.4	65.8	68.0	64.6	68.8
corn	44.3	44.3	46.0	41.8	41.1	38.9
crude	65.2	60.2	63.1	64.4	65.8	61.8
earn	91.1	90.9	90.5	90.5	90.5	90.4
grain	65.7	63.9	56.7	60.3	56.2	57.5
interest	44.4	48.8	52.6	48.9	47.2	47.6
money-fx	49.4	48.1	47.5	47.1	48.8	50.4
ship	44.3	42.7	47.1	46.0	43.6	45.6
trade	57.7	57.5	51.9	53.2	52.3	58.1
wheat	56.0	59.7	55.7	65.0	63.2	56.0

Examples of Applying EM to Text Classification (10)

- Effect of model mismatch (Cont.)

Table 8. Performance of using multiple mixture components when the number of components is selected via cross-validation (EM*CV) compared to optimal selection (EM*) and straight naive Bayes (NB1). Note that cross-validation usually selects too few components.

Category	NB1	EM*	EM*CV	EM*CV vs NB1
acq	69.4	83.9 (10)	75.6 (1)	+6.2
corn	44.3	52.8 (5)	47.1 (3)	+2.8
crude	65.2	75.4 (8)	68.3 (1)	+3.1
earn	91.1	89.2 (1)	87.1 (1)	-4.0
grain	65.7	72.3 (8)	67.2 (1)	+1.5
interest	44.4	52.3 (5)	42.6 (3)	-1.8
money-fx	49.4	56.9 (10)	47.4 (2)	-2.0
ship	44.3	52.5 (7)	41.3 (2)	-3.0
trade	57.7	61.8 (3)	57.3 (1)	-0.4
wheat	56.0	67.8 (10)	56.9 (1)	+0.9

Problems Frequently Observed (1)

- Adopt *Ad Hoc* Features Selection:
  - Over simplified bigram, trigram models may fail to gain success on complicated NLP tasks
  - Class-based features not used, resulting in a large number of parameters (that is, available training data may not be enough to support the model)
  - Feature Space determines the Upper Bound of performance
- Overlook Feature Dependency : Model Deficiency
  - Inappropriate independence assumptions, inappropriate dependency relationship assumed (to be described in the afternoon session)
- Over Fitting of Model:
  - Using high Model Complexity with Small Training Corpus

Problems Frequently Observed (2)

- Un-hinted Initial Guess:
  - Trapped in undesired Local Maximum
- Unseen and Untrained Events not Well Estimated
  - Give poor performance when the case involves the unseen event
- Mismatch between ML Estimation & Human Preference not Compensated when it can be done
  - Maximizing training set likelihood does not imply to have good training set performance
  - Sometimes, performing adaptive learning on incomplete data space (i.e., based on observations) is possible (e.g., adjusting HMM parameters for generating corresponding text string, or adjusting parameters to generate corresponding target sentences)

Problems Frequently Observed (3)

- System Sensitivity (versus statistical characteristics variation) not considered
  - System Sensitivity : the degree of variation of system performance that will be caused by the variation of the statistical characteristics between the training set and the testing set
  - Statistical characteristics variation: variation between inherited statistical characteristics (implied by the parameters) of the training set and the testing set
  - Models built upon deeper structures (not word n-gram) is usually less sensitive