Improve Speech Recognition by Re-ranking Methods

Presented by Tzan-Hwei Chen

main Reference

- [1] Z. Zhou, J. Gao, F.K. Soong and H. Meng ,"A Comparative Study of Discriminative Methods for Reranking LVCSR N-Best Hypotheses in Domain Adaptation and Generalization", ICASSP 2006 .

- [2] L. Maugu and M. Padmanabhan, "Error corrective mechanisms for speech recognition.", ICASSP 2001 .

Reference

- [3] M. Collins, “Discriminative Training Methods for Hidden Markov Models : Theory and Experiments with Perceptron Algorithms”, EMNLP 2002.

- [4] M. Collins, and T. Koo, “Discriminative Reranking for Natural Language Parsing”, ICML 2000.

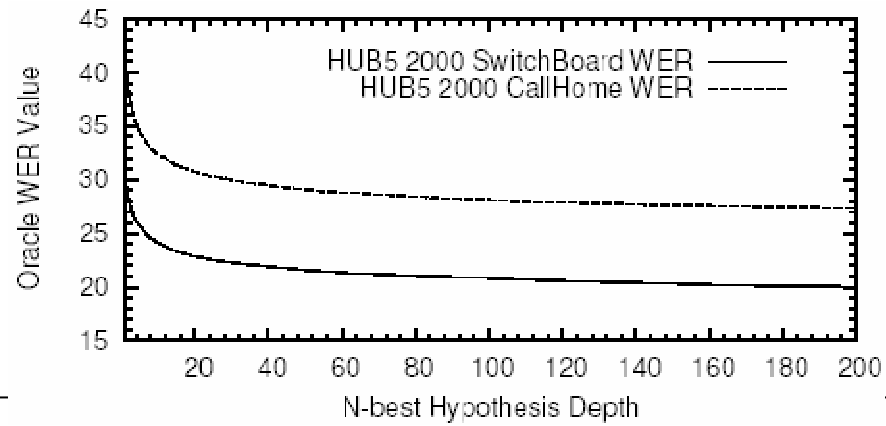
- [5] J. Gao, H. Yu, W. Yuan and P. Xu, “Minimum Sample Risk Methods for Language Modeling”, EMNLP 2005.

- [6]L. Mangu and E. Brill, “Automatic Rule Acquisition for Spelling Correction”, ICML 1997.

- [7]M. Collins, “Discriminative Training Methods for Hidden Markov Models : Theory and Experiments with Perceptron Algorithms”, EMNLP 2002

Introduction (1/2)

- Current LVCSR system rely on a HMM based acoustic mode and a n-gram language model.
- Several LVCSR perform multiple recognition pass on each speech utterance to get the best performance.
- Even in such multi-pass LVCSR system, a significant amount untapped improvements remain hidden inside the LVCSR n-best list and word lattices.



Introduction (2/2)

- There are some methods to extract the performance in n-best list or lattices.
  - Using the additional knowledge sources
  - Discriminative algorithms

[1] introduction (1)

- This work attempts to utilize various discriminative algorithms to improve LVCSR performance by reranking the N-best Hypotheses
  - Perceptron
  - Boosting
  - SVM
  - Minimum sample risk (MSR)

- Comparing discriminative algorithms in terms of their performances in domain adaptation, generalization and time efficiency.

[1] problem definition of N-best reranking (1)

- In the training data set, there are n speech utterance, and n_i sentence hypothesis for each utterance.
- Define $x_{i,j}$ as the $j-\mathrm{th}$ hypothesis of the $i-\mathrm{th}$ utterance, $x_{i,R}$ as the best utterance among $\{x_{i,j}\}$
- There is a separate test set of $y_{i,j}$ with similar definitions as the training set
- Define $D+1$ features, $f_d(h), d = 0, \ldots D, h$ is a hypothesis

[1] problem definition of N-best reranking (2)

- Define a discriminant function $g(h,\vec{w}) = \sum_{d=1}^{D} w_d f_d(h)$ (1)

- The reranking problem becomes searching for a $\vec{w} = \{w_1, w_2, \ldots, w_d\}$ that satisfies the following condition on the testing set

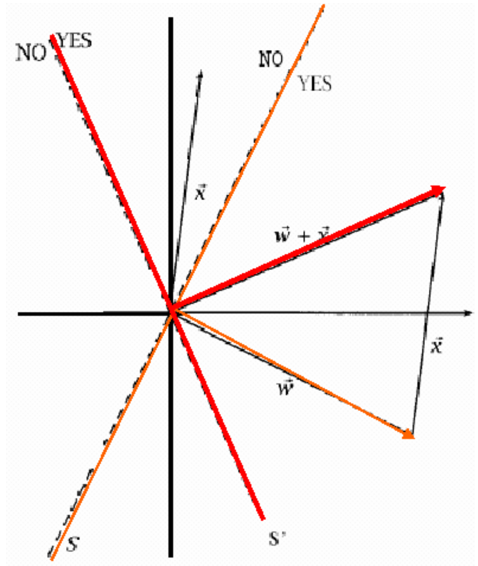
$$g(y_{i,R},\vec{w}) > g(y_{i,j},\vec{w}) \qquad \forall i, \forall j \neq R$$

- *sample risk* : $SR(\vec{w}) = \sum_{i=1}^{n} Er(x_{i,R}, g^*(x_{i,j},\vec{w}))$

- Training method :
  - Directly minimize SR - minimum sample risk
  - Indirectly minimize SR - perceptron, boosting

[1] introduction of perceptron and boosting

- Perceptron :
  - a binary classifier
  - Find a *margin* to separate the two classes.



- Boosting
  - To Combine a set of simple “weak” classification method in to a single “strong” method
  - To find a small subset of the features that contribute most to reducing the loss function.

[1] the perceptron algorithm (1)

- Find a vector $\vec{w}$ which minimizes the classification errors on training data.

- Perceptron optimizes a minimum square error (MSE) loss function :

$$MSELoss_i(\vec{w}) = \frac{1}{2}[g(y_{i,R}, \vec{w}) - g(y_{i,j}, \vec{w})]^2$$

- The algorithm :

Set $w_0 = 1$ and $w_d = 0$, $d = 1 \ldots D$
For $j = 1 \ldots t$ (t is the total number of iterations)
    For each n_i, $i = 1 \ldots n$
        Choose the $x_{i,j}$ with the largest $g(x_{i,j})$ value
        For each w_d (η = size of learning step)
            $w_d = w_d + \eta(f_d(x_{i,R}) - f_d(x_{i,j}))$

Figure 1. The perceptron algorithm

[1] the perceptron algorithm (2)

- Convergence proof :
  - Definition 1:

Let $\overline{GEN}(x_i) = \{x_{i,j}\} - \{x_{i,R}\}$, we will say tha a training set $(x_i, x_{i,R})$ is ***separable with margin*** $\delta > 0$ if there exists vector U with $\|\mathrm{U}\| = 1$ such that

$$\forall i, \forall z \in \overline{GEN}(x_i), \mathrm{U}\,\vec{\mathrm{f}}(x_{i,R}) - \mathrm{U}\,\vec{\mathrm{f}}(x_{i,z}) \geq \delta$$

  - Theorem 1

For any training set $(x_i, x_{i,R})$ whice is separable with margin δ, then for the perceptron algorithm in figure 1

$$\text{number of mistakes} \leq \frac{R^2}{\delta^2}$$

where R is a constant such that

$$\forall i, \forall z \in \overline{GEN}(x_i), \left\| \vec{\mathrm{f}}(x_{i,R}) - \vec{\mathrm{f}}(x_{i,z}) \right\| \leq R$$

[1] the perceptron algorithm (3)

- Convergence proof (cont):
  - Proof of the theorem 1

Let w^{k} the weight vector before the k'th mistake is made. It follows that $w^{1} = [1,0,\ldots,0]$.

Suppose the k'th is made at the i'th utterance

$$z = \arg \max_{x_{i,j} \in \overline{GEN}(x_i)} w^{\mathrm{k}} \vec{\mathrm{f}}(x_{i,j})$$

It follows that from the algorithm updates $w^{\mathrm{k+1}} = w^{\mathrm{k}} + (\vec{\mathrm{f}}(x_{i,R}) - \vec{\mathrm{f}}(x_{i,z}))$

taking the inner products of both sides with the vector U

$$\mathrm{U}w^{\mathrm{k+1}} = \mathrm{U}w^{\mathrm{k}} + \mathrm{U}\,\vec{\mathrm{f}}(x_{i,R}) - \mathrm{U}\,\vec{\mathrm{f}}(x_{i,z}) \geq \mathrm{U}w^{\mathrm{k}} + \delta$$

it follow by induction on k that for all k

$$\mathrm{U}w^{\mathrm{k+1}} \geq k\delta$$

and $\mathrm{U} \cdot w^{\mathrm{k+1}} \leq \|\mathrm{U}\| \|w^{\mathrm{k+1}}\|$, so $\|w^{\mathrm{k+1}}\| \geq k\delta$

[1] the perceptron algorithm (4)

- Convergence proof (cont):
  - Proof of the theorem 1

Deriving an upper bound for $\left\| w^{k+1} \right\|^2$

$$\left\| w^{k+1} \right\|^2 = \left\| w^{k} \right\|^2 + \left\| \vec{f}(x_{i,R}) - \vec{f}(x_{i,z}) \right\|^2 + 2w^{k} \cdot \vec{f}(x_{i,R}) - \vec{f}(x_{i,z})$$

$$\le \left\| w^{k} \right\|^2 + R^2$$

It follows by induction that $\left\| w^{k+1} \right\|^2 \le kR^2$

Combining the upper bound and lower bound of $\left\| w^{k+1} \right\|$:

$$(k\delta)^2 \le \left\| w^{k+1} \right\|^2 \le kR^2 \Rightarrow k \le \frac{R^2}{\delta^2}$$

[1] the perceptron algorithm (5)

- If training data are not separable :
  - Definition 2

Given a sequence $(x_{i,j}, x_{i,R})$, for a U, δ pair define

$$m_i = \mathrm{U}\,\vec{\mathrm{f}}(x_{i,R}) - \max_{z \in \overline{GEN}(x_i)} \mathrm{U}\,\vec{\mathrm{f}}(x_{i,z})$$

and

$$\varepsilon_i = \max\{0, \delta - m_i\},\ D_{U,\delta} = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_i^2}$$

  - Theorem 2

$$\text{number of mistakes} \le \min_{U,\delta} \frac{(R + D_{U,\delta})^2}{\delta^2}$$

[1] the boosting algorithm (1)

- Focuses on modeling linguistic features and attempts to minimize the ranking error on training data.

- Using the following loss function to approximate the ranking error

$$BLoss(\vec{w}) = \sum_{i=1}^{n} \sum_{j=2}^{n_i} \exp(-[\vec{w}^t \vec{f}(x_{i,1}) - \vec{w}^t \vec{f}(x_{i,j})])$$

$$RError(\vec{w}) = \sum_{i=1}^{n} \sum_{j=2}^{n_i} I[\vec{w}^t \vec{f}(x_{i,1}) - \vec{w}^t \vec{f}(x_{i,j})]$$

$$I[\alpha] = \begin{cases} 1 & \text{if } \alpha \le 0 \\ 0 & \text{otherwise} \end{cases}$$

[1] the boosting algorithm (2)

- Definition :

$$Upd(\vec{w}, d, \delta) = \{w_0, w_1, ..., w_d + \delta, ..., w_D\}$$

$$M_{i,j}(\vec{w}) = g(x_{i,1}, \vec{w}) - g(x_{i,j}, \vec{w})$$

- The optimal feature/weight pair

$$(d^*, \delta^*) = \arg\min_{d,\delta} BLoss(Upd(\vec{w}, d, \delta))$$

$$BestWt(k, \vec{w}) = \arg\min_{\delta} BLoss(Upd(\vec{w}, d, \delta))$$

$$BestBLoss = \min_{\delta} BLoss(Upd(\vec{w}, d, \delta)) = BLoss(Upd(\vec{w}, d, BestWt(k, \vec{w})))$$

[1] the boosting algorithm (3)

Initialize

Set $w_0 = \arg\min_{w_{best}} \sum_{i=1}^{n} \sum_{j=2}^{n_i} I[w_{best} f_0(x_{i,1}) - w_{best} f_0(x_{i,R})]$

Set $w_k = 0$ for $d = 1 \ldots D$

For all $i, 2 \le j \le n_i$, set Margin $M_{i,j} = w_0[f_0(x_{i,1})] - w_0[f_0(x_{i,j})]$

For all $d = 1 \ldots D$, set

$$A_d^+ = \{(i,j) \mid f(x_{i,1}) - f(x_{i,j}) = 1\}$$

$$A_d^- = \{(i,j) \mid f(x_{i,1}) - f(x_{i,j}) = -1\}$$

[1] the boosting algorithm (4)

Repeat for $t = 1$ to N

Calculate $Z = \sum_{i=1}^{n} \sum_{j=2}^{n_i} \exp^{-M_{i,j}}$

For $d = 1$ to D

Set $W_d^+ = W_d^- = 0$

For $(i, j) \in A_d^+$, $\quad W_d^+ = W_d^+ + e^{-M_{i,j}}$

For $(i, j) \in A_d^-$, $\quad W_d^- = W_d^- + e^{-M_{i,j}}$

$G_d = \left| \sqrt{W_d^+} - \sqrt{W_d^-} \right|$

Choose $d^* = \arg\max_d G_d$ and $\delta^* = \frac{1}{2} \log \frac{W_d^+ + \varepsilon Z}{W_d^- + \varepsilon Z}$

For $(i, j) \in A_{d^*}^+$, $\quad M_{i,j} = M_{i,j} + \delta^*$

For $(i, j) \in A_{d^*}^-$, $\quad M_{i,j} = M_{i,j} - \delta^*$

$w_{d^*}^t = w_{d^*}^{t-1} + \delta^*$

[1] the boosting algorithm (5)

- Derivation of updates

$$BestWt(d,\vec{w}) = \arg\min_{\delta} BLoss(Upd(\vec{w},d,\delta))$$

$$BestBLoss = BLoss(Upd(\vec{w},d,BestWt(d,\vec{w})))$$

Note that and update in parameters from $\vec{w}$ to $Upd(\vec{w},d,\delta)$ results in

$$g(x_{i,j},Upd(\vec{w},d,\delta)) = g(x_{i,j},\vec{w}) + \delta f_d(x_{i,j})$$

and

$$\begin{aligned} M_{i,j}(Upd(\vec{w},d,\delta)) &= g(x_{i,1},Upd(\vec{w},d,\delta)) - g(x_{i,j},Upd(\vec{w},d,\delta)) \\ &= g(x_{i,1},\vec{w}) - g(x_{i,j},\vec{w}) + \delta f_d(x_{i,1}) - \delta f_d(x_{i,j}) \\ &= M_{i,j}(\vec{w}) + \delta[f_d(x_{i,1}) - f_d(x_{i,j})] \end{aligned}$$

[1] the boosting algorithm (6)

The Update $BLoss(Upd(\vec{w}, d, \delta)) = \sum_{i=1}^{n} \sum_{j=2}^{n_i} \exp(-M_{i,j}(Upd(\vec{w}, d, \delta))]$

$$= \sum_{i=1}^{n} \sum_{j=2}^{n_i} \exp(-M_{i,j}(\vec{w}) - \delta[f_d(x_{i,1}) - f_d(x_{i,j})])$$

Next we note that $[f_d(x_{i,1}) - f_d(x_{i,j})]$ can take on three values: +1, -1, or 0

$$A_d^+ = \{(i,j) \mid f(x_{i,1}) - f(x_{i,j}) = 1\}$$

$$A_d^- = \{(i,j) \mid f(x_{i,1}) - f(x_{i,j}) = -1\}$$

$$A_d^0 = \{(i,j) \mid f(x_{i,1}) - f(x_{i,j}) = 0\}$$

Give these definition we define

$$W_d^+ = \sum_{(i,j) \in A_d^+} e^{-M_{i,j}(\vec{w})}$$

$$W_d^- = \sum_{(i,j) \in A_d^-} e^{-M_{i,j}(\vec{w})}$$

$$W_d^0 = \sum_{(i,j) \in A_d^0} e^{-M_{i,j}(\vec{w})}$$

[1] the boosting algorithm (7)

$$\frac{dBLoss(Upd(\vec{w},d,\delta))}{d\delta}=\frac{e^{-\delta}W_d^{+}+e^{\delta}W_d^{-}+W_d^{0}}{d\delta}=0$$

$$=> -e^{-\delta}W_d^{+}+e^{\delta}W_d^{-}=0$$

$$=> e^{-\delta}W_d^{+}=e^{\delta}W_d^{-}$$

$$=> -\delta+\log(W_d^{+})=\delta+\log(W_d^{-})$$

$$=> \delta=\frac{1}{2}\log(\frac{W_d^{+}}{W_d^{-}})$$

$$BestBLoss=e^{-\delta}W_d^{+}+e^{\delta}W_d^{-}+W_d^{0}$$

$$=e^{-\frac{1}{2}\log(\frac{W_d^{+}}{W_d^{-}})}W_d^{+}+e^{\frac{1}{2}\log(\frac{W_d^{+}}{W_d^{-}})}W_d^{-}+W_d^{0}$$

$$=\sqrt{\frac{W_d^{-}}{W_d^{+}}}\times W_d^{+}+\sqrt{\frac{W_d^{+}}{W_d^{-}}}\times W_d^{-}+W_d^{0}$$

$$=2\sqrt{W_d^{-}W_d^{+}}+W_d^{0}$$

$$=2\sqrt{W_d^{-}W_d^{+}}+Z-W_d^{+}-W_d^{-}$$

$$=Z-(W_d^{+}-W_d^{-})^2$$

[1] Minimum sample risk algorithm (1)

- MSR employs a simple heuristic training algorithm that minimizes the error rate on training sample directly.

- Is has not been proved in theory that MSR is always "robust"

- MSR operates like a multidimensional function :
  - It select subset of features that are most effective among all candidate features
  - The parameters of the model are then optimized iteratively

[1] Minimum sample risk algorithm (2)

- Training algorithm :
  - Taking the feature vector as a set of direction; the first direction is selected and the objection is minimized along that direction using *line search*
  - Then from there along the second direction to its minimum.
  - The simple method can work under two assumptions
    - There exists an implementation of line search that optimizes the function along one direction efficiently.
    - The number of candidate feature is not too large and these features are not highly correlated.

[1] Minimum sample risk algorithm (3)

- Grid line search
  - Determining for each feature a sequence of grids with differently sized intervals.
  - We begin with a discussion on minimizing $Er(.)$
  - Let $\vec{w}$ be the current model parameter vector, and f_d be the selected feature. The line search aims to find the optimal parameter w_d^* so as to minimize $Er(.)$
  - For a training sample $(x_i, x_{i,R})$, the score of each candidate word string $x_{i,j} \in GEN(x_i)$ can be decomposed :

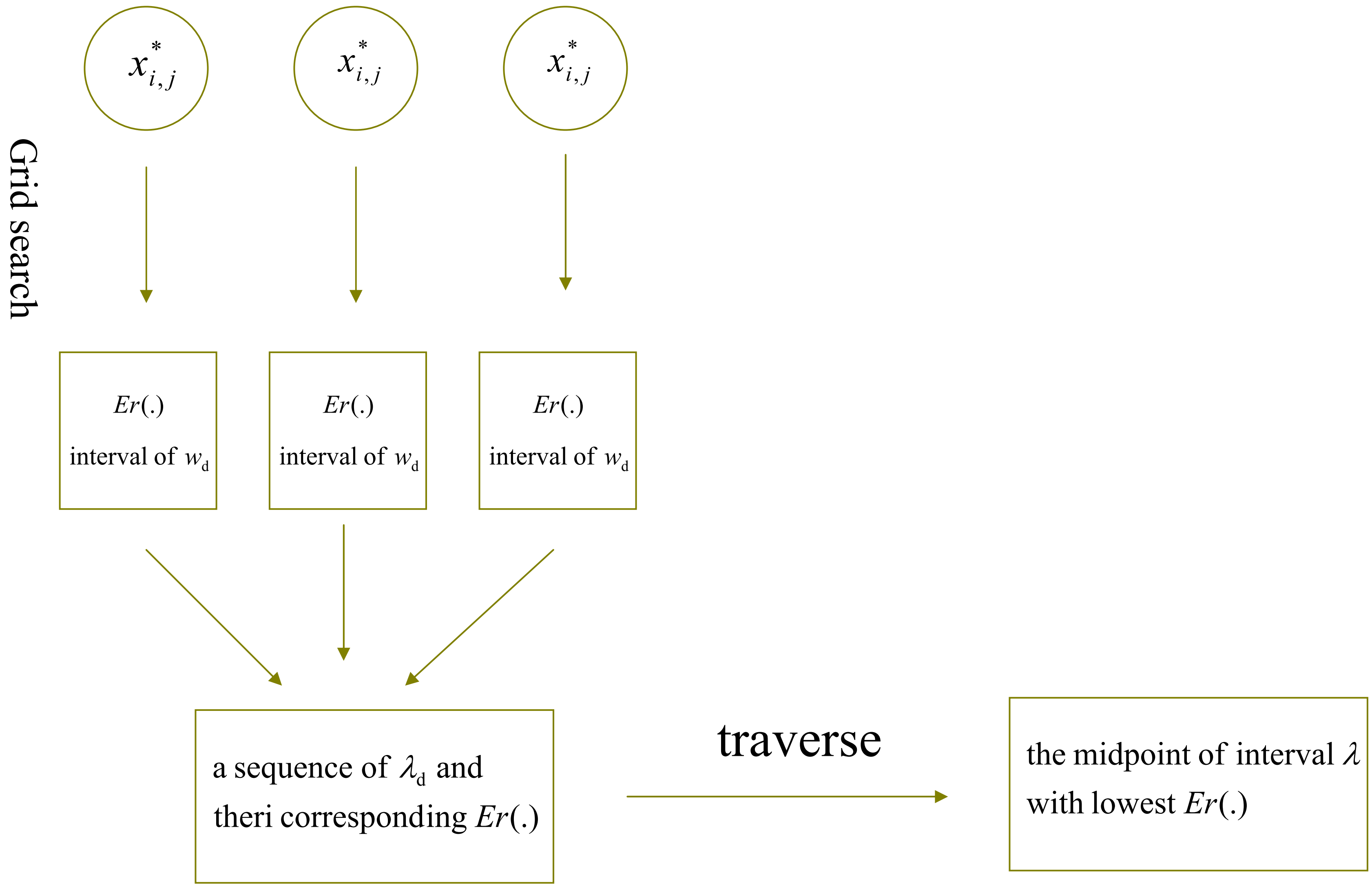
$$\sum_{d'=0 \vee d' \neq d}^{D} w_{d'} f_{d'}(h) + w_d f_d(h)$$

[1] Minimum sample risk algorithm (4)

- Grid line search (cont)
  - If several candidate word string have the same feature value $w_d f_d(h)$ their relative rank will remain the same for any $f_d(h)$
  - We group the candidate using $f_d(h)$, so that candidates in each group have the same value of $f_d(h)$
  - In each group, we define the active candidate $x^*_{i,j}$ with the highest value

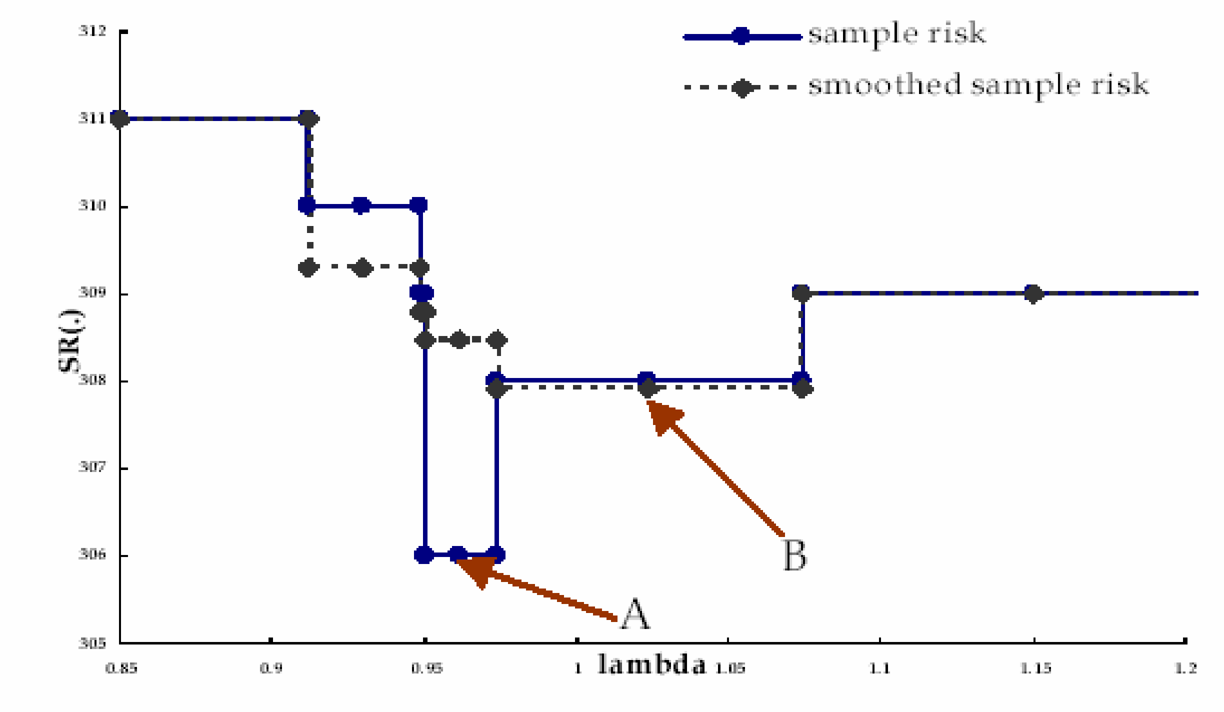
$$\sum_{d'=0 \vee d' \neq d}^{D} w_{d'} f_{d'}(h)$$

[1] Minimum sample risk algorithm (5)



[1] Minimum sample risk algorithm (5)

Grid line search (cont)



$$\text{smoothed sample risk} = \int_{w-b}^{w+b} SR(w)dw$$

[1] Minimum sample risk algorithm (6)

- Feature subset selection :
  - Reducing the number of features is essential for two reason
    - Reduce computational complexity
    - Ensure the generalization property of linear model.
  - the effectiveness of a features is measured

$$E(f_d) = \frac{SR(f_0) - SR(f_0 + w_d f_d)}{\max\limits_{i=1,\ldots,D} SR(f_0) - SR(f_0 + w_i f_i)}$$

  - The cross correlation coefficient between two features

$$C(i,j) = \frac{\sum_{m=1}^{M} x_{mi} x_{mj}}{\sqrt{\sum_{m=1}^{M} x_{mi}^2 \sum_{m=1}^{M} x_{mj}^2}}$$

[1] Minimum sample risk algorithm (7)

- Feature subset selection (cont):
  - For each of the candidate features, compute the value of $E(f_i)$. Choose the one with highest $E(f_i)$. Let us denote this feature as f_1
  - To select the second feature, compute the cross correlation coefficient of the remain $D-1$ feature.
  - Select the second feature f according to

$$j^* = \arg \max_{j=2,\ldots D} \left\{ \alpha E(f_j) - (1-\alpha) C(1,j) \right\}$$

  - Select $k-\text{th}$ feature, $k = 1,\ldots,K$ according to

$$j^* = \arg \max_{j=2,\ldots D} \left\{ \alpha E(f_j) - \frac{(1-\alpha)}{k-1} \sum_{i=1}^{k-1} C(i,j) \right\}$$

[1] Minimum sample risk algorithm (7)

- The MSR algorithm

1	Set $w_0 = 1$ and $w_d = 0$, $d = 1 \ldots D$
2	Rank all features by its expected impact on reducing training error, and select the top N features
3	For $j = 1 \ldots t$ (t is the total number of iterations)
4	For each $n = 1 \ldots N$
5	Update w_n using linear search

[1] experiment (1)

- Two Mandarin dictation speech corpora
  - A large novel-domain speech corpus with a balanced set of speakers in terms of gender and age
  - General-domain data test set

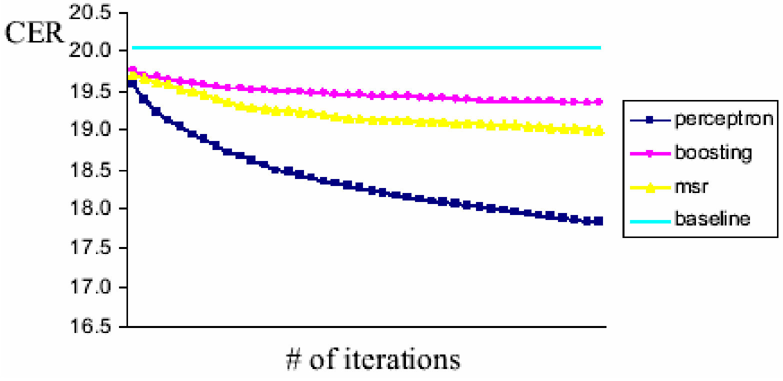
- We divide the novel-domain corpus into
  - Domain-specific training set (DTr-Set)
  - Domain-specific test set (DTe-Set)

Data Sets	Task	Utterance Count	Domain
DTr_Set	Training	84,498	Novel
DTe_Set	Testing	21,123	Novel
GTe_Set	Testing	500	General

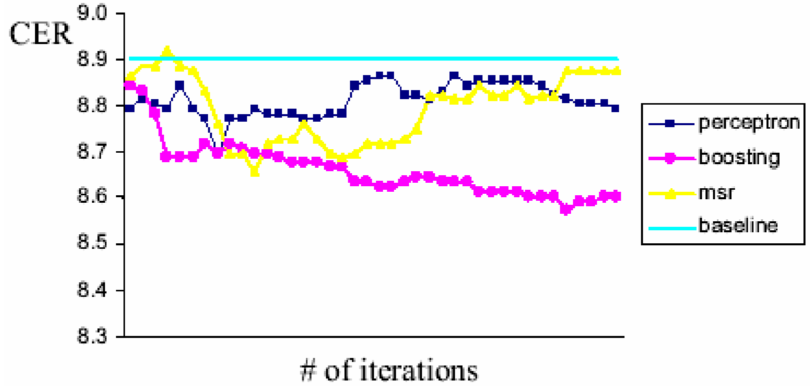
[1] experiment (2)

- Feature selection :
  - Base feature f_0: the recognizer score
  - Assign each word unigram/bigram a unique id $i, i = 1,...,D$
  - $f_i(h)$ is the count of the unigram/bigram with id i in h

- Top 20 hypotheses were adopted in training, 100-best hypotheses in testing.



Comparison on DTe_Set



Comparison on GTe_Set

[1] experiment (3)

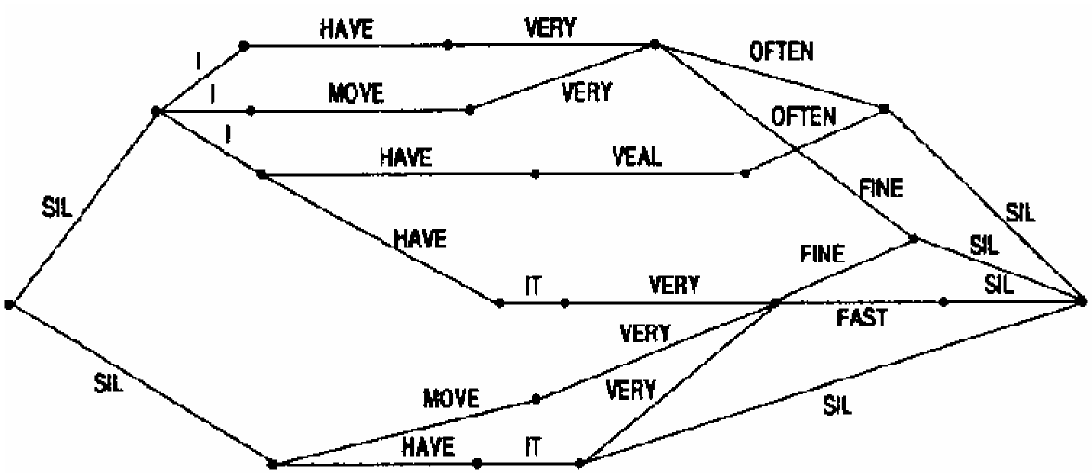
- Discussion :
  - For domain-adaptation, perceptron performs the best
  - For generalization, perceptron performs the worst and boosting performs the best
  - Ranking SVM provides similar CER reduction as the boosting method for both domain adaptation and generalization

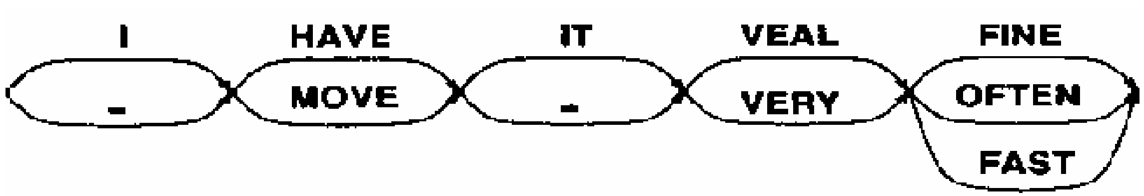
Algorithm	Training Time	DTe-Set CER %	GTe-Set CER %
Baseline	--	20.04	8.90
Perceptron	27 minutes	17.83	8.79
Boosting	16 minutes	19.35	8.60
MSR	16 minutes	19.00	8.87
Ranking SVM	54.8 hours	19.30	8.60
Oracle	--	11.29	4.16

[1] Discussion

- For domain adaptation, perceptron perform the best
  - Perceptron treats all training samples equally while the other three algorithm only concentrate most distingushing.
- For generalization, boost perform the best
  - Domain-specific training data
    - Rules which are applicable only in the specific domain
    - Rules which are applicable in general
- Ranking SVM provides similar CER

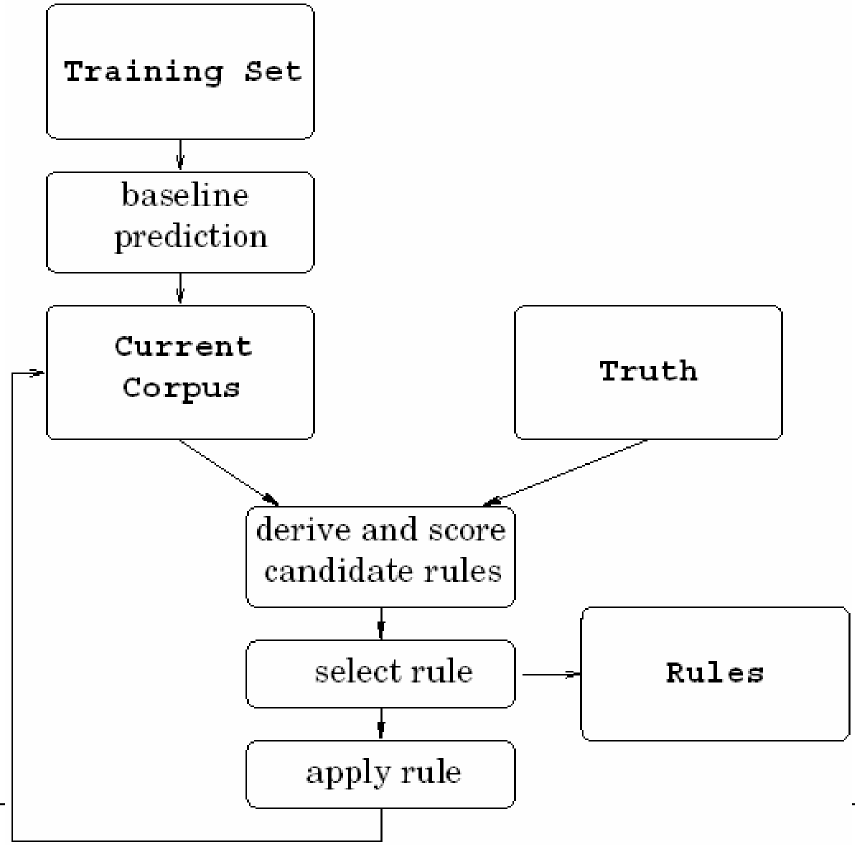
[2] consensus decoding (1)





[2] transformation-based learning (1)

- We must specify a
  - baseline predictor
  - a set of allowable transformation types
  - objective function for learning.



[2] transformation-based learning (2)

- The baseline predictor : assumes the highest ranked candidate is the correct one.

- The allowable transformations in these experiments

$$\text{Change } c_1 \text{ to } c_2 \text{ if}$$
$$A_1 op_1 v_1 \text{ and } A_2 op_2 v_2 \text{ and} \ldots A_k op_k v_k$$

- The feature used by the learner
  - Word identity, duration and posterior probability of the two competing words
  - Difference in the posterior probabilities of the two candidates
  - Temporal position of the confusion set in the sentence
  - Number of candidates in the confusion set

[2] transformation-based learning (3)

- Ex : “choose the second candidate if the first candidate is the word “A”, the second is “-” and the difference in posterior probabilities between them is less than 0.1”
- The objective function used in this experiment is the classification accuracy

[2] experiments (1)

- Prerequisite for the success is the same error pattern are observed in training and testing

- Built two system :
  - *Small* : trained on 60h
  - *Big* : trained on 243h

- From acoustic data not used in training data system *small* :
  - 4000 utterances for rule training
  - 2000 utterances for held-out data

[2] experiments (2)

Rank	Classification accuracy (%) Training set	Held-out set
1	73.3	73.2
2	10.3	10.8
3	3.8	3.9

k	Oracle WER (%) Training set	Held-out set
1	38.0	37.5
2	25.1	24.5
3	20.0	19.3

[2] experiments (3)

- In the training data : 23% only one candidate and this word is correct 95% of the time.

- Examining the initial training set which the correct word is either the first or second word :
  - This word is correct in more than 92% which posterior probability greater than 0.8
  - The final training set contains 23% of all confusion set and the top word has a baseline classification accuracy of 67%
  - The potential overall WER improvement is around 10% absolute.

[2] experiments (4)

- In addition to improving word error rate, this method has the advantage of producing corrective rules learned are :
  - Choose the second candidate if the first is a short word with a posterior probability less than 0.46 and the second word is "-"
  - Choose the second candidate if the first word is "A", the second word is "UH" and the difference in posterior probability is less than 0.63
- Experiment result

Hypothesis	Word Error Rate (%) WS97 (*Small*)	WS97 (*Big*)
MAP	38.0	36.0
Consensus	37.2	35.1
Consensus+	36.4	34.6