Statistical Machine Translation

LECTURE – 8

DISCRIMINATIVE TRAINING

APRIL 21, 2010

Brief Outline

- Introduction - Non-generative Approach
- The Overall Principle
- Representation using features
- Methods of Parameter Tuning
- Maximum Entropy Approach & Other Methods
- Issues of Large Scale Training
- Objective Functions
- System Combination

Introduction

- Earlier modeling approaches were generative.
- We break the problem into smaller steps and Build component models using MLE.
- Decomposition is done using Bayes' rule.
- We now move into a new paradigm.

The question is:

Can we directly optimize translation performance?

The New Approach

Based on Machine Learning.

The machine is made to learn between *good* and *bad* translations.

Then the machine is fine tuned so that it can pick a good translation for an input sentence.

The selector is commonly called ranker –
which picks from among a set of given translations the *best* one.

How to achieve this?

The New Approach

The key issue is representation of a translation.

Each candidate solution is represented using a set of features $f_1, f_2, .. f_n$

Each f_i describes some property of the Translation

Each feature has some weight ($w_1, w_2, .. w_n$) to show its Relative importance.

For the best translation we expect *optimal* value for $w_1 f_1 + w_2 f_2 + \ldots\ldots + w_n f_n$

The New Approach

Thus there are the following issues:

- How to find good candidate translations.
- How to represent a translation using features.
- How to tune the parameters
- How to manage a large scale discriminative training

Finding Good Candidate Solutions

Finding Candidates

Important – as we need to see the characteristics of Good translations.

Different algorithms have been developed:

The output is a list of N translations – N-best list.

Graph Search

- The Beam-search algorithm produces a graph of partial translations.
- All the leaf nodes are complete translations produced by Hypothesis expansion .
- Each node is associated with a score.
- Beam Search returns the one with the highest score.
- But it also retains a few other top scorers!
- These are all good potential candidates.

Finding Candidates

One can think of many other schemes e.g:

-Word Lattice (Weighted Finite State Automata)

- Use of Simple Syntactic rules after Word Alignment

To come up with a set of Possible, Good Translations.

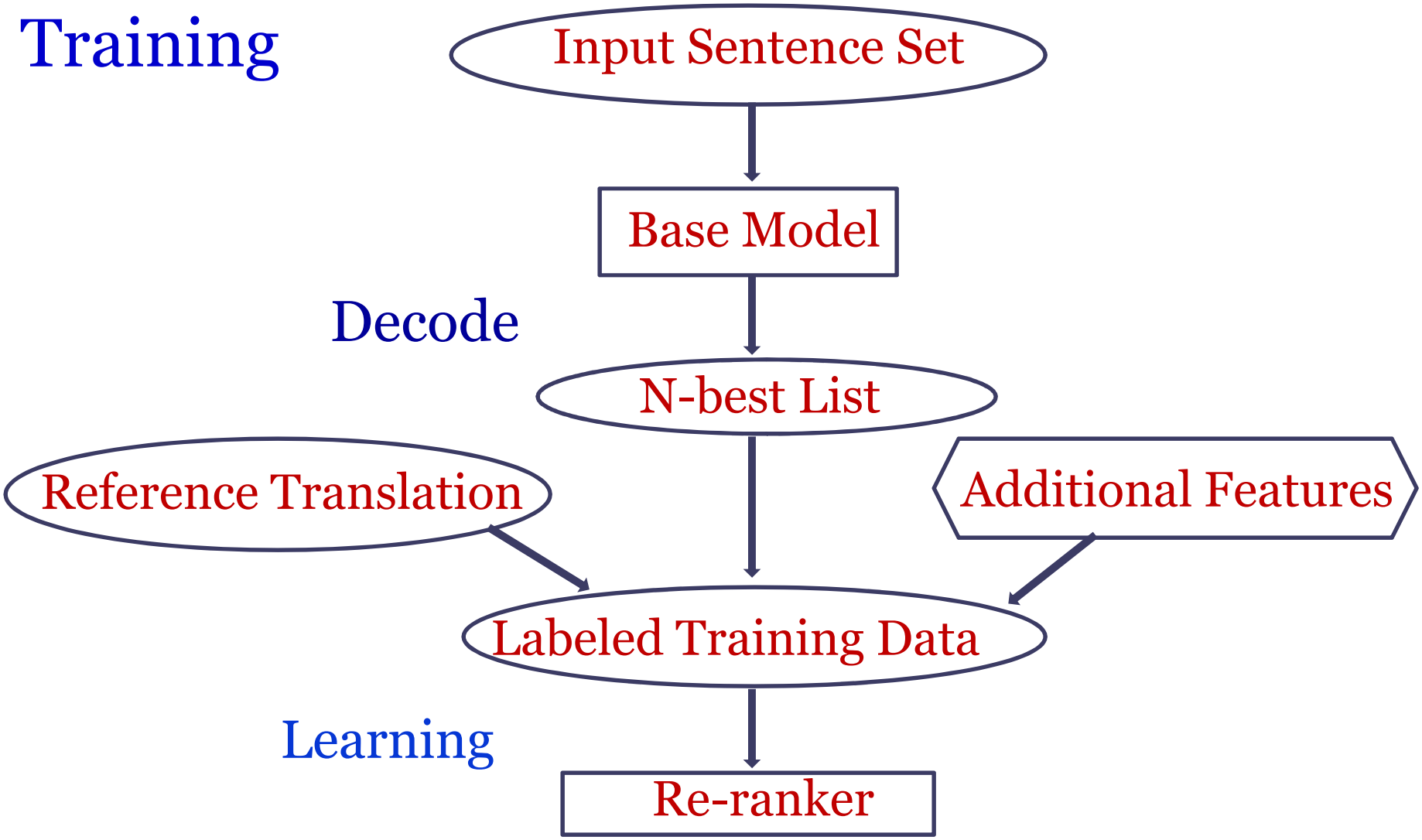
The Overall Principle

The Basic Scheme

Training Phase

- A Base Model is used to represent all the sentences using Features.
- A Decoder then picks the N-Best list.
- The system is trained on a corpus of training samples.
- Reference Translations and Additional features (if any) are then used to label the sentences.
- A Re-ranker is then trained on this labelled data to get the correct weights of the features – essentially so that it can pick the best translation.

The Basic Scheme



The Basic Scheme

Testing Phase

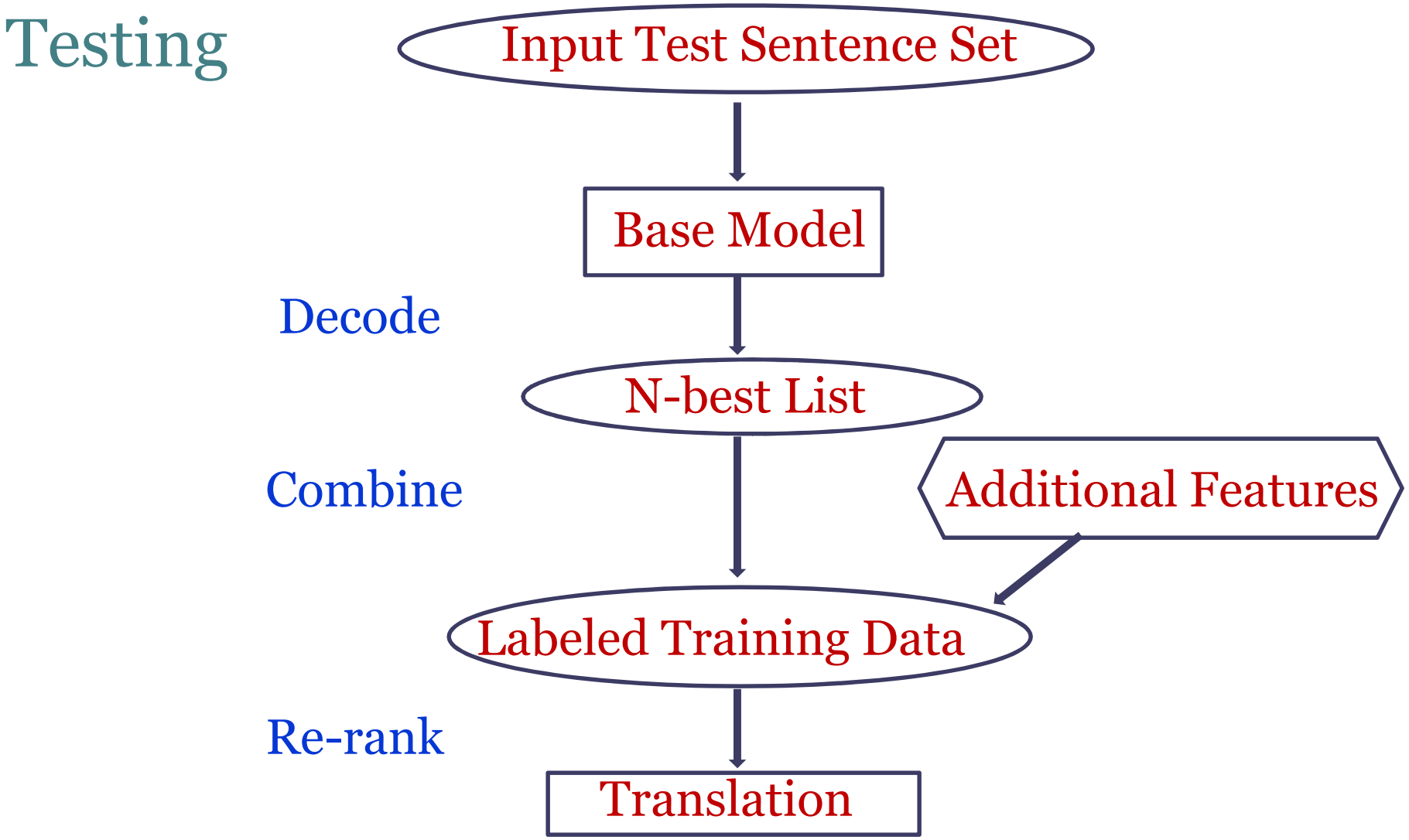
- Again using Base Model all the sentences are represented using Features.

- Using a Decoder the N-best list are identified

- These list along with Additional features are fed to the Re-ranker

- Re-ranker then picks the best translation.

The Basic Scheme



Representation Using Feature

What is a Good Feature

This is a very difficult problem to deal with!!

Breaking a sentence into smaller Components should give a better result

Earlier approaches have used different sub-models:

-Translation Model – how words and phrases match between Input and Output.

- Language Model - how to format the Output well

- Reordering Model - likelihood of movements

All above can be used as effective Translation Features

What is a Good Feature

Further, one can think of many surface level Features:

- the length of the translation (compared to the input sentence length)
- Co-occurrences of certain sets of words.
- Phrase/words translation probabilities.
- Simple linguistic properties (e.g. articles, addition / deletion of words, etc.)

Accordingly we can define features.

What is a Good Feature

Consider translating the sentence:

He parked his car near the park.

Let us do the exercise of finding some good features!!

Features are described in the form of Feature Function
In order to give numeric values to non-numeric features

What is a Good Feature

We can have *feature functions* described as :

- $h_1(x) = h_1(e, f, a) = P_{LM}(e)$
- $h_2(x) = h_2(e, f, a) = \Pi_i \varphi(\bar{f_i} \mid \bar{e_i})$
- $h_3(x) = h_3(e, f, a) = \begin{cases} 1 \; if \; \exists \, e_j \in e \ni e_j \; is \, a \; verb \\ 0 \qquad\qquad\qquad\qquad Otherwise \end{cases}$
- $h_4(x) = h_4(e, f, a) = \sum_{i,j} \delta(e_i = bark) \, \delta(e_j = dog)$
- $h_5(x) = h_5(e, f, a) = \sum_{i,j} \delta(f_i = me \; piace) \, \delta(e_j = I \; like)$

Methods for re-ranking and Parameter Tuning

We shall discuss two common methods;

- Supervised Learning

- Maximum Entropy

Supervised Learning

Once we know how to represent a translation using features, the task of MT is *Supervised Learning.*

So we have feature-set $\{h_i\}$ for each example.

Each example $x \equiv$ **(e, f, a)**

Our task is to discover the evaluation function

$g: x \in X$ or $g(x) = g(h_1(x), \ldots, h_n(x)) \rightarrow$ y,

Where $\text{y} = \begin{cases} 1 \text{ if } \mathbf{e} \text{ is a correct translation of } \mathbf{f} \\ 0 \quad \text{Otherwise} \end{cases}$

Supervised Learning

Generally we like to have a probabilistic model.
giving likelihood that example $x \in$ *CORRECT*

The re-ranker picks the one with the highest likelihood
From the N-best list.

Generally used: - log-linear function:

$$g(x) = exp \sum_i \lambda_i h_i(x)$$

The λ s are learnt so that the training set error
is minimum.

Maximum Entropy

Och and Ney have used this technique for Optimal training.

Recall that: ME recommends the model that gives highest Entropy.

i.e. the one that gives uniform distribution.

What does it mean in this domain??

Maximum Entropy

Here statistical observations are used as features.

Consider the English pronoun : he

Suppose 4 Italian words are mapped into the word :he
These are egli, lui, colui and quello.

Under uniform distributions we have:
p(he | egli) = p(he | lui) = p(he | colui) = p(he | quello) = 0.25

We can have a feature :

$$h_1(x,y) = \begin{cases} 1 & if \;\; x\, is\, "he" \;\&\; y \in \{egli, lui, colui, quello\} \\ 0 & Otherwise \end{cases}$$

Maximum Entropy

But suppose the training data suggests that actually *egli* occurs 40% of the time. Then the reamining probability will be distributed uniformly over the rest of the dataset.

In order to give some more weight to *egli* we can think of another feature:

$$h_2(x,y) = \begin{cases} 1 & if \;\; x \; is \;\; "he" \;\&\; y = "egli" \\ 0 & \quad Otherwise \end{cases}$$

Thus statistics from training data are used as constraints for this model.

In general feature values can be any non-negative Reals for the following learning algorithm

Maximum Entropy

We want that the expected value for h matches the empirical distribution.

Now $\tilde{p}(h) = \sum_{x,y} \tilde{p}(x,y)\, h(x,y)$

Where $\tilde{p}(x,y)$ is the empirical joint distribution.

Note: (1) Any statistic of the sample can thus be expressed as an expected value of an appropriate Binary-value Indicator function.

(2) Whenever a useful statistic is discovered, its importance can be acknowledged by forcing the model to go accordingly.

This is done by constraining the expected value that the Model assigns to the corresponding feature function h

Maximum Entropy

The Expected value of h with respect to the model $p(y|x)$ is:

$$p(h) = \sum_{x,y} \tilde{p}(x)\, p(y|x)\, h(x,y)$$

Where $\tilde{p}(x)$ is the empirical distribution of x in the sample.

We constrain this expected value to be same as the one Got from the training sample – i.e.

$$p(h) = \tilde{p}(h)$$

Combining all these we get the constraint equations

$$\sum_{x,y} \tilde{p}(x)\, p(y|x)\, h(x,y) = \sum_{x,y} \tilde{p}(x,y) h(x,y)$$

Maximum Entropy

The ME principle suggests that the expected value and the empirical distribution matches for each feature.

Otherwise: it should be Uniform distribution

There are different learning algorithms for training an ME model.

E.g. Iterative Scaling Algortihms

•J.N.Darroch and D.Radcliff , "Generalized Iterative Scaling for Log-linear Models", Flinders University of South Australia and C.S.I.R.O. Adelaide, The Annals of Mathematical Statistics, Vol.43,No.5,1470– 1480, 1972.

Maximum Entropy

Typically, $p(y \mid x)$ takes log-linear form:

$$p(y|x) = \frac{1}{Z(x)} \exp \sum_i \lambda_i h_i(x, y)$$

Where, Z(x) is a Normalization Constant so that $\Sigma_y p(y \mid x) = 1 \, for \, all \, x.$

The learning task is to estimate the λ_is.

Start with some initial values for λ_i , $\forall$ I (Say = 0)
Then some iterative scaling scheme is applied
To adjust the values, until they converge.

More on Parameter Tuning

Parameter Tuning

ME is one machine learning method used to learn feature weights.

There are other schemes also. E.g.

- Powell search

All the optimizations are aimed at choosing to find best λ_is so that we get optimal feature weights, where The probability of a sentence is represented in terms of Feature weights:

The model here is:

$$p(x) = \exp \sum_i \lambda_i h_i(x, y)$$

Parameter Tuning

Usually the huge training set is not used.

A representative set of sentences are selected.

Also the features set size is typically restricted to a small number - 10 to 15.

Simplex algorithm has also been used in some systems.

Parameter Tuning - **Powell search**

Analytically finding functions which map parameters To translation error score is complex, and costly to compute because of huge dimension.

Also note that we are in discrete values, and hence Taking derivatives is meaningless.

However, the insight is that the number of possible optimal values is relatively small.

Hence a systematic search produces the result.

Parameter Tuning - **Powell search**

In Powell Search one parameter is tuned at a time.
Let us call it λx

As the other values are kept constant, varying one parameter allows to represent each sentence as a line.

Note that

a. At the intersection of two lines two different Translations have the same score, for the same value of the parameter under consideration viz. λx.
b. The score of the translation also can change at some intersection point only.

This insight is exploited in the Powell Search.

Large-Scale Discriminative Training

Large-Scale Discriminative Training

Different methods have been proposed for Fine tuning the parameters

Typically they work well on reasonable set of parameters: say around 15.

But realistically we are talking about a large no. of parameters ~ $O(10^6)$

Hence the problem of determining how useful a parameter is also becomes relevant.

Large number of parameters also means that we have to deal with a large data set.

Large-Scale Discriminative Training

This gives new sets of issues:

- Problem of scaling
- Mismatch with reference
- Problem of Over-fitting

All these give rise to more novel *Machine Learning* techniques.

This opens up a lot of research areas.

Scaling

To model an effective system we need to train
The model on a large volume of parallel sentences

- Size of parallel corpus ~ order of millions.
- Suppose we consider 100-best translations.
- There can be few scores of features per sentence

Thus we have to deal with of the order of billions!!

Fortunately a lot of computing can be done in parallel.
Hence clusters of computers can be used.

Problem of Mismatch

What happens if no member of the N-best list matches The single reference?

This can happen because of a large number of reasons:

- Different ambiguities (we talked in the first class)
- Different styles of presentations
- Even because of structure of the input sentences.

For illustration consider the following sentences from Europarl:

It is very unlikely to have an exact match.

Examples

- Mr President, Mr President of the Commission, Mr President-in-Office, ladies and gentlemen, I should like to thank the European Parliament delegation in the Convention and the President of the Convention, Mr Roman Herzog, for their committed work.
- In Mr Gawronski' s report, the Commission is urged, by the rapporteur among others, to examine measures for furthering development and for strengthening the synergy between Poland, Lithuania and the Russian region of Kaliningrad.
- It was only logical for the European Council in Helsinki in December last year to give the green light for the resumption of accession negotiations with Malta when the accession negotiations were extended.

Translation Examples

- Signor Presidente, signor Presidente della Commissione, signor Presidente in carica del Consiglio, onorevoli colleghi, vorrei ringraziare la delegazione del Parlamento europeo alla Convenzione e al Presidente della Convenzione, Roman Herzog, per l'impegno profuso.
- In Gawronski 's relazione, la Commissione Ã¨ invitata, dal relatore, tra l'altro, di esaminare misure per favorire lo viluppo e per rafforzare la sinergia tra la Polonia, la Lituania e la regione russa di Kaliningrad.
- Era logico che il Consiglio europeo di Helsinki nel dicembre dello scorso anno per dare il via libera per la ripresa dei negoziati di adesione con Malta, quando i negoziati di adesione sono state estese.

Problem of Mismatch

Various options have been suggested:

- Discard the example.
  (May end up with discarding all complicated sentences!!)

- Find substitute reference
  (Not easy to implement – can be done if the style of translation of the system is well known.
  But identifying the pattern is difficult.)

- Early updating of parameters.
  (as soon as it is clear that Beam search is going to fail)

Over-fitting

In spite of al the trainings often a model fails when Applied to Test set.

Typically:

- Smaller phrases are well translated – but problem in stitching them together.
- Larger phrases are translated well – but rare to occur.

This occurs due to assigning more weight on **Insignificant Features.**

- discard the features which occur in less number of examples?
- smooth the data - but how?

Objective Function

Prologue

In Maximum Entropy learning we want to match the empirical feature values with the Expected values from the training data.

Can we be different?

Suppose we have some measure of error that a translation makes w.r.t the reference.

We like to optimize some function – so that we can directly choose the best translation.

Such a function is called “loss Function”

How to define a Loss function

Let f_i be the ith source sentence.

Let e_{ij} be the j-th translation in the N-best list for f_i .

Let λ' be a parameter setting.

Let $S(e_{ij}, \lambda')$ be the score given to a translation e_{ij} of f_i .

The e_{ij}s are in decreasing order of the score with respect to the reference translation $e_{i,ref}$

How to define a Loss function

Sentence error function.

Counts for how many f_i s the actual correct Translation can be obtained?

$$\textbf{SENTENCEERROR}\ (\lambda) = \sum_i TRUE\,(e_{i,1} \neq e_{i,\,ref})$$

Note: TRUE is 1 if the statement is correct.

Of course this is not a very effective function
Particularly for long sentences

How to define a Loss function

Ranking error function.

Checks where the reference translation stands with Respect to other candidate translations:

$$\textbf{RANKINGERROR}\ (\lambda) = \sum_i \sum_j TRUE(S(e_{i,j,}\ \lambda) > S(e_{i,ref,}\ \lambda))$$

Many such functions can be formed

How to define a Loss function

Likelihood function.

$$\text{LIKELIHOOD } (\lambda) = \prod_{i} \frac{S(e_{i,ref}, \lambda)}{\sum_{j} S(e_{i,j}, \lambda)}$$

Logloss function
(Reverse of the above – Here aim is to minimize).

$$\text{LOGLOSS}(\lambda) = \sum_{i} -\log \frac{S(e_{i,ref}, \lambda)}{\sum_{j} S(e_{i,j}, \lambda)}$$

Other techniques

Once an Objective function is defined different other Techniques are also being tried to obtain optimal λ.

e.g.

- Gradient Decent
- Perceptron

Gradient Descent

Gradient descent is a 1st order optimization algorithm To find a local minimum of a function using *gradient descent,* one takes steps proportional to the *negative* of The gradient (or of the approximate gradient) of the Function at the current point.

(WiKipedia)

If instead one takes steps proportional to the gradient, one approaches a local maximum of that function; the procedure is then known as **gradient ascent**.

Gradient Descent

- An iterative strategy.
- Start with some initial values of the parameters.
- Each iteration changes the values so that error is reduced.
- Essentially we differentiate the ERROR($\bar{\lambda}$) to determine the direction of movement in the high dimensional Parameter space:

$$\text{UPDATE}(\bar{\lambda}) = -\Delta \text{ERROR}(\bar{\lambda}) \,/\, \Delta(\bar{\lambda})$$

- Can be computed for each single parameter separately

$$\text{UPDATE}(\lambda_i) = -\mu_i \, \delta\text{ERROR}(\bar{\lambda}) \,/\, \delta(\lambda_i)$$

where, μ_i is called the Learning Rate.

Perceptron

Perceptron is a binary classifier that maps its input x (a real-valued vector) to an output value $f(x)$ (a single binary value) across the matrix.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

Wikipedia

w is a vector of real-valued weights.

b is the 'bias', a constant term.

The following algorithm shows how it can be used for learning the weights.

Perceptron: Learning

Let the training set contains m sentences.

Set initial weights = 0

Until convergence do

For each sentence **f**

If $e_{best} \neq e_{ref}$ then

for all features x_j set

$$\lambda_j = \lambda_j + x_j(\mathbf{f}, e_{ref}) - x_j(\mathbf{f}, e_{best})$$

Mathematically it has been proved that the Weights converge under some general conditions

Posterior Methods

Posterior Methods

These are the methods that work on

- posterior probabilities.
- the different high probability outputs

Implies we are not looking at the probability of a translation - rather We look at the similarity between the most likely translations.

Alternatively we try to combine the outputs of different systems

Minimum Bayes Risk

A Decoder typically picks the translation that Maximizes the probability:

$$\text{argmax}_{\mathbf{e}}\, p(\mathbf{e}, a \mid \mathbf{f})$$

But note that the same translation can be achieved thru Different alignments.

E.G *non parlo italiano* >> I do not speak English

The translation can be achieved using:

1) *non* >> not *parlo* >> I do speak
2) *non* >> do not *parlo* >> I speak
3) *non parlo* >> do not speak ϕ >> I

Etc.

With different probabilities

Minimum Bayes Risk

Hence it is better if instead of relying on one Best alignment we look at the sum of probabilities of al the good translations

The MBR approach is aimed at achieving this - which Means we are looking at a consensus candidate.

Essentially here similarity is measured between Different translations and an weighted average is Computed.

The Error function can be designed in many ways.

Minimum Bayes Risk

Formally, the decision rule for Bayes risk decoding is:

$$\mathbf{e}_{\text{MBR-best}} = \text{argmax}_{\mathbf{e}} \; \Sigma_{e',a'} \, L((e, a), (e',a')) \, p(e',a' \mid f)$$

The translation selected by the above rule can be considered as a consensus candidate.

Note:

1) L is the Loss function - used for measuring similarity between translations
2) Any error measurement scheme (e.g. BLEU) which gives the correctness of one translation in comparison with a reference translation can be used as a Loss Function.
c) The traditional maximum a posteriori decoder is a special case of the above if we consider L((e, a), (e',a')) as $\delta_{e,e'} * \delta_{a,a'}$

Of course the search is conducted on a manageable set of translations.

System Combination

System Combination

This can be applied when more than one system is available.

One can combine their outputs to make a better one: e.g. by sharing components; sharing resources.

Let us discuss sharing components.

Consider the example from Koehn.

We have three translations:

It does not go home.
He goes not to the house
He does not go house.

Is it possible to combine them to get a better translation?

The question is how to combine them?

System Combination

This is done using Confusion network.

- Here all the words of all the translations are considered in a Sequence.
- The sequences are merged in such a way that words of the same concept are at the same position:

E.g.

It_1 $does_2$ not_3 go_4 $home_7$
He_1 $goes_4$ not_3 to_5 the_6 $home_7$
He_1 $does_2$ not_3 go_4 $house_7$

The linearization gives the following with the probability at each location -

It 0.33	Does 0.67	Not 1.0	Go 0.67	To 0.33	The 0.33	Home 0.67
He 0.67	ε 0.33		Goes 0.33	ε 0.67	ε 0.67	House 0.33

System Combination

One can now get a consensus translation, by selecting a word from each column.

-It is not mandatory to pick the highest probability one.
- System preference can be conveyed through weights.
- Regarding choosing the word order also one may stick to the most preferred one.

Thank you