Statistical Machine Translation

LECTURE – 4

WORD BASED MODELS-1

APRIL 15, 2010

Niladri Chatterjee
IIT Delhi, INDIA

Brief Outline

-Translation by words

- IBM model 1

- Introduction to Higher Models

Introduction

Word models come from the original work at IBM.

The MT technologies have advanced since then.

These works helps us to understand the foundations of SMT and its techniques.

IBM has proposed five models - with gradually Improving versions.

Ref: The mathematics of Statistical machine Translation: Parameter Estimation - Peter F Brown et.al - Computational Linguistics, Vol 19, No. 2, 1993

MT by Translating Words

In the simplest form: It is *Lexical Translation*

A string can be translated by translating each word of the *source text* to the *target text*.

However, there is a difficulty:

A source language word may have more than one translation in the target language:

Haus (G) → House, Home, Household, Building (Eng)
→ *ghar, bhavan, mahal, prasad* ... (Hindi)

How to choose the best one?

MT by Translating Words

How about computing the statistics?

After scanning a large number of documents we can estimate probability of each of the translations!!

(Question: How to do this?)

How does it solve our purpose?

We can use the probabilities of the individual words of a foreign language text **f** to determine the most probable translation in the language **e**.

MT by Translating Words

A foreign language sentence may have words:

f1 f2 … fn

Each has its own choice of alternatives, and corresponding translation probabilities :

$t(e \mid f)$ - Prob. that word *f* translates into word e where *e* is a word in the target language

These t 's are called Translation Probabilities

MT by Translating Words

For example consider the following tables of translation probabilities (hypothetical):

yah	
this	0.5
the	0.3
that	0.1
–	0.1

makaan	
house	0.4
beautiful	0.3
bungalow	0.2
residence	0.075
flat	0.025

sundar	
beautiful	0.45
nice	0.3
pretty	0.15
cute	0.1

hai	
is	0.75
exists	0.15
remains	0.1

What is the most likely translation of :

yah makaan sundar hai?

Word Alignment

A word-by-word translation gives us :

this house beautiful is

Thus implicitly we are using a mapping from the foreign words to the English words.

Depending on the grammar we can have different mappings. In this case we have:

$1 \longrightarrow 1,\quad 2 \longrightarrow 2,\quad 3 \longrightarrow 4,$ and $4 \longrightarrow 3$

The correct one is : this house is beautiful

Word Alignment

Other than a permutation, word alignment may suffer from Alignment pattern:

0 - 1 : das haus ist **ja** klein → the house is small

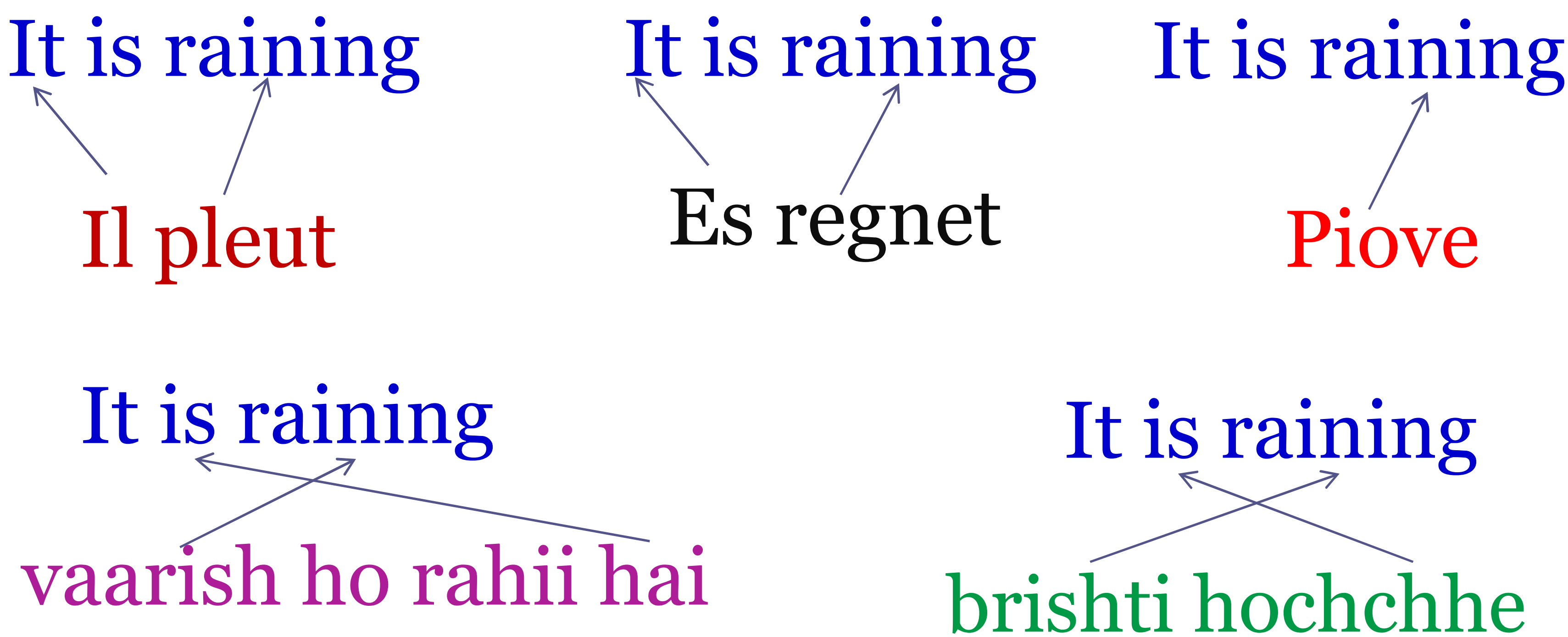
2 – 1 : das haus ist **klitzeklein** → the house is **very small**

1 – 0 : ich gehe ja nicht zum haus → **I do** not go to the house

etc.

Word Alignment

And it varies with language pairs:



Word Alignment function

- An alignment is best represented using an alignment function.
- It maps for each word of the Target Language to a word of the Source language

E.G

das haus ist klein

↑ ↑ ↑ ↑

the house is small

$$a: \{1 \rightarrow 1,\ 2 \rightarrow 2,\ 3 \rightarrow 3,\ 4 \rightarrow 4\}$$

Note: Alignment function is from target to source.

Word Alignment function

E.G 2.

das haus ist ja klein

the house is small

$$a: \{1 \longrightarrow 1,\ 2 \longrightarrow 2,\ 3 \longrightarrow 3,\ 4 \longrightarrow 5\}$$

E.G 3.

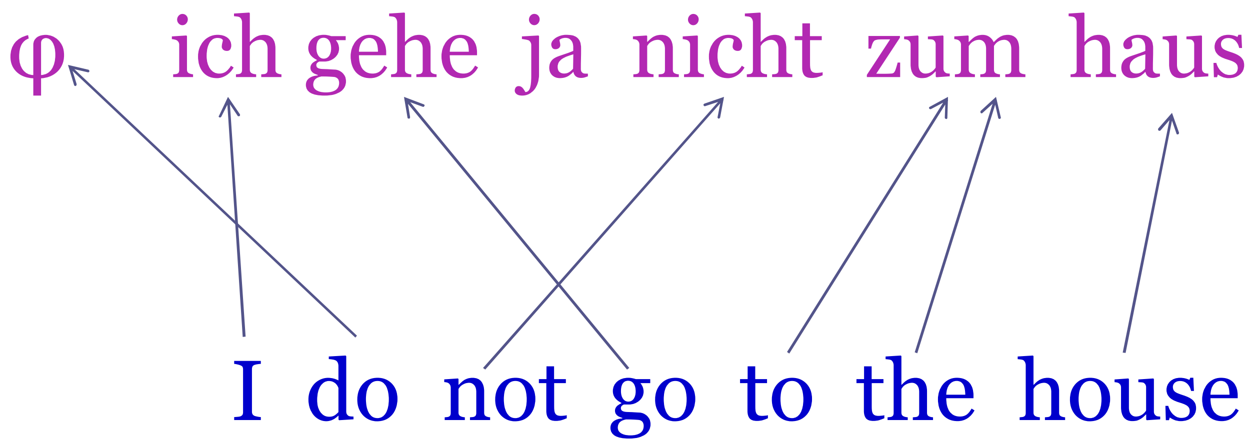
das haus ist klitzeklein

the house is very small

$$a: \{1 \longrightarrow 1,\ 2 \longrightarrow 2,\ 3 \longrightarrow 3,\ 4 \longrightarrow 4,\ 5 \longrightarrow 4\}$$

Word Alignment function

E.G 4



$a: \{1 \rightarrow 1,\ 2 \rightarrow 0,\ 3 \rightarrow 4,\ 4 \rightarrow 2,\ 5 \rightarrow 5,\ 6 \rightarrow 5,\ 7 \rightarrow 6\}$

Alignment function will be needed in the models

IBM Model 1

IBM Model 1

The simplest of the five IBM models.

Produces different translations of a sentence with associated probabilities

It is a Generative modeling - i.e.

- breaks up the translation process into smaller steps
- calculate their probabilities
- combine them into a coherent piece of text
- based on lexical translation probabilities only

It is hard to get the distribution of a full sentence !!

So we go word by word.

IBM Model 1

Input: foreign sentence $\mathbf{f} = f_1 f_2 \ldots\ldots\ f_n$

The probability of it being translated into

$\mathbf{e} = e_1\, e_2 \ldots.\ e_m$ is?

Given an underlying alignment function

$a:\ j \rightarrow i$ (e_j is aligned with f_i)

where :

$$p(\mathbf{e}, a \mid \mathbf{f}) = \frac{c}{(n+1)^m} \prod_{j=1}^{m} t(e_j \mid f_{a(j)})$$

IBM Model 1

- n is the number of words in **f**.
- m is the number of words in ***e***
- each a_i can map into any of the n words in **f** viz. f_1 .... f_n and φ
- Joint probability of ***e*** and an alignment a given **f** is computed by the product of the individual translation probabilities $t\,(e_j \mid f_{a(j)})$
- c is a normalization constant.

IBM Model 1

It implicitly assumes that word alignments are known – and hence could calculate ***t***.

But this is NOT true.

Need to learn *word alignments* from the data itself

Thus we have a chicken-&-egg problem!!

- If word alignments are known we can estimate the translation probability of a sentence.

  or

- If the model is given we can estimate the likely alignments.

IBM Model 1

Parallel texts are used for this learning.

The most common technique to learn from incomplete data is Expectation-Maximization (EM) Algorithm

EM algorithm is nicely interwound into IBM models

Learning from Data

We need to find alignment *a* from **e** and **f**.

Example.

mangsho aami bhaalobasi → I like meat
aami roj phal khaai → I take fruits daily
phal tumi bhaalobaso → You like fruits
tumi maangsho raandho → You cook meat

Can we learn the alignment??
Let us make a try.

EM algorithm works on a similar principle.

EM-Algorithm

- Given by Dempster, Laird and Rubin 1977.

- The EM algorithm has become a popular tool.

- It is used in statistical estimation problems involving incomplete data

- Iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing or hidden data (ladden variables)

EM-Algorithm

Iterative procedure – consisting of 2 steps:

E-step - we estimate the missing data
from - 1. the observed data
2. current estimate of the model parameters.

M-Step - we maximize the likelihood function
assumption –
1. missing data are known.
2. estimate of the missing data from the E-step are used in lieu of the actual missing data.

EM-Algorithm

Let X be random vector which results from a Parameterized family. We wish to find θ s.t. $P(X|\theta)$ is a maximum.

In problems where missing variables exist, the Expectation Maximization (EM) Algorithm provides a natural framework for their inclusion.

Let Z denote the hidden random vector and a given realization by z. The total probability $P(X|\theta)$ may be written in terms of the hidden variables z as,

$$P(X|\theta) = \sum_z P(X|z, \theta)P(z|\theta)$$

The similar concept is applied here with ***e***, ***a***, ***f***

EM-Algorithm for IBM 1

$p(\mathbf{e} \mid \mathbf{f}) =$

$= \Sigma_a \, p(\mathbf{e}, a \mid \mathbf{f})$

$$= \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} p(\mathbf{e}, a \mid \mathbf{f})$$

Q: What is the Complexity?

$$= \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} \frac{c}{(n+1)^m} \prod_{j=1}^{m} t(e_j \mid f_{a(j)})$$

EM-Algorithm for IBM 1

$$= \frac{c}{(n+1)^m} \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} \prod_{j=1}^{m} t(e_j | f_{a(j)})$$

$$= \frac{c}{(n+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{n} t(e_j | f_i)$$

Thus complexity is Reduced to O(mn) i.e. Roughly quadratic.

The sum of $(n+1)^m$ terms -> product of m terms each Being a sum of (n+1) terms

EM-Algorithm for IBM 1

We now estimate the probability of a given alignment a, given $\mathbf{e}$ and $\mathbf{f}$:

$$p(a|\mathbf{e},\mathbf{f}) = \frac{p(\mathbf{e},a|\mathbf{f})}{p(e|\mathbf{f})} = \frac{\frac{c}{(n+1)^m}\prod_{j=1}^{m} t(e_j | f_{a(j)})}{\frac{c}{(n+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{n} t(e_j | f_i)}$$

$$= \frac{\prod_{j=1}^{m} t(e_j | f_{a(j)})}{\prod_{j=1}^{m}\sum_{i=0}^{n} t(e_j | f_i)}$$

EM-Algorithm for IBM 1

We wish to adjust the transition probabilities $t(e_k | f_l)$ s.t. for each foreign word f_j

$$\sum_i t\left(e_i \mid f_j\right) = 1, \; j = 1, \quad , n$$

We use Lagrange Multiplier technique for this purpose

Lagrange Multiplier

In mathematical optimization, the method of Lagrange multipliers (named after Joseph Louis Lagrange) provides a strategy for finding the *maximum* /*minimum* of a function subject to constraints.

[Wikipedia]

For example suppose we want to maximize:

$$f(x,y) \text{ s.t. } g(x,y) = c$$

We introduce a new variable (λ) called a Lagrange multiplier, and study the Lagrange function defined by:

$$h(x,y,\lambda) = f(x,y) + \lambda\,(g(x,y) - c)$$

Lagrange Multiplier

In this case we have n constraints – each pertaining To a SL word f_j - Let us call it λ_j

So we are looking at a function of translation Probabilities t( ) and the λ s.

$$h(t,\lambda) = \frac{c}{(n+1)^m} \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} \prod_{j=1}^{m} t(e_j | f_{a(j)}) - \sum_{q} \lambda_q \left(\sum_{p} t(e_p | f_q) - 1 \right)$$

In order to get an extremum we have to differentiate w.r..t all the variables – i.e. All the $t(e_i \mid f_j)$ and λ_j

EM-Algorithm for IBM 1

Differentiating $h(t, \lambda)$ *w.r.t* $t(e_p \mid f_q)$ and equating with o, we get

$$\frac{\partial h}{\partial t(e_p \mid f_q)} = 0 =$$

$$\frac{c}{(n+1)^m} \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} \sum_{i=1}^{m} \delta(e_p, e_i)\, \delta(f_q, f_{a(i)})\, t(e_p \mid f_q)^{-1} \prod_{k=1}^{m} t(e_k \mid f_{a(k)}) - \lambda_q$$

Hence

$$t(e_p \mid f_q) = \lambda_q^{-1} \frac{c}{(n+1)^m} \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} \sum_{i=1}^{m} \delta(e_p, e_i)\, \delta(f_q, f_{a(i)}) \prod_{k=1}^{m} t(e_k \mid f_{a(k)})$$

This appears to be a solution - but it is NOT. Why?

EM-Algorithm for IBM 1

However, this gives us an iterative way of solving the equations – starting with some default values.

Putting $p(\mathbf{e}, a | \mathbf{f}) = \frac{c}{(n+1)^m} \prod_{j=1}^{m} t(e_j | f_{a(j)})$

We have $t(e_p | f_q)$

$$= \lambda_q^{-1} \sum_a p(e, a | f) \sum_{i=1}^{m} \delta(e_p, e_i) \delta(f_q, f_{a(i)})$$

This probability computation helps us to fill the gap due to *incomplete data* in the E-step.

EM-Algorithm for IBM 1

In the M-step we update as follows:

- Count the word translations over all possible Alignments by choosing their estimated probabilities as their weights

- This can be done using a *count function*: which computes for a sentence pair **(e, f)** the evidence that a particular word f_q gets translated into a word e_p.

NOTE: e_p may occur more than once in **e**, and so is f_q in **f**

EM-Algorithm for IBM 1

Thus the count function is defined as follows:

$$count(e_p \mid f_q; \mathbf{e}, \mathbf{f})$$

$$= \sum_a p(a \mid \mathbf{e}, \mathbf{f}) \sum_{j=1}^{m} \delta(e_p, e_j) * \delta(f_q, f_{a(j)})$$

Where the last sum suggests the number of times e_p connects with f_q in the alignment a

EM-Algorithm for IBM 1

Now, p(a | **e**, **f**) = p(a, **e** | **f**)/ p(**e** | **f**)

Hence, $t(e_p \mid f_q)$ can be compactly written as:

$$t(e_p | f_q) = \lambda_q^{-1} \sum_a p(\mathbf{e}, a | \mathbf{f}) \sum_{i=1}^{m} \delta(e_p, e_i)\,\delta(f_q, f_{a(i)})$$

$$= \frac{\lambda_q^{-1}}{p(\mathbf{e}|\mathbf{f})} \sum_a p(a | \mathbf{e}, \mathbf{f}) \sum_{i=1}^{m} \delta(e_p, e_i)\,\delta(f_q, f_{a(i)})$$

$$= \lambda^{-1} count\ (e_p | f_q; \mathbf{f}, \mathbf{e})$$

Where λ is the normalizing constant.

EM-Algorithm for IBM 1

Thus we get a relationship between the *transition probabilities* and count.

However, this has been w.r.t only one sentence pair (**f**, **e**). But in practice we have many such pairs – say S in number.

Thus $t(e_p \mid f_q)$ can be estimated as

$$\frac{\sum_{(e,f)} count(e_p \mid f_q; \mathbf{e}, \mathbf{f})}{\sum_{e_w} \sum_{(e,f)} count(e_w \mid f_q; \mathbf{e}, \mathbf{f})}$$

EM-Algorithm for IBM 1

Input: S sentence pairs (f, e) Output: Translation probabilities $t(e_i \mid f_j)$

S1: Choose initial values for $t(e_i \mid f_j)$

S2: For each pair of sentences $((\mathbf{f}^{(s)}, \mathbf{e}^{(s)}), s = 1, 2, \ldots S)$ do compute $count(e_i \mid f_j; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})$

(Note: count will be non-zero only if $e_i \in \mathbf{e}^{(s)}$ and $f_j \in \mathbf{f}^{(s)}$)

S3: for each f_j that appears in at least one $\mathbf{f}^{(s)}$)

compute λ_j using $\sum_{e_w} \sum_{(\mathbf{e}^{(s)}, \mathbf{f}^{(s)})} count(e_w \mid f_j; \mathbf{e}^{(s)}, \mathbf{f}^{(s)})$

S4: for each e_i that appears in at least 1 $\mathbf{e}^{(s)}$)

compute new $t(e_i \mid f_j)$ using:

$$\frac{\sum_{(e,f)} count(e_p \mid f_q; \mathbf{e}, \mathbf{f})}{\sum_{e_w} \sum_{(e,f)} count(e_w \mid f_q; \mathbf{e}, \mathbf{f})}$$

Note: Complexity: Linear in S; Quadratic in max(m,n)

S5. Repeat S2 – S4 until the t values converge.

EM-Algorithm for IBM 1

Ex: Write a program to find the t values for the En-Bn pair given below.

bhat aami bhalobasi	→	I like rice
tumi raandho bhat	→	You cook rice
roj aami phal khai	→	I take fruits daily
tumi phal bhalobaso	→	You like fruits

Fluency

Fluency

In Model 1 we do not talk about *context.*

However, same translation of the same word May not appear as *fluent* as in another context.

E.G Consider the Google search frequencies (as in January 2010):

big	– 891,000,000	large	– 701,000,000
small	– 818,000,000	little	– 826,000,000
cute	- 158,000,000	pretty	- 313,000,000
tall	- 83,000,000	long	- 1,070,000,000

Fluency

small step -	1,790,000	little step -	507,000
large crowd -	1,170,000	big crowd -	614,000
big boy -	3,980,000	large boy -	36,500
cute girl -	25,600,000	pretty girl -	4,100,000
tall tree -	376,000	long tree -	80,200

Shows importance of "fluency" for better translation

Hence we need something superior to simple "word model" !!!

This prompts us to go for some additional Modeling on the top of Word Model.

Higher models of IBM

Higher Models of IBM

IBM has proposed a series of models on the top of the Lexical Translation based Model 1.

Model 2: Adds Alignment Model

A more realistic assumption is that probability of connection between words depend on the positions of the words in **f** and **e**, and on the lengths of the strings: n and m.

Higher Models of IBM

Model 3: Adds Fertility Model

Fertility: How many output words an input word produces.

- It is not necessary that each input word produces only one word.
- Some may produce more than 1.
- Some may produce no word!!

Higher Models of IBM

E.g. 1. *John ne Mary se shaadi kii*

John married Mary.

Words *ne* and *se* have no correspondence.

E.g. 2. *phir milenge*

Shall see you later

A model for *fertility* addresses this aspect of translation.

Higher Models of IBM

Model 3: Fertility Model

-The scheme starts by attaching with each word in **f** the number of **e** words that will be connected to it.

- Their positions are determined next.

-Finally, the connections are made.

This model ushers in biggest change in computational process.

- *Exhaustive collection of count* is too expensive.

- Hence sampling techniques are used on *highly probabilistic* alignments.

Higher Models of IBM

Model 4: Adds *Relative Alignment* Model

Here it is argued that the probability of connection Depends on:

- *fertility*
- Identities of the SL (f) and TL (e) words connected;
- Positions of any other e words that are connected to the same f word.

Higher Models of IBM

Model 5: Takes care of *deficiency*

Problem of models 3 and 4 is that they allow multiple output words to be placed at the same position.

Some prob. mass is wasted on Impossible outcomes.

Model 5 keeps track of vacant positions, and allows new words to be inserted only in these positions.

Thus it is an improvement on Models 3 & 4.

Higher Models of IBM

Model 5: Built upon all previous models, fixes their shortcomings.

Model 1 and 2 are computationally simple. The EM algorithm can be computed exactly – as we can make sum over all possible alignments.

Model 1 has unique local maximum of the L function.

Model 2-5 – Do not have unique local maximum. We start with the estimates of previous model.

Model 3 and 4 we have to approximate the EM Algorithm.

Model 5. Computationally feasible.

Word Alignment Based on IBM Models

IBM models for word-based SMT can be used nicely
As word alignment tool.

During EM algorithm fractional counts are collected
Over a probability distribution of possible alignments.
The most probable one is finally taken. It is also called
Viterbi Alignment.

However, the problem here is each **e** word is allowed
To align with only one **f** token at most. Hence it rules
out alignment of one **e** token with multiple **f** tokens.

What is the way out??

Word Alignment Based on IBM Models

The trick is apply IBM models both ways.

Generally -

- Intersection gives very good Precision,
- Union gives very good Recall.

Typically used in information retrieval

We now proceed to discuss the higher IBM models.

Thank you