Statistical Machine Translation

LECTURE – 5

HIGHER IBM MODELS

APRIL 16, 2010

Brief Outline

- IBM Model 2
- IBM Model 3
- IBM Model 4
- IBM Model 5

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

IBM Model 2

IBM Model 2

Model 1 takes no notice of where the words appear in the translation:

E.g. questa casa è bella naturalmente →

Of course this house is beautiful

This house beautiful is of course

Are equally probable under Model 1

Model 2 takes care of this.

IBM Model 2

Alignment Model:

The assumption is that the translation of
f_j to e_i depends upon alignment probability:

$$P_a(j \mid i, m, n)$$

Q1. What does it mean??

Q2. How to compute ??

IBM Model 2

Thus translation is a two-step process:

Lexical Translation step:

modeled by $t(e_p \mid f_q)$

Alignment Step: modeled by $P_a(j \mid i, m, n)$

e.g. Questa casa è bella naturalmente

this house is beautiful naturally

Translation Step

Naturally this house is beautiful

Alignment Step

IBM Model 2

Under this model we have:

$$p(\mathbf{e}, a \mid \mathbf{f}) = c \prod_{i=1}^{m} t(e_i \mid f_{a(i)})\, p_a(a(i) \mid i, m, n)$$

Hence :

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

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

$$= c \prod_{j=1}^{m} \sum_{i=0}^{n} t(e_j \mid f_i)\, p_a(a(j) \mid j, m, n)$$

IBM Model 2

We need to maximize this $p(\mathbf{e} | \mathbf{f})$

Subject to the following constraints:

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

2. $$\sum_{j=0}^{n} p_a(\, j | \, i, m, n \,) = 1, \; i = 1, m$$

Thus we have a larger set of Lagrangian constants

IBM Model 2

The auxiliary function becomes:

$$h(t, p_a, \lambda, \mu) =$$

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

$$- \sum_j \lambda_j \Big(\sum_i t(e_i \mid f_j) - 1\Big) - \sum_i \mu_{imn} \Big(\sum_j p_a(j \mid i, m, n) - 1\Big)$$

To find the extremum we need to differentiate

IBM Model 2

We now need a new count:

$$count(j \mid i, m, n; \mathbf{f}, \mathbf{e})$$

the expected number of times the word in position i of the TL string $\mathbf{e}$ is connected to the word in the position j of the SL string $\mathbf{f}$, given that their lengths are m, n, respectively.

$$count(j \mid i, m, n; \mathbf{f}, \mathbf{e}) =$$

$$\sum_{a} p(a \mid \mathbf{e}, \mathbf{f}) * \delta(j, a(i))$$

IBM Model 2

So here we shall look at groups of sentence –pairs Who satisfy the m and n, criterion. Then we look at the alignment probabilities.

Example: m = 4 n = 3

aami bari jachchhi

I am going home

1>1 2>0 3>3 4>2

tumi ki khachchho

What are you eating

1>2 2>0 3>1 4>3

tomar naam ki

What is your name

1>3 2>0 3>1 4>2

kaal kothay chhile

Where were you yesterday

1>2 2>3 3>0 4>1

IBM Model 2

Then by analogy with the Model 1, we get:

$$p_a(j|i,m,n) = \frac{1}{\mu_{imn}} count(j|i,m,n;\, \mathbf{e}, \mathbf{f})$$

for a single translation, and

$$p_a(j|i,m,n) = \frac{1}{\mu_{imn}} \sum_{s=1}^{S} count(j|i,m,n;\, \mathbf{e}^{(s)}, \mathbf{f}^{(s)})$$

for a set of translations

IBM Model 2

Although apparently the expression for *count* is complicated, we can make it simple, as in the case of Model 1:

$count(j \mid i, m, n; \mathbf{f}, \mathbf{e}) =$

$$\frac{t(e_i \mid f_j)\, p_a(j \mid i, m, n)}{t(e_i \mid f_0)\, p_a(0 \mid i, m, n) + \ldots\ldots + t(e_i \mid f_n)\, p_a(n \mid i, m, n)}$$

IBM Model 2

One can now design an algorithm for Expectation Maximization, as in case of Model 1

IBM Model 3-5

Intermodel Interlude

Models 1 and 2 have been created on the basis of the following generalized principle:

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

$$p(m \mid \mathbf{f})$$

$$* \prod_{i=1}^{m} p(a(i) \mid a(1)..(i-1), e_{1,..,i-1}, m, \mathbf{f})$$

$$* \; p(e_i \mid a(1)...(i), e_{1\,...\,i-1}, m, \mathbf{f})$$

Note that, each $a(j)$ takes value between 0 to n

It works on the following:

Intermodel Interlude

It represents the joint likelihood of e and a as A product of conditional probabilities.

Each product corresponds to a generative process for developing e and a from f.

- Choose the length of the translation e
- Decide which position in f corresponds to e_1 and the identity of e_1 is.
- Do the same for positions 2 to m

IBM Model 3-5

Here we consider the *fertility* of a word in conjunction with the Word model.

How many **e**-*words* a single **f-**word will produce is NOT deterministic.

E.g. Nonostante $_{\text{(It)}}$ >>
despite, even though, in spite of $_{\text{(En)}}$

Consequently, corresponding to each word of **f** we get a random variable $\varphi_{\mathbf{f}}$ which gives the fertility of **f** including 0.

In all models 3 - 5 fertility is explicitly modeled

IBM Model 3-5

Example: Dovete andare il giorno dopo $_{\text{(It)}}$
May have many possible English translations:

- You must go next day
- You have to go the following day
- You ought to be there on the following day
- You will have to go there on the following day
- I ask you to go there on the following day

Can we now get the alignment?
Look at the *fertility* of the source words.

IBM Model 3-5

Fertility:

1) we can assume that the fertility of each Word is governed by a probability distribution $p(n \mid \mathbf{f})$

2) Deals explicitly with dropping of input words, by putting n = 0.

3) Similarly we can control words in TL sentence that have NO equivalent in **f** – calling them NULL words.

Thus models 3 -5 are Generative Process – given a **f**-string we first decide the fertility of each word and a list of **e**-words to connect to it. This list is called a Tablet.

IBM Model 3-5

Definitions:

Tablet: Given a **f** word the list of words that may connect to it.

Tableau: A collection of Tablets.

Notation:

T – tableau for **f**.

T_j – tablet for the j^{th} **f**-word.

T_{jk} – k^{th} e-word in j^{th} tablet T_j .

IBM Model 3-5

Example: **Come ti chiami$_{(It)}$**

Tableau

Come	ti	chiami
Tablet 1 (T_1)	Tablet 2 (T_2)	Tablet 3 (T_3)
T_{11} = Like	T_{21} = you	T_{31} = call
T_{12} = What	T_{22} = yourself	T_{32} = Address
T_{13} = As	T_{23} = thyself	
T_{14} = How		

IBM Model 3-5

The generation is as per the following formula

$$p(\tau,\pi | f) =$$

$$\prod_{j=1}^{n} p(\phi_j | \phi_{1,j-1}, f)\ p(\phi_0 | \phi_{1,n}, f) \quad *$$

$$\prod_{j=0}^{n}\prod_{k=1}^{\phi_j} p(\tau_{jk} | \tau_{j1,.k-1}, \tau_{0,j-1}, \phi_{0,n}, f) \ *$$

$$\prod_{j=1}^{n}\prod_{k=1}^{\phi_j} p(\pi_{jk} | \pi_{j1,.k-1}, \pi_{1,i-1}, \tau_{0,n}, \phi_{0,n}, f) \ *$$

$$\prod_{k=1}^{\phi_0} p(\pi_{0k} | \pi_{0,k-1}, \pi_{1,l}, \tau_{0,l}, \phi_{0,l}, f)$$

IBM Model 3-5

After choosing the Tableau the words are Permuted to generate **e.**

This permutation is a random variable π

The position in **e** of the k^{th} word of the j^{th} Tablet is called π_{jk}.

In these models the generative process is expressed as a joint likelihood for a tableau τ and a permutation π, in The following way:

IBM Model 3-5

First Step: Compute

$$\prod_{j=1}^{n} p(\phi_j | \phi_{1,j-1}, \mathbf{f})\ p(\phi_0 | \phi_{1,n}, \mathbf{f})$$

- Determine ϕ_j the number of tokens that f_j will produce.
- This will depend upon the no. of words produced by $f_1 \ldots f_{j-1}$.
- Determine ϕ_0 the number of words generated out of NULL.

IBM Model 3-5

Second Step: Compute

$$\prod_{j=0}^{n}\prod_{k=1}^{\phi_j} p(\tau_{jk} \mid \tau_{j1,.k-1}, \tau_{0,j-1}, \phi_{0,n}, \mathbf{f})$$

- Determine τ_{jk} the k^{th} word produced by f_j
- This depends on all the words produced by $f_1 \ldots f_{j-1}$. and all the words produced so far by f_j

IBM Model 3-5

Third Step: Compute

$$\prod_{j=1}^{n}\prod_{k=1}^{\phi_j} p(\pi_{jk} \mid \pi_{j1,k-1}, \pi_{1,i-1}, \tau_{0,n}, \phi_{0,n}, \mathbf{f})$$

- Determine π_{jk} the position in **e** of the k^{th} word produced by f_j.
- This depends on the positions of all the words produced so far.

IBM Model 3-5

Fourth Step: Compute

$$\prod_{k=1}^{\phi_0} p(\pi_{0k} \mid \pi_{0,k-1}, \pi_{1,n}, \tau_{0,n}, \phi_{0,n}, \mathbf{f})$$

- Determine π_{0k} the position in **e** of the k^{th} word produced by *NULL*.

- This depends on the positions of all the words produced so far.

Thus the final expression is a product of 4 expressions:

IBM Model 3-5

$$p(\tau,\pi|\mathbf{f}) =$$

$$\prod_{j=1}^{n} p(\phi_j|\phi_{1,j-1},\mathbf{f})\ p(\phi_0|\phi_{1,n},\mathbf{f}) \quad *$$

$$\prod_{j=0}^{n}\prod_{k=1}^{\phi_j} p(\tau_{jk}|\tau_{j1,.k-1},\tau_{0,j-1},\phi_{0,n},\mathbf{f}) *$$

$$\prod_{j=1}^{n}\prod_{k=1}^{\phi_j} p(\pi_{jk}|\pi_{j1,.k-1},\pi_{1,i-1},\tau_{0,n},\phi_{0,n},\mathbf{f}) *$$

$$\prod_{k=1}^{\phi_0} p(\pi_{0k}|\pi_{0,k-1},\pi_{1,l},\tau_{0,l},\phi_{0,l},\mathbf{f})$$

IBM Model 3-5

This obviously is very difficult to manipulate.

Hence concessions are made for different models.

The concessions come in the form of assumptions.

Let us first look at the IBM Model 3.

IBM Model 3

IBM Model 3

Assumptions

1. For j between 1 and n $p(\phi_j \mid \phi_{1,j-1}, \mathbf{f})$ depends only on ϕ_j and f_j.
2. For j between 1 and n $p(\tau_{jk} \mid \tau_{j1,k-1}, \tau_{0,j-1}, \phi_{0,n}, \mathbf{f})$ depends only on τ_{jk} and f_j.

3. For j between 1 and n $p(\pi_{jk} \mid \pi_{j1,k-1}, \pi_{1,i-1}, \tau_{0,n}, \phi_{0,n}, \mathbf{f})$ depends only on π_{jk}, j, n, m.

This reduces the number of variables

IBM Model 3

Thus parameters for Model 3 are:

1. A set of Fertility probabilities: $\eta(\phi | f_j)$
   which is equal to $p(\phi | \phi_{1,j-1}, \mathbf{f})$

2. A set of Transition probabilities $t(e | f_j)$
   which is equal to $p(\tau_{jk} = e | \tau_{j\,1,.k-1}, \tau_{0,\,j-1}, \phi_{0,n}, \mathbf{f})$

3. A set of Distortion probabilities $d(i \mid j, m, n)$
   which is equal to $p(\pi_{jk} = i | \pi_{j\,1,.k-1}, \pi_{1,i-1}, \tau_{0,n}, \phi_{0,n}, \mathbf{f})$

IBM Model 3

The distortion and fertility probabilities for f_o (NULL) are treated in a different way:

These are meant for handling the words in TL sentence Which cannot be accounted for.

Obviously they are plugged-in once all the ϕ_j words Are place, for j = 1, .. n.

So $\varphi_1 + \varphi_2 + \quad + \varphi_n = m - \varphi_0$

We have to estimate these probabilities.

IBM Model 3

It is assumed that each of the Tableau word can produce at most one NULL word.

Assume each Tableau word produces a NULL word with Prob. p_1 and does not produce one with Prob. p_o

Hence $p(\varphi_o) = \binom{\varphi_1 + \varphi_2 + \quad + \varphi_n}{\varphi_0} p_1^{\varphi_0} p_0^{\varphi_1 + \varphi_2 + \quad + \varphi_n - \varphi_0}$

$$= \binom{m - 2\varphi_0}{\varphi_0} p_1^{\varphi_0} p_0^{m - 2\varphi_0}$$

IBM Model 3

As with Models 1 and 2, an alignment of (**e** | **f**) is Determined by specifying $a(i)$ for each position of the TL string.

The fertilities φ_j $j = 0, .., n$, are functions of the $a(j)$ s.: φ_j is equal to the number of i's such that $a(i) = j$.

Hence P (**e** | **f**) can be obtained as summing over All the alignments:

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

IBM Model 3

$$= \sum_{a(1)=0}^{n} \cdots \sum_{a(m)=0}^{n} \prod_{j=1}^{m} \binom{m-2\varphi_0}{\varphi_0} p_1^{\varphi_0} p_0^{m-2\varphi_0} \prod_{j=1}^{n} \phi_j! \, \eta(\phi_j | f_j) *$$

$$\prod_{i=1}^{m} t(e_i | f_{a(i)}) d(i | a(i), m, n)$$

With the following constraints

1. $\sum_{e} t(e | f) = 1$

2. $\sum_{i} d(i | j, m, n) = 1$

3. $\sum_{\phi} \eta(\phi | f) = 1$

4. $p_o + p_1 = 1$

IBM Model 3

Remarks:

1. Here also we have exponential number of alignments.
2. Count collection is too high even for moderate length sentence.
3. Sampling is used from the space of possible alignments
4. Sampling should be such that most probable ones are included.

IBM Model 3

Remarks:

5. Still it is much harder for Model 3.
6. Hence Hill-climbing type heuristics are used.
7. Typically they start from Model 1 solution.
8. From there go to neighboring alignments- where distance between two alignments is measured on the no. of points they differ.

IBM Model 4

IBM Model 4

Model 3 has been found to be a very powerful one.
It takes care of all the major aspects:

- word translation
- reordering
- insertion of words
- dropping of words
- one to many translation

But it has one major shortcoming:

formulation of distortion probabilities

$$d(i \mid j, m, n)$$

IBM Model 4

Model 3 does not take into account the following fact:

often a group of words are translated together, and therefore when they move they move together.

E.g

Riding a bicycle >> in sella a una bicicletta

Coming here riding a bicycle is dangerous >> venire qui in sella a una bicicletta è pericoloso

Try many sentences with the phrase "riding a bicycle" one can notice that the phrase "in sella a una bicicletta" will remain together.

But Model 3 considers the *distortion probabilities* in isolation.

IBM Model 4

Model 4 introduces the concept of Relative Distortion

It assumes that the placement of the translation of an Input word is based on the placement of the preceding input word.

It is however difficult to conceptualize: as words are being added, dropped, converted from one-to-many.

Model 4 is based around the concept of cept.

IBM Model 4

Definition: each input word that is aligned to at least one output word is called a cept. Typically represented by [ ] or π.

Definition: the ceiling of the average of the positions is called the center of a cept . We shall denote as C_j.

For each output word the Relative Distortion is defined With the help of cepts.

Let us first see an example:

IBM Model 4

Consider the following Bengali-English pair:

1 2 3 4 5 6

φ lambaa moton chhele cycle-e chore aaschhe

A tall boy is coming riding a bicycle.

1 2 3 4 5 6 7 8

Note: The does not align with anything!

moton – an ornametation is not a cept.

IBM Model 4

cept	π_1	π_2	π_3	π_4	π_5
Foreign Word Position	1	3	4	5	6
Foreign Word	lambaa	chheleta	cycle-e	chore	aaschhe
English Word	Tall	boy	a , bicycle	riding	is, coming
English word position	2	3	7,8	6	4, 5
Center of cept	2	3	8	6	5

IBM Model 4

Relative Distortion:

-Words generated by φ are Uniformly distributed.

- The position of the first word of a cept is defined w.r.t. the centre of the previous cept.

$$d_1(j - C_{j-1})$$

Consider for example : the word "riding"

it is generated by cept 4 (л4)

its English position is: 6

Centere of the preceding cept is 8.

Thus there is a distorion of -2.

This shows a forward movement of the word.

Normally the distortion will be +1

IBM Model 4

Relative Distortion:

- For subsequent words of a cept the position is defined w.r.t. the position of the previous word of the same cept.

$$d_{>1}(j - \pi_{j,k-1})$$

Where $\pi_{j,k-1}$ refers to the k^{th} word of the j^{th} cept.

For example, in "a bicycle" "is coming" the distortion Probability of the second word is calculated in relation with the previous word.

IBM Model 5

IBM Model 5

The key term for Model 5 is Deficiency.

Models 3 & 4 do not take care of whether two words Are being put in the same place.

Thus it puts positive probabilities on some impossible Translations.

In Model 5 the distortion probabilities are calculated By considering cepts (as before) plus vacancies.

Also it takes care of the problem of multiple tableaus.

This makes it a better word-based model.

IBM Model 5

Model 5 keeps track of the vacancies in the m-word long **e** sentence.

Let

- v_{max} be the maximum no. of vacancies possible.
- v_j be the no. of vacancies available in the sentence **e** in the positions [1, j]

Hence the distortion probabilities are functions of 3 quantities: $d_1(v_j, C_{j-1}, v_{max})$

Similarly the relative distortion of the subsequent words in the cept are: $d_{>1}(v_j - v_{\Pi_{j,k-1}}, v_{max})$

Conclusion

Still we go by word based translation.

Can we do better? Because looking at translations
As word-by-word is not the best thing.

E.G The train is in.
The train is in motion.
The train is in station.
The train is danger.

Proper translation demands that we need to see the
Word along with the context.

This gives us the concept of "Phrase-based Translation"

Thank You