# Statistical Machine Translation

*NILADRI CHATTERJEE*

*DEPARTMENT OF MATHEMATICS*

*IIT DELHI*

December 21, 2013

## Workshop on

## Machine Learning and Text Analytics

# Translation

Historically Translation between languages is taking place for centuries, if not millennia.

But it was by human

Computers is only a few decades old

But translation with computers has been restricted to formal languages

Natural languages are far more complex!!!

# What is Machine Translation

- Machine Translation (MT) pertains to automated translations of text or speech from one natural language to another.
- MT aims at providing a tool for breaking the language barrier.
- In a multilingual environment (like EU, India) there may be two types of translation system:
  - Between two languages local to the environment
  - Between a foreign language and a local language
- The focus is on text translation.

# Difficulties in Dealing with Natural Languages

- Expression is not unique. The same sense may be conveyed in many different ways.
- Construction of sentences is governed by a set of rules or grammar. But often there are exceptions.
- How this information is organized in our brains is not known. Consequently knowledge representation in NLP systems is a significant area of research.
- This is true for different NLP applications. In this talk we shall focus on Machine Translation.

We start with some Examples:

# Example with Existing Translator

- The girl is beautiful
- The girl is pretty
- The girl looks beautiful
- She is good-looking
- I find the girl beautiful
- To me the girl is beautiful

All these convey the same meaning. But can a MT system understand it?

Google Translator has the following outputs:

# Example with Existing Translator

The girl is beautiful -> लड़की खूबसूरत है (Ladkii khubsurat hai)

মেয়ে সুন্দর (Meye Sundor)

આ છોકરી સુંદર છે (Ā chōkarī sundara chē)

The girl is pretty -> लड़की सुंदर है (Ladkii sundar hai)

মেয়ে সুন্দর হয় (Meye Sundor hoy)

આ છોકરી ખૂબ સુંદર છે (Ā chōkarī khūba sundara chē)

The girl looks beautiful -> लड़की सुंदर लग रहा है

(Ladkii sundor lag rahaa hai)

মেয়ে সুন্দর দেখায় (Meye sundor dekhaay)

આ છોકરી સુંદર દેખાય છે (Ā chōkarī sundara dēkhāya chē)

# Example with Existing Translator

The girl is good-looking -> लड़की अच्छी लग रही है

*Ladkii achhi lag rahii hai*

মেয়ে সুদর্শন হয় (*Meye sudarshan hoy*)

આ છોકરી દેખાવડું છે (*Ā chōkarī dēkhāvaḍu chē*)

I find the girl beautiful -> मैं लड़की सुंदर मिल

*Main ladkii sundar mil*

আমি মেয়ে সুন্দর এটি (aami meye sundor eTi)

હું છોકરી સુંદર શોધવા (Huṁ chōkarī sundara śōdhavā)

To me the girl is beautiful -> मेरे लिए लड़की सुंदर है

আমার মেয়ে সুন্দর (Aamaar meye sundor)

મને છોકરી સુંદર છે (Manē chōkarī sundara chē

*Mane chokari sundara lagechhe* **is correct one**

# Some More Interesting Translations

| | | |
|---|---|---|
| The Fan is on | प्रशंसक पर है | *Prashansak par hai* |
| | **পাখা হয়** | *Paakhaa hoy* |
| It is raining | यह बारिश हो रही है | *Yah barish ho rahii hai* |
| | **বৃষ্টি হচ্ছে** | *Brishti hochchhe* |
| They had a big fight | वे एक बड़ी लड़ाई है | *Way ek bade ladaai hai* |
| | **তারা একটি বড় যুদ্ধ আছে** | *Taaraa ekTi baro juddho achhe* |
| Shopkeeper ran out of Vegetables | दुकानदार सब्जियों से बाहर भाग गया | *Dukaandaar sabzion se bahar bhaag gayaa* |
| | **দোকানদার সবজি পরিমাণ স্বল্প** | *Dokaandaar sabzi parimaan swalpo* |

# Some Other Translators

Angla Bharti - IIT Kanpur
Rule Based

Mantra RajBhasa - C-DAC
(Tree Adjoining Grammer)
Lexical Tree to Lexical Tree

MaTra2 - Hybrid
(Rule-based + Statistical)
Transfer based

# Example -1

The headquarters of planning committee are located in New Delhi.

- मुख्यालय का/की/के नियोजना समिति हैं स्थित में नई दिल्ली
(Angla Bharti)

- समिति योजना के मुख्यालय नई दिल्ली में स्थित हैं.
(Google)

- योजना बनाने वाले समिति का केंद्रस्थान नई दिल्ली स्थित हैं
(Matra Rajbhasa)

- प्लैनीग समिति का मुख्यालय न्यू दिल्ली में पता लगाया जाता है
|
(Matra2)

# Example -2

Where did you hide the can opener?

- आपने डिब्बा ओपनर को कहाँ छिपाया

(Angla Bharti)

- सलामी बल्लेबाज कर सकते कहां है

(Google)

- जहाँ किया हुआ आप प्रारंभ करने वाला छुपाते हैं

(Matra Rajbhasa)

- आप कैन खोलनेवाला छिपाते हो

(Matra2)

# Where lies the problem?

# Problems of Machine Translation

- Word level difficulties
- Syntactic ambiguity
- Referential ambiguity
- Semantic ambiguity
- Metaphors and symbols

# Word Level difficulties

- **Polysemy:** Same word may have different meaning.
  - *I am going to the bank.*
  - *This is of high interest.*
- **Synonymy:** Synonymous words may not be recognized.
  - *He has a car.*
  - *He has an automobile.*

# Word Level difficulties (2)

- **Hyponyms:** Class/subclass identification may be a difficulty.
  - *He has a car.*
  - *He has a sedan.*
  - *He has a SX4*
  - *He has a Maruti*

***Question:*** *How much semantics do we need?*

Check their Google Translation

# Word Level difficulties (2)

| | |
|---|---|
| He has a car. | वह एक कार है. |
| He has a sedan. | उन्होंने कहा कि एक पालकी है |
| He has a SX4 | उन्होंने कहा कि एक SX4 है |
| He has a Maruti | उन्होंने कहा कि एक मारुति है |

# Word Level difficulties (3)

**Homograph:** Same word may be used as different part of speech.

*Drinking more water is good for health.*

*Please water the saplings carefully.*

- **Idiomatic expressions:** Idioms often do not have any correspondence with the constituent words.
  - *My mother gave me a piece of cake*

    मेरी माँ ने मुझे केक का एक टुकड़ा दिया.
  - *The test was a piece of cake for me.*

    परीक्षण मेरे लिए केक का एक टुकड़ा था.

# Syntactic Ambiguity

Structure of sentence does not clearly convey the sense.

- *Flying planes can be dangerous.*
- *I saw the man with a telescope.*
- *This my favourite chef 's dish.*
- *My father loves me more than my brother.*

# Semantic Ambiguity

Sentences may have the same syntactic structure, but their meaning changes with constituent words.

- He took the rice *with* a curry
- He took the rice *with* a spoon
- He took the rice *with* difficulty
- He took the rice *with* a bad smell
- He took the rice *with* a sad face
- He took the rice *with* a friend.

# Semantic Ambiguity

Similarly one may consider the following set:

He came by car.

He came by three o'clock.

He came by London.

He came by himself.

He came by night.

He came by a forest

# Complex Semantic Ambiguity

- **Homonymy:** to understand the sentence specific sense has to be used.
  - *The box is in the pen* → **बॉक्स कलम में है**
- **Metonymy:** substituting the name of an attribute or feature for the name of the thing itself
  - *They counted heads*
  - *While driving John hit the tree. He called a mechanic.*

# Langauge Specific Features

- Metaphors
- Idioms
- Proverbs
- Symbols

Are often difficult to translate.

# Selection of Right Word

# What is the Right Word?

Target language may have many words
Corresponding to one source-language word:

Uncle -> चाचा मामा ताऊ मौसा
Bird -> पक्षी पंछी चिड़िया
Pumpkin -> कुम्हड़ा, कद्दू, काशीफल, कोहँड़ा, कोहड़ा, कूष्मांड, कूष्मांड, सीताफल, मीठा-कद्दू, पिंडफल, पिण्डफल, पुष्पफल, वृहत्फल, वेष्टक, आमक

Neela (Hindi) -> Blue, Indigo, Azure etc.
Ice -> 32 varieties in Eskimo language

# Pattern Ambiguity

This is another difficulty observed with respect to English to Hindi MT [Chatterjee et. al. 2005]

This happens when the same verb is used in different senses.

E.g *Run* has 41 different senses. *Have* has 19 different senses.

They need to be translated differently:

| English Sentence | Hindi Verb |
|---|---|
| The river ran into the sea. | *milnaa* |
| The army runs from one end to another. | *failnaa* |
| They run an N.G.O | *chalaanaa* |
| He runs for treasurer. | *khadaa honaa* |
| Wax runs in sun. | *galnaa* |
| We ran the ad three times | *prakaashit* |

# Noun Compounds

A compound noun is a noun that is made up of two or more words.
The joined words may form a single word also.
Examples:

*tooth + paste = toothpaste.* (N + N)
*black + board = blackboard.* (Adj + N)

*pressure cooker*
*marriage ceremony*
} Separated Words

Eg: *Database* but *Knowledge base*

# Noun Compounds Translation

| | | |
|---|---|---|
| rule base | → | बेस शासन |
| rulebase | → | rulebase |
| blackboard | → | ब्लैकबोर्ड |
| black board | → | श्यामपट्ट |
| database | → | डेटाबेस |
| data base | → | डेटा बेस |
| knowledge base | → | ज्ञान का आधार |
| knowledgebase | → | नॉलेजबेस |

# Noun Compound Translation

The primary difficulty for an MT system is to understand the semantics.

If semantics can be known and conveyed properly then translation becomes easier:

Often this boils down to grouping the words:

Sometimes it is easy from the adjective:
E.g.

Deep Blue Shirt -> ((Deep Blue) Shirt)
Round Copper Lid -> (Round (Copper Lid))

# Noun Compound Translation

Sometimes very similar structure has different semantics:

Statistical Machine Translation->
(Statistical (Machine Translation))

Statistical Inference Problem ->
((Statistical Inference) Problem)

OR

Public Service Commission ->
((Public Service ) Commission)

Public Underground Transport ->
(Public (Underground Transport))

# What did we Observe?

# Observations from the Examples

Word-by-word translation does not help.

Translation does not depend upon structures.

Eg. It is raining vs. It is running

Translation of words change with context.

Opener -> खोलनेवाला

Can opener -> सलामी बल्लेबाज कर सकते हैं

Lot of knowledge is required

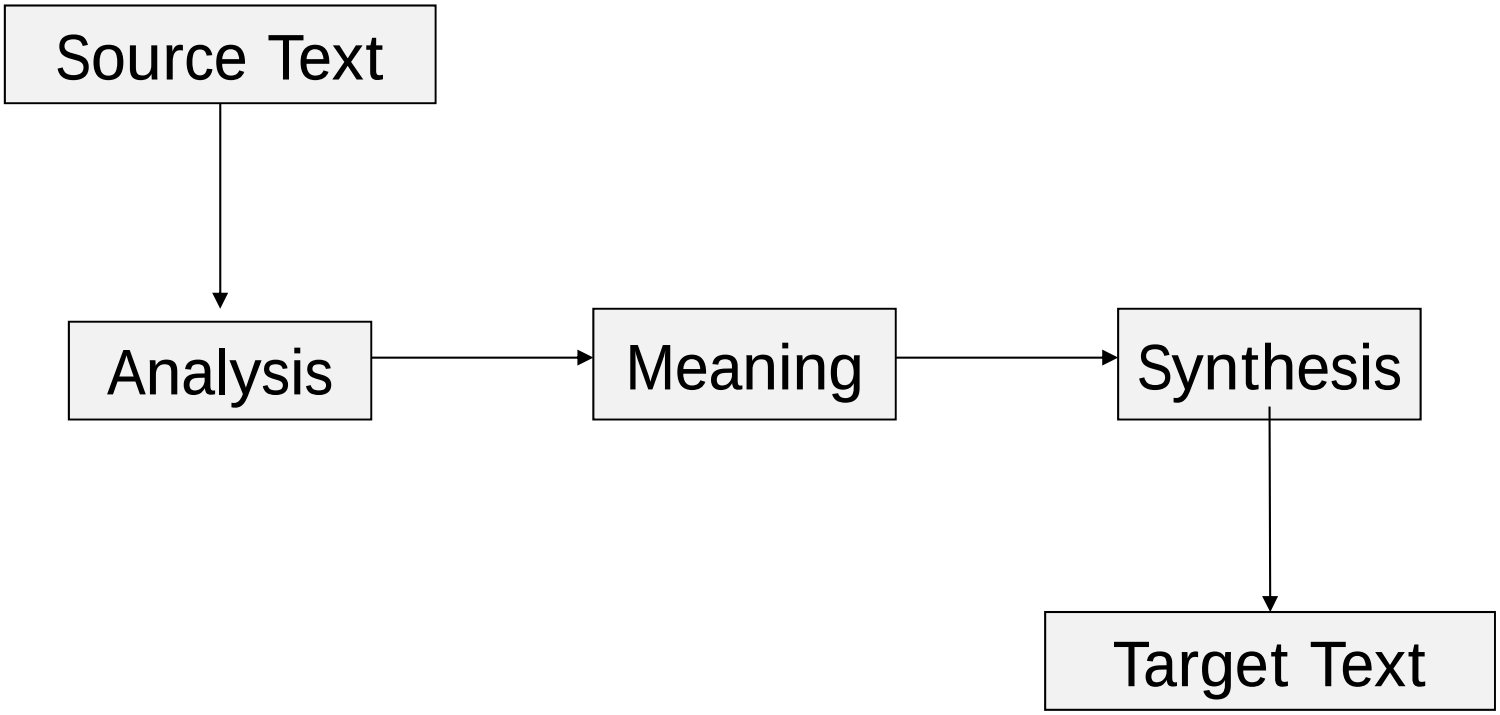
E.g. I have an Irish setter / Newfoundland / Lamborghini

# Knowledge Used in Translation

One would expect use of different types of knowledge:

- Knowledge of the source language
- Knowledge of the target language
- Knowledge of correspondences between the source and target languages
- Knowledge of the subject matter and general knowledge used to understand what the text means
- Knowledge of the culture, social conventions, customs, expectations of speakers of the source and target languages

# Translation Process




However we shall see that even without such explicit knowledge huge success can be achieved

How to capture this knowledge?

# Different Paradigms

The Major Paradigms (historically)

Lexical-Based (LBMT) -- Dorr, Tsujii & Fujita
Rule-Based (RBMT) – Kaplan, Okumura
Knowledge-Based (KBMT) – Carbonell, Nirenburg
Example- Based (EBMT) – Nagao, Somers
Neural-Net Based (NBMT) -- McLean

# Different Paradigms

Lexicon-based MT—Based on relating the lexicon entries of one language to the lexicon entries of the other language e.g. Anusaarka (IIT-K, IIIT Hyderabad) late1990s.

Issue: How to capture polysemous word
Rearrangement of terms

Knowledge-based MT– concentrates on development of knowledge intensive morphological, syntactic and semantic information for the lexicon e.g. Pangloss [CMU,1980], GAZELLE [USC, 1990].

Issue: Which knowledge and how much
How to capture all the domain knowledge

# Different Paradigms

EBMT Proposed as an MT technique by Nagao in 1984. Based on the idea of performing translation by imitating examples of translations of sentences of similar structure.

Issues: How to measure similarity.
How to handle divergence

Rule-based MT– relies on different linguistic levels of rules for translation between two languages.

Issues: Complicated unconventional sentence structures

Eg. *This is the dog that worried the cat that killed the rat that ate the malt that lay in the house that Jack built*

# Issues between Hindi & Bengali

Reduplication.

I don't like taking fish daily ->

ami **roj roj** machh khete bhalobasi naa.
mujhe **har roz** machhli khaana pasand nahin hai.

He comes here secretly ->

se **chupi chupi** ekhane ase.
woh **chhup chhup ke** idhar aataa hai

# Insight into the Languages

Consider translation of this sentence:

The boy who comes here everyday riding a blue bicycle sings well.

यहाँ आता है लड़का है जो हर रोज सवारी नीले साइकिल अच्छा गाती है.

এখানে যারা আসে ছেলে দৈনন্দিন অশ্বচালনা একটি নীল সাইকেল ভাল sings.

# Insight into the Languages

The girl **in white dress** sings well

*safed kapde**walli** ladkii achhi gaatii hai*

*shada kapor **pora** mayeta bhalo gaay.*

Consequently Knowledge-based approaches do not fit the bill perfectly.

Here comes *Statistical Machine Translation*

# Prologue

- Gained tremendous momentum in recent years

- Generally languages are so rich, complex, different that it is difficult to distil knowledge to frame exhaustive set of rules or examples, which can be encoded into program

- Can then the rules be discovered automatically?

  (perhaps from a pair of corpus, and analyzing the data statistically)

This begins a new line of research and gives rise to

**Statistical Machine Translation**

# Statistical Machine Translation

It is Based on n-gram modeling, and probability distribution of the occurrence of a source-target language pair in a very large corpus. e.g.

IBM model, Matador (Univ. of Maryland)

- Started in the ‘90s,
- Became more popular after 2000
- Modeling Translation Task as optimization

# Observation

- The rules cannot be very *deterministic*.
- An *element of probability* is associated with translation.
- Perhaps one can estimate the probabilities from a huge set of *parallel corpus*.
- One needs to consider the *associations between different pairs of words*.
- Hence arises the question of modeling – *statistical modeling*.

# *Statistical Modelling*

# Statistical MT

A true translation which is both Faithful and Fluent is often impossible.

A translation is said to be faithful if it conveys the full sense of the source sentence.

E.g. *wo ladkaa kal sham ko yahaan aayaa thaa* >>
The boy came here yesterday (NOT Faithful)

A translation is said to be fluent if its construction correctly follows the grammar of the target language.

E.g. *wo ladkaa kal sham ko yahaan aayaa thaa* >>
The boy came yesterday evening here
(NOT Fluent)

# Statistical MT

A compromise is often tried for.

We want a model that maximizes a value Function.

SMT is about building a probabilistic model to combine faithfulness and fluency:

$$\text{Best translation } \hat{T} = \underset{T,\,S}{\text{argmax}} \; \text{faithful}(T,S) * \text{fluency}(T)$$

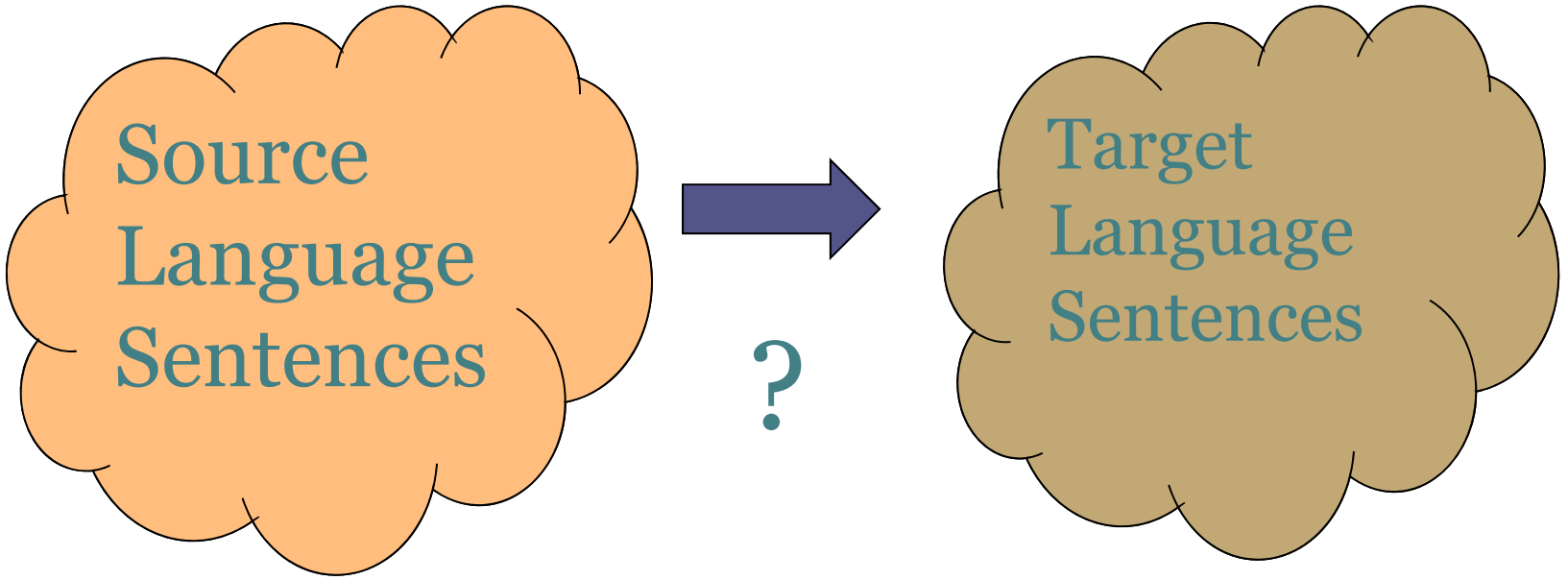
Consider that a source language sentence S may translate into any target language sentence T.

Some translations are just more likely than others.

How do we formalize "more likely"?

# Statistical MT

Of course the problem is how to estimate the probabilities:




Both are of infinite size.
Hence individual probabilities are zero.
Hence estimation is done in an indirect way.

But we look at the modeling aspect first.

# Statistical MT

P(**s**) -- a priori probability. The chance that **s** happens.

For example, If **s** = "May I know your name"

Then P(**s**) is the chance that a certain person at a certain time will say "May I know your name" as opposed to saying something else.

P(**t** | **s**) -- conditional probability. The chance of **t** given **s**.

For example, Let **s** = May I know your name

and

**t** = *kyaa main aapkaa naam puchh saktaa hoon*

then P(**t** | **s**) is the chance that upon seeing **s**, a translator will produce **t**.

# Statistical MT

Given a Source Sentence **s**, we seek Target Sentence **t** that maximizes p(**t** |**s**). ("most likely" translation)

This we write: $\underset{t}{\text{argmax}}\ p(t|s)$

Using Bayes thm: $p(s, t) = p(s) * p(t \mid s)$

Therefore:

$$p(t \mid s) = p(s, t) / p(s) = p(s \mid t) * p(t) / p(s)$$

Hence $\underset{t}{\text{argmax}}\ P(t \mid s) = \underset{t}{\text{argmax}}\ p(s \mid t) * P(t)$

Thus Bayes'rule helps us to find the right **s** based on simpler probabilities.

This is known as Noisy-Channel Model, where three modelings are involved:

- Source model to compute P(**t**)
- Channel model to compute P(**s** | **t**)
- Decoder to produce **t** given **s**

# *Modeling Basics*

However, obviously things did not start with such difficult approaches.

Things started with modeling using Words.

We shall start with WORD modeling

But before that we look at

* Word Alignment

* Language Modeling

# Word Alignment

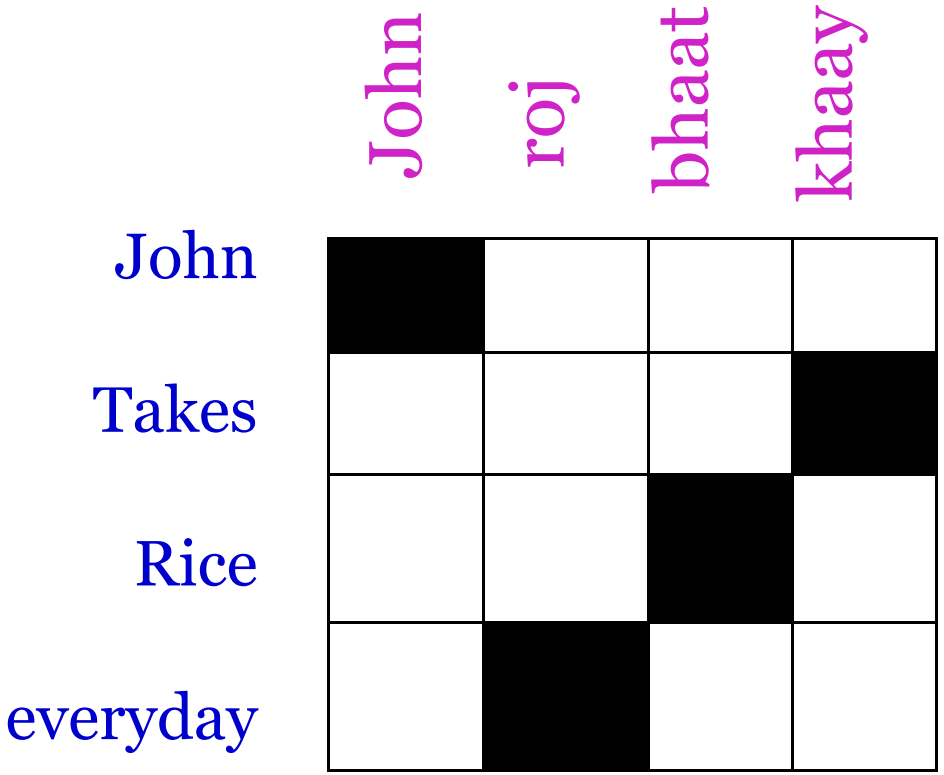
Word alignment is the NLP task of identifying translation relationships among the words (or more rarely multiword units) in a bi-lingual Text (bitext)

It can be shown in many ways:

-Matrix : where the $(i, j)^{th}$ cell is darkened if $e_i$ corresponds to $f_j$ s

- bipartite graph: between the two sides of the bitext, with an arc between two words if and only if they are translations of one another.

# Word Alignment




However, it is not straightforward always.

# Word Alignment

Consider

John does not take rice everyday>>

*John roj roj bhaat khaay naa* (B)

(Sometimes to repeat a word for emphasizing)

Consider the English word “does”. Which word it will be aligned to?

- Should we leave it unaligned?
- Should we align with “khaay”?
- Should we align with “naa”?

# Word Alignment

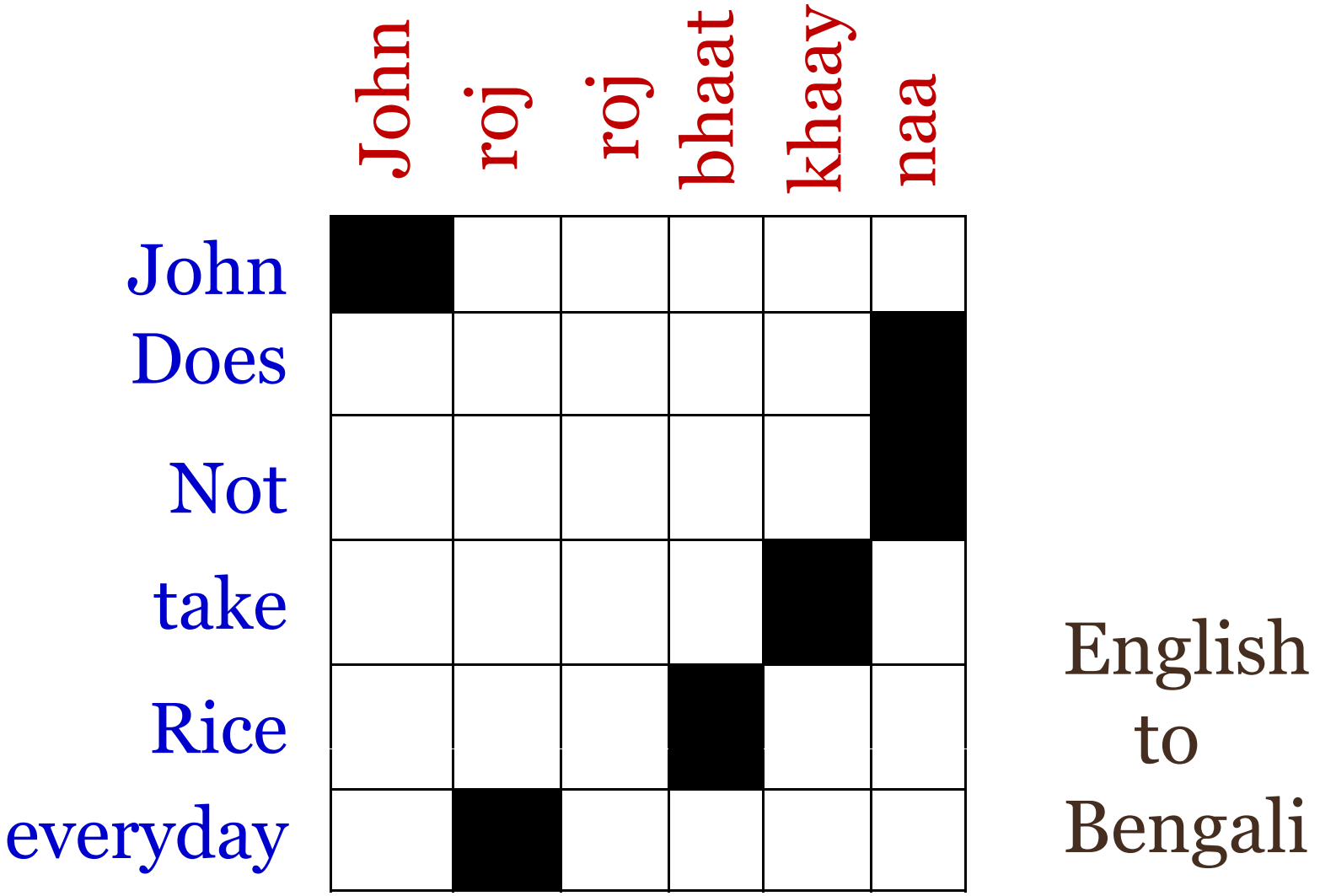
One solution is to look it both ways:

$e$ >> $b$ and $b$ >> $e$

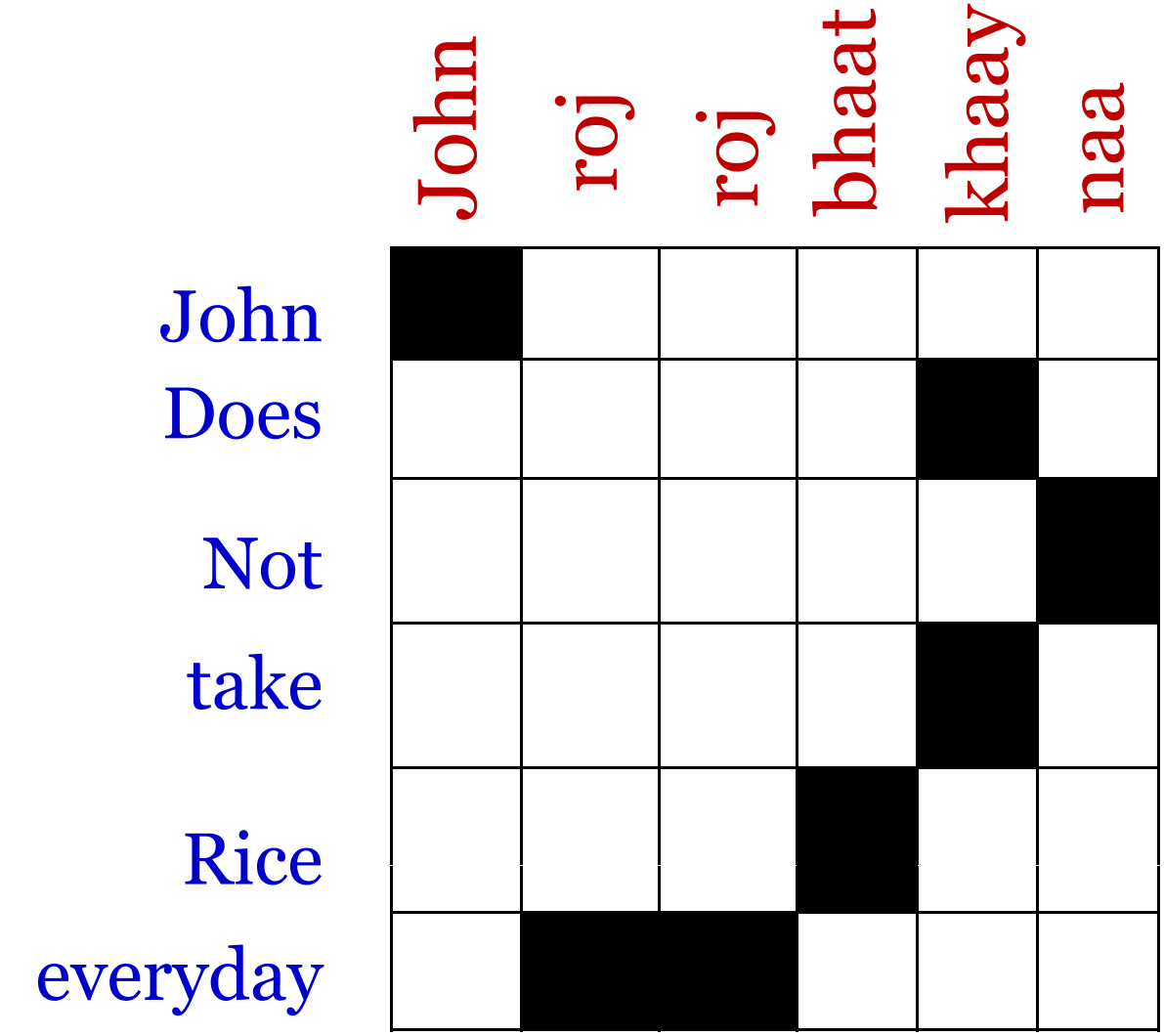
Then consider Union and Intersection.

- Paint the cells in the Intersection as Dark.
- Paint remaining cells in Union as Grey.

# Word Alignment

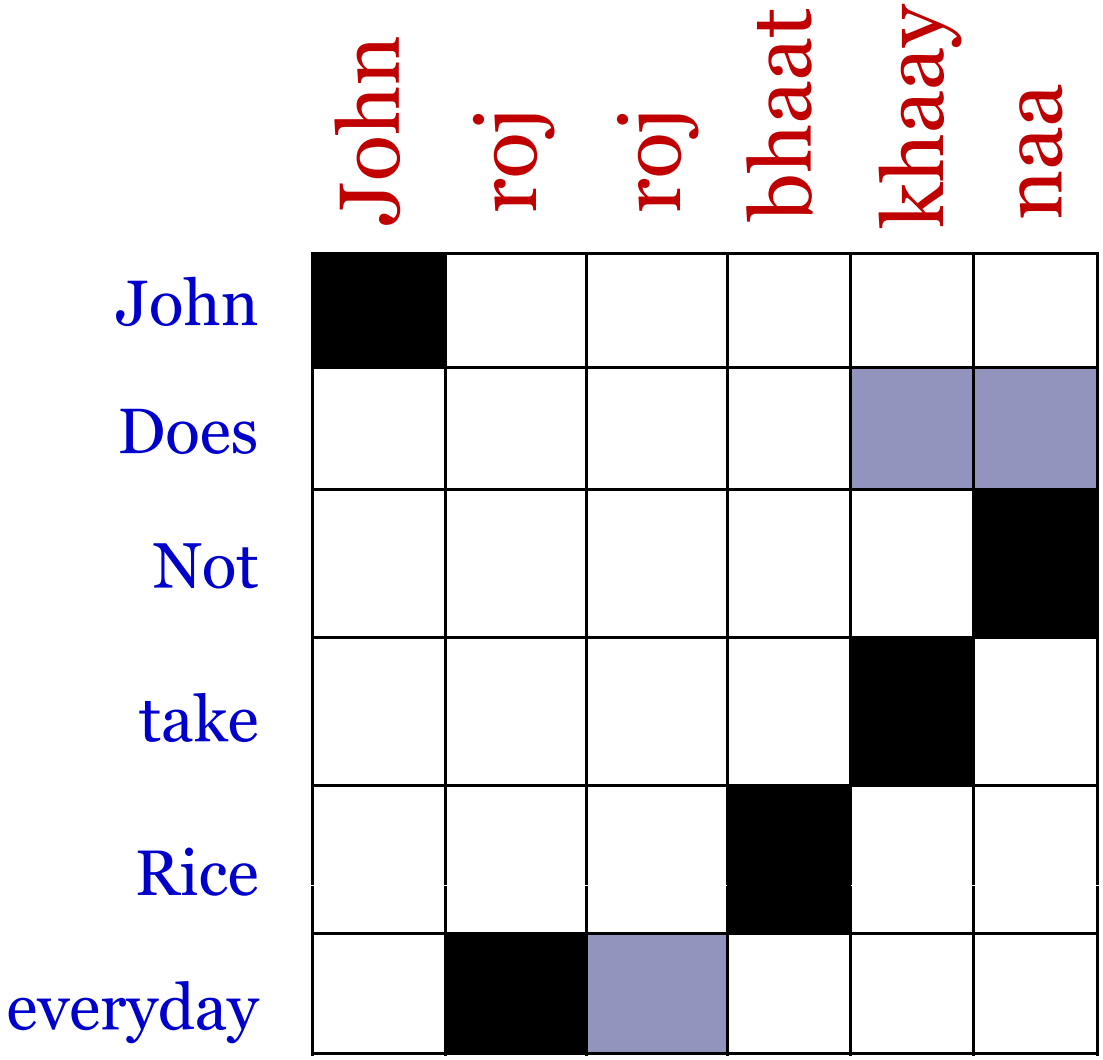


# Word Alignment




Bengali to English

# Word Alignment




Final Word Alignment

# Word Alignment

Many algorithms have been developed for Word Alignment.

One can study :

Somers – Recency Vector Based

Fung and McKeown - - do -

Chatterjee & Agrawal – Constraint based

We look at it in the form of Alignment function

Shall come back to it while talking on IBM Models.

# Language Modeling

# Introduction

Let us relook at the equation again:

$$\underset{t}{\operatorname{argmax}}\ P(t \mid s) = \underset{t}{\operatorname{argmax}}\ P(s \mid t) * P(t)$$

If we analyse the two terms of the RHS:

- the first term talks about faithfulness
- the second term talks about fluency

*Just word translation does not give a good translation.

*We need to take care of features of the TL also.

*Language modeling is developed with this objective.

*Tells how likely it is that a sequence of words will be written in the language.

# Language Model

We can think of modeling the *target language*

Helps in : fluency, word order etc., which vary across languages

E.g. <Noun> <Adj>$_{it}$ >> <Adj> <Noun>$_{eng}$

Formally, LM is a function that gives the probability of a given sentence. E.g. $P_{EM}$(He is a good boy) > $P_{EM}$(He is a boy good)

The obvious difficulty is:

- there are infinitely many sentences in
- any language.

So how to obtain??

# Language Model

We try to compute probabilities from language corpus.

Computing probabilities of sentences are meaningless
Hence n-gram modeling is used.

For different values of n (e.g. 1, 2, 3, ..) the probability of the sequence of n words $w_1\ w_2\ ..\ w_n$.

Using Bayes' Theorem:

$$P(w_1\ w_2\ ..\ w_n) = P(w_1)\ P(w_2 \mid w_1)\ P(w_3 \mid w_1\ w_2)\ .....\ P(w_n \mid w_1\ w_2\ ..\ w_{n-1})$$

The size on n depends on language, corpus etc.

# Estimation

The question is how to estimate the probabilities

1. Unigram $(w)$ = $$\frac{count\ (w)}{Total\ \ no.of\ Words}$$

2. Bigram $(w_1, w_2)$ = $$\frac{count\ (w_1, w_2)}{\sum_w count\ (w_1, w)}$$

3. Trigram $(w_1, w_2, w_3)$ = $$\frac{count(w_1, w_2, w_3)}{\sum_w count(w_1, w_2, w)}$$

The bigger the corpus, the better the estimate!

# Example

## Consider 3 sentences:

<S> I am John </S>
<S> John I am </S>
<S> I like river Don and friend John </S>

P(I | <S>) = 2/3
P(am | I) = 2/3
P(</S> | John) = 2/3

P (John | <S>) = 1/3
P (John | am) = 1/2
P(Don | river) = 1.0

Relative frequency: sequence: prefix is calculated
In reality millions of words are used to estimate these Probabilities.

# N-grams

N-grams allow us to ascertain associations:
e.g.

salt and pepper (noise)
centre forward (soccer)
deep fine leg (cricket)
yellow journalism
purple patch
European Parliament

A good model should have higher probabilities For these phrases.

# N-grams

N-gram probabilities helps in translation:

e.g. *aami bhaat khai* >> I eat rice
*aami jal khai* >> I drink water
*aami churut khai* >> I smoke cigar
*aami osudh khai* >> I take medicine

N-grams allow us to choose the right translation

# N-grams

Very similar things can happen with other Language pairs also:

The boy >> Le garçon FR
>> Il ragazzo IT

The boys >> Les garçons FR
>> I ragazzi IT

The girl >> La fille FR
>> La ragazza IT

The girls >> Les filles FR
>> Le ragazze IT

# Probability Calculations

# Calculation

Consider the example from Jurafsky and Martin
Based on 9332 sentences and 1446 words.
The table shows bigram counts

| | I | Want | To | Eat | Chinese | food |
|---|---|---|---|---|---|---|
| I | 5 | 827 | 0 | 9 | 0 | 0 |
| Want | 2 | 0 | 608 | 1 | 6 | 6 |
| To | 2 | 0 | 4 | 686 | 2 | 0 |
| Eat | 0 | 0 | 2 | 0 | 16 | 2 |
| Chinese | 1 | 0 | 0 | 0 | 0 | 82 |
| Food | 15 | 0 | 15 | 0 | 1 | 4 |

# Calculation

Consider the example from Jurafsky and Martin
Based on 9332 sentences and 1446 words

| | I | Want | To | Eat | Chinese | food |
|---|---|---|---|---|---|---|
| I | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 |
| Want | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 |
| To | 0.0008 | 0 | 0.0017 | 0.28 | 0.00083 | 0 |
| Eat | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 |
| Chinese | 0.0063 | 0 | 0 | 0 | 0 | 0.52 |
| Food | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 |

**Suppose**
**Further given that**
**P(I | <S>) = 0.25**
**P(</S> | food) = 0.68**

P(<S> I want to eat food </S>) can be calculated as:
0.25 * 0.33 *0.66 * 0.0027 * 0.68 = 0.000099

# Data Used

Typically 3 data sets are used:

- Training data – used for training a model for estimating statistical parameters.
- Held out data – an augmented training set, used for fine-tuning the model e.g. for smoothing.
- Test data - the parameter values thus obtained are used for estimating the probabilities

# *Smoothing*

# Why Smoothing?

No training set can cover all possible English Word sequence.

What happens in test set we get an n-gram *not* seen in the training set?

The conditional probabilities will give 0.

- not useful from practical point of view.

Also, how do we know the number of times an n-gram is expected to be in the test set.

What is the implication that an n-gram occurs c times in the training set.

# Smoothing techniques

- Empirical approach.
- Mathematical Approach

-Interpolation & Back off

In this talk we restrict to some simple functions

# Count Smoothing / La Place Smoothing

- Simplest form of smoothing.
- For unigram: $p(w_i) = \frac{c_i}{N}$ , N is the total no. of words
- For smoothing 1 is added to each count.
- After Laplace smoothing: $p_{LP}(w_i) = \frac{c_i + 1}{N + V}$
- Calculate for example: N = 100000, V = 10000, c = 500
- Small weights are given to unseen words. But it affects the probabilities of seen words hugely. Hence
- Modified La Place (add $\alpha$ smoothing): $p_{LP\alpha}(w_i) = \frac{c_i + \alpha}{N + \alpha V}$ where $\alpha < 1$, is experimentally determined.

# Count Smoothing / La Place Smoothing

For bigram: Let the count for a bigram $(w\ v) = c$

MLE for the its probability $= \frac{c}{N}$, N is the total no. of bigrams

If the size of vocabulary is V - possible bigrams is $V^2$

- After Laplace smoothing: $p_{LP}(w,v) = \frac{c+1}{N+V^2}$

- Modified La Place $p_{LP\alpha}(w,v) = \frac{c+\alpha}{N+\alpha V^2}$

Look how drastically the probabilities change!!

# Count Smoothing / La Place Smoothing

## Results from Europarl corpus: Philip Koehn

| Count c | Add 1 smoothing $(c+1) * n/ (n + V^2)$ | Add $\alpha$ smoothing $(c+\alpha ) * n/ (n + \alpha V^2)$ | Test Count |
|---|---|---|---|
| 0 | 0.00378* | 0.00016 | 0.00016 |
| 1 | 0.00755 | 0.95725 | 0.46235 |
| 2 | 0.01133 | 1.91433 | 1.39946 |
| 5 | 0.02266 | 4.78558 | 4.35234 |
| 10 | 0.04155 | 9.57100 | 9.11927 |
| 20 | 0.07931 | 19.14183 | 18.95948 |

$\alpha = 0.00017$

$V = 86700$

*Too much Weight for Unseen N-grams

Note: $n > 29x10^6$ size of corpus

# Count Smoothing / La Place Smoothing

## Results from Europarl corpus: Philip Koehn After Fine Tuning using held out data

| Bi-gram Counts | Count Add α smoothing | Test Count | Actual count Training | Actual count held out | Expected Count |
|---|---|---|---|---|---|
| 0 | 0.00016 | 0.00016 | 7,515,623,434 | 938,504 | 0.00012 |
| 1 | 0.95725 | 0.46235 | 753,777 | 353,383 | 0.46900 |
| 2 | 1.91433 | 1.29946 | 170,913 | 239,736 | 1.40322 |
| 5 | 4.78558 | 4.35234 | 31,413 | 134,653 | 4.28820 |
| 10 | 9.57100 | 9.11927 | 9,106 | 85,666 | 9.41129 |
| 20 | 19.14183 | 18.95948 | 2,797 | 53,262 | 19.04992 |

Note: How closely the Add α count matches the expected count

# *IBM Word Models*

# Introduction

Word models come from the original work at IBM.

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

# Introduction

The MT technologies have advanced since then.

These have then extended to

Phrase Based models

More recently other models are coming up:

- Treelet Based models
- FactorBased Models

In this talk we shall visit up to PBMT in more details

# 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:

$f1\ f2\ \ldots fn$

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:

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

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

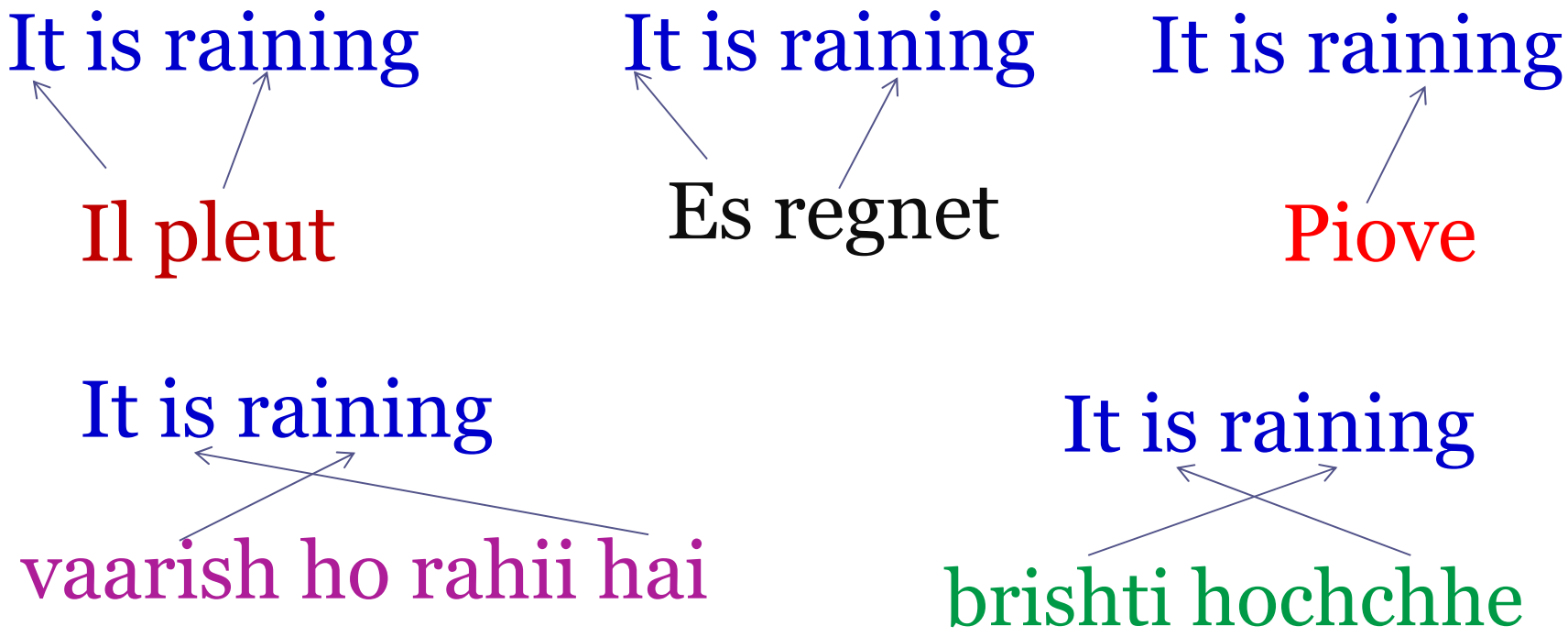
2 – 1 : das haus ist **klitzeklein** →
the house is **very small**

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

etc.

# Word Alignment

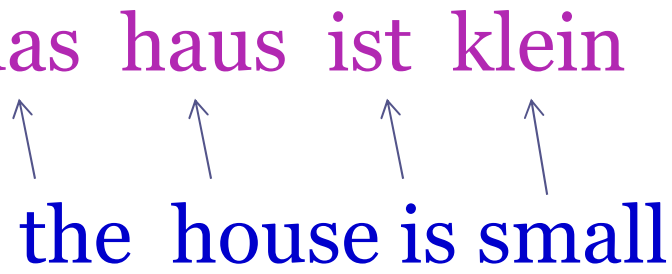
And it varies with language pairs:



# 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




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

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 → 1, 2 → 2, 3 → 3, 4 → 5}

E.G 3.

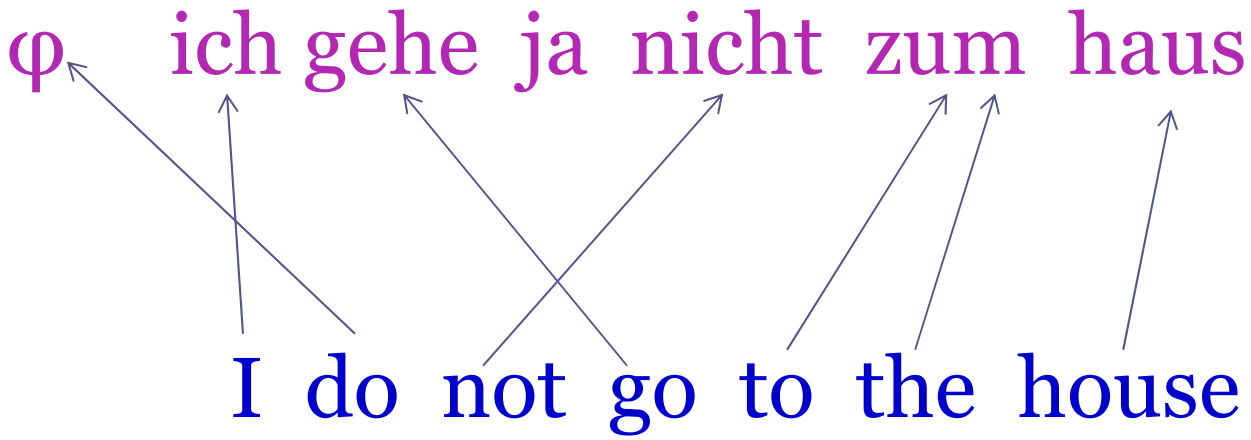
das haus ist klitzeklein

the house is very small

$a$: {1 → 1, 2 → 2, 3 → 3, 4 → 4, 5 → 4}

# Word Alignment function

E.G 4




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

Alignment function will be needed in the models

# *IBM Models*

# IBM Models

- IBM Model 1 : Based on Lexical Model
- IBM Model 2 : Adds Alignment Model
- IBM Model 3 : Adds Fertility Model - How many output words an input word produces.
- IBM Model 4 : Adds *Relative Alignment* Model Positions of any other *e* words that are connected to the same *f* word
- 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. Model 5 keeps track of vacant positions, and allows new words to be inserted only in these positions.

# *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

$a: j \longrightarrow i$ ($e_j$ 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.*

| | |
|---|---|
| *mm aa bb* | → I like meat |
| *aa rr pp kk* | → I take fruits daily |
| *pp tt bb* | → You like fruits |
| *tt mm rr* | → You cook meat |

Can we learn the alignment??

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| θ) 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| θ)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

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

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

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

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

# Mathematical Jugglery

Consider:

$$a_1b_1 + a_1b_2 + \ldots + a_1b_m +$$
$$+ a_2b_1 + a_2b_2 + \ldots + a_2b_m +$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots +$$
$$a_nb_1 + a_nb_2 + \ldots + a_nb_m$$

# Mathematical Jugglery

Consider:

$$a_1b_1 + a_1b_2 + \ldots + a_1b_m +$$
$$+ a_2b_1 + a_2b_2 + \ldots + a_2b_m +$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots +$$
$$a_nb_1 + a_nb_2 + \ldots + a_nb_m$$

This can be simplified into:

$$(a_1 + a_2 + \ldots\ldots + a_n)\ (b_1 + b_2 + \ldots\ldots + b_m)$$

# EM-Algorithm for IBM 1

We now estimate the probability of a given alignment $a$, given **e** and **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 $f_j$

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

Lagrange Multiplier technique is used 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 $\lambda_j$

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 $\lambda_j$

# EM-Algorithm for IBM 1

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

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

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

Hence

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

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:

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

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

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

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

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

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

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

Where $\lambda$ 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 $\lambda_j$ using $\sum_{e_w} \sum_{(\mathbf{e}^{(s)}, \mathbf{f}^{(s)})} count(e_w \mid f_j ; \mathbf{e}^{(s)}, \mathbf{f}^{(s)})$

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

S4: for each $e_i$ that appears in at least 1 $\mathbf{e}^{(s)}$)
compute new $t(e_i \mid f_j)$ using:

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

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

# *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

# 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 ??

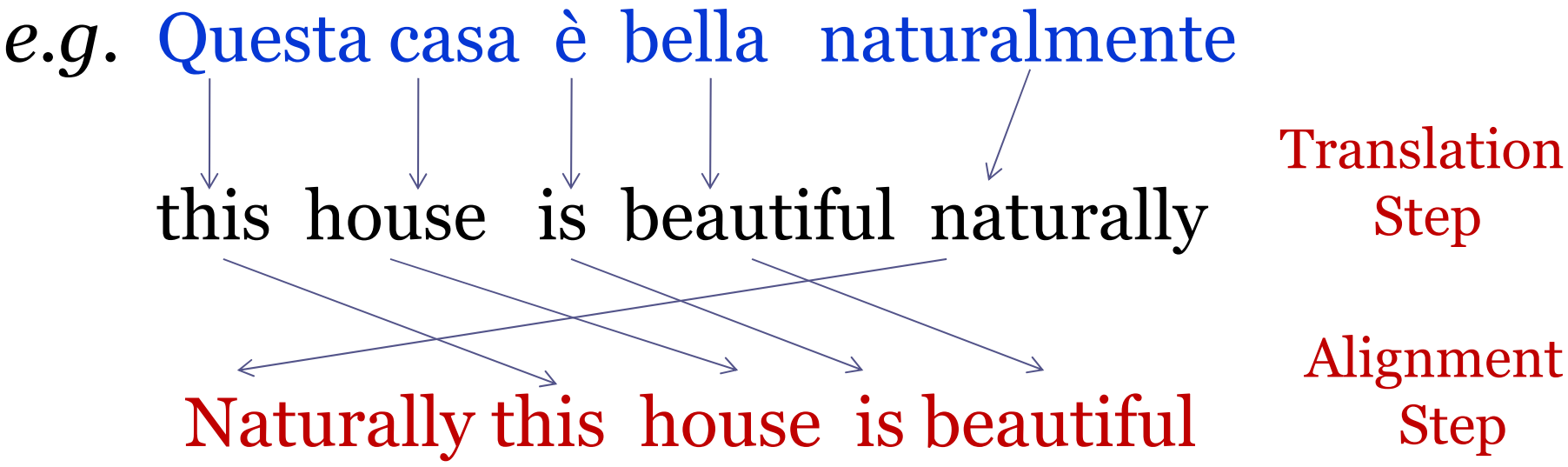
# IBM Model 2

Thus translation is a two-step process:

Lexical Translation step:

modeled by $t(e_p \mid f_q)$

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



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

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

Thus we have a larger set of Lagrangian constants

# 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**

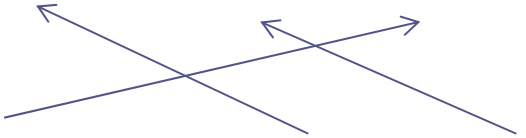
**1>1 2>0 3>3 4>2**

tumi ki khachchho

**What are you eating**

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

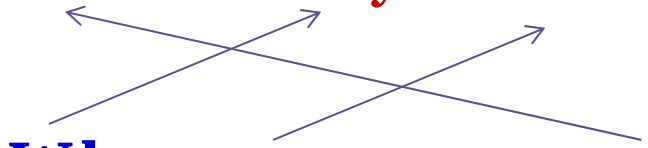
tomar naam ki



**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

The auxiliary function becomes:

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

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

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

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 **e** is connected to the word in the position $j$ of the SL string **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

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

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

# IBM Model 3-5

# 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 (It) >>
*despite, even though, in spite of* (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

# Higher Models of IBM

## Model 3: Adds Fertility Model

*Fertility:* How many output words an input word produces.

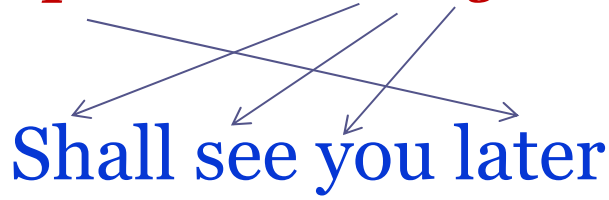
- It is not necessary that each input word produces only one word.
- Some may produce more than 1.
- Some may produce no word!!

# Higher Models of IBM

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


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

E.g. 2. *phir milenge*




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

# IBM Model 3-5

Example: *Tumi kaal abashyoi jabe*
May have many possible English translations:

- You must go next day
- You have to go tomorrow
- You ought to be there on the following day
- You will have to go there tomorrow
- 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

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

This permutation is a random variable $\Pi$

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

In these models the generative process is expressed as a joint likelihood for a tableau $\tau$ and a permutation $\pi$, 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 $\phi_j$ 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 $\phi_0$ 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 $\tau_{jk}$ 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**

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

- Determine $\pi_{jk}$ 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**

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

- Determine $\pi_{0k}$ 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

**The generation is as per the following formula**

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

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

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

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

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

# 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.

# PHRASE-BASED MODELS

# Introduction

1. Often group of words taken together are translated in a different way:

   e.g. My daughter is *apple of my eyes*.

| | | | |
|---|---|---|---|
| My | मेरे | My daughter | मेरी बेटी |
| daughter | बेटी | is apple of | सेब की है |
| is | है | my eyes | मेरी आँखों |
| apple | सेब | My daughter is | मेरी बेटी है |
| of | की | apple of my eyes | मेरी आँखों का तारा |
| my | मेरे | My daughter is apple of my eyes | |
| eyes | आंखें | मेरी बेटी मेरी आँखों का तारा है | |

# Introduction

Solution is ***phrase based model***

Note: A *phrase* here is NOT the way a parser defines based on syntactic role: NP, VP. Rather it is a *multi-word unit* as a sequence of words.

When such set of consecutive words occur frequently and statistics about their translations are calculated, it may lead to better translation.

# Mathematical Definition

Here we improve upon the Bayesian Model:

$$\mathbf{e}_{Best} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} p(\mathbf{f}|\mathbf{e})p(\mathbf{e})$$

For phrase-based model, the term $p(\mathbf{f} \mid \mathbf{e})$ is *further broken down:*

Let **f** be split into *I* phrases: $\overline{f_1}, \overline{f_2}, \ldots, \overline{f_I}$

*Not modeled explicitly – so all segments are Equally likely*

# Mathematical Definition

Each of the foreign phrases $\overline{f_1}$ , $\overline{f_2}$ , … , $\overline{f_I}$ is translated into corresponding **e** phrase $\bar{e}_i$

Hence we get:

$$p(\mathbf{f} \mid \mathbf{e}) = \prod_{i=1}^{I} \phi(\overline{f_i} \mid \overline{e_i}) * pd(d_i)$$

- *pd is the relative distortion probability*
- $d_i$ is a relative distortion of the $i^{th}$ phrase.

This is because the phrases need not be in the same order in **f** as in **e**.

# Mathematical Definition

Example:


There should be a way to handle this reordering

Often *pd* is taken as a decaying function : $\alpha^{|x|}$, s..t. $\alpha \in (0, 1)$.

Although it is possible to study their distribution from a corpus.

However, it is not easy.

# Mathematical Definition

A simple measure for $d_i$ is $start_i - end_{i-1} - 1$

- Counts no. of **f** words skipped when taken out of sequence.
- Computed on the basis of **e** phrase and **f** words.

Note: wihtout any reordering, $start_i = end_{i-1} + 1$

Ex: natuerlich hat john spass am spiel (DE)

of course john has fun with the game

d1 = 0 d2 = 1 d3 = -2 d4 = 1 d5 = 0

# Phrase Translation table

Gives information about how the phrases are translated.

The phrase translations are extracted from word alignment

Can be accomplished in many ways.

We shall use an algorithm based on *consistency*.

The following algorithm is designed on the top of word alignment in a parallel corpus.

It extracts phrase pairs that are *consistent* with the word alignment

**Q: Can we develop a good algorithm for Phrase alignment?**

# What is Consistency?

Definition: A phrase pair ( $\bar{f}, \bar{e}$ ) is said to be consistent w.r.t an alignment A, if all words $f_1 .. f_n$ in $\bar{f}$ that have alignment points in A have these with words $e_1 .. e_n$ in $\bar{e}$ and vice versa:

( $\bar{f}, \bar{e}$ ) is consistent with A $\Leftrightarrow$

$$\forall e_i \in \bar{e} \mid (e_i, f_j) \in A \Rightarrow f_j \in \bar{f}$$

$$\wedge \quad \forall f_j \in \bar{f} \mid (e_i, f_j) \in A \Rightarrow e_i \in \bar{e}$$

$$\wedge \quad \exists e_i \in \bar{e}, f_j \in \bar{f} \mid (e_i, f_j) \in A$$

# Phrase Extraction

We need an algorithm to extract consistent phrases:

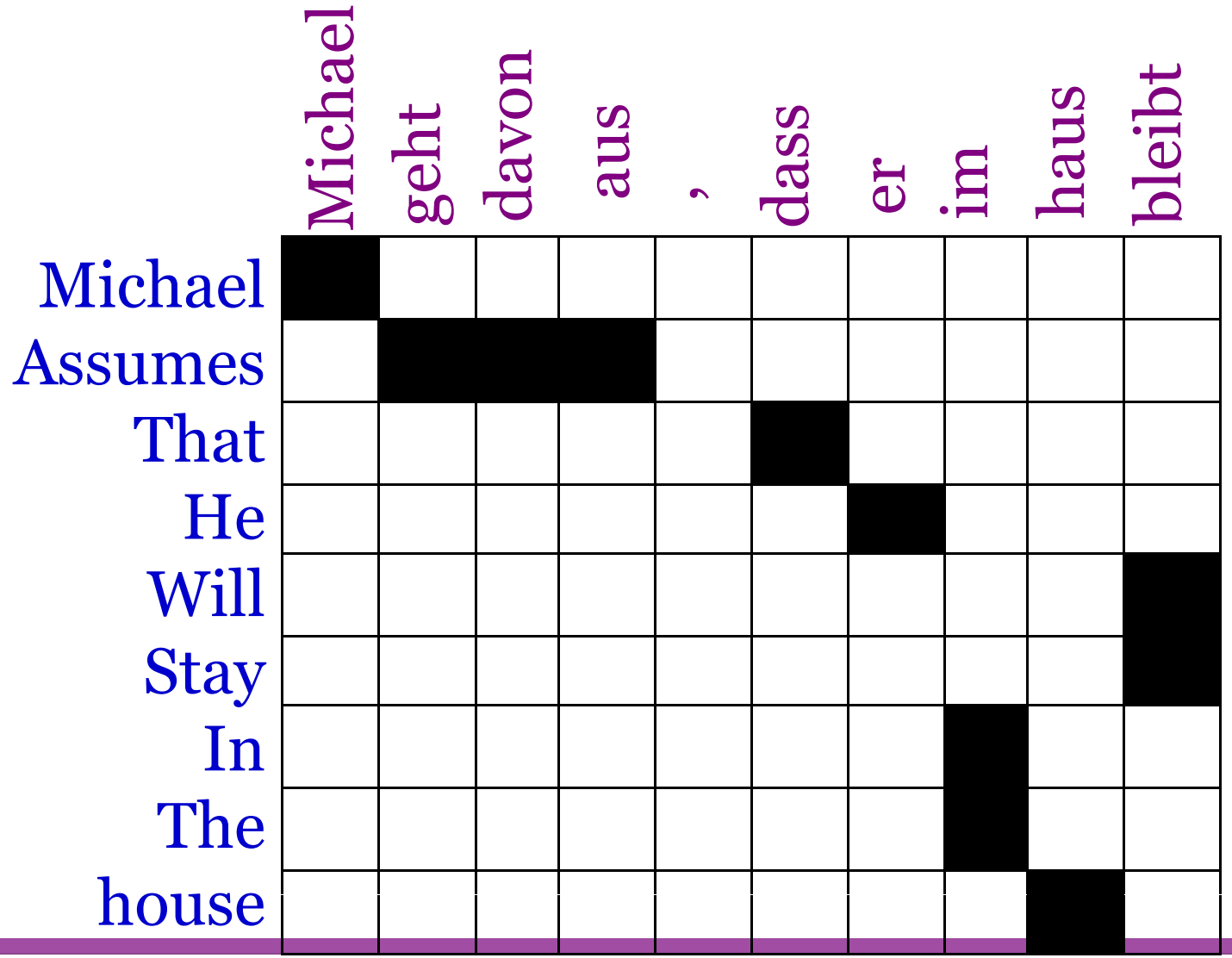
- It should go over all possible sub-sentences
- For each it extracts the minimal foreign phrase
- Matching is done by identifying all alignment points for $\bar{e}$, and then identifying the shortest $\overline{f}$ that includes all the alignment points.

NOTE:

- If $\bar{e}$ contains only nonaligned points then ∃ no $\overline{f}$
- If the matched $\overline{f}$ has additional alignment points, then *it cannot be extracted.*
- If the minimal phrase borders unaligned words, then the extended phrase also extracted.

# Phrase Extraction Example

From Philip Koehn.



# Phrase Extraction Example

The extracted phrases are: From Philip Koehn.

Michael – michael
Michael assumes – michael geht davon aus
michael geht davon aus ,
Michael assumes that –
michael geht davon aus , dass
Michael assumes that he –
michael geht davon aus , dass er
Michael assumes that he will stay in the house–
michael geht davon aus , dass er im haus bleibt
assumes – geht davon aus | geht davon aus ,
assumes that – geht davon aus , dass

# Phrase Extraction Example

assumes that he – geht davon aus , dass er

assumes that he will stay in the house–
geht davon aus , dass er im haus bleibt

that – dass | , dass

that he - dass er | , dass er

that he will stay in the house – dass er im haus bleibt |
, dass er im haus bleibt

he – er

he will stay in the house – er im haus bleibt

will stay – bleibt

will stay in the house - im haus bleibt

in the - im

in the house - im haus

house - haus

# Phrase Extraction Example

Some Statistics:

- 9 english Words  vs. 10 German words
- 11 alignment points
- 45 contiguous English phrases
- 55 contiguous German phrases
- 24 pairs have been extracted.

# *Translation Process*

# Translation Process

Let us now look at the most practical aspect:

How the translation is generated for a given new sentence **f** ?

For illustration consider the Italian sentence:

non gli piace il cibo indiano

We expect one person non-conversant with the language will proceed as follows

# Translation Process

Non gli piace il cibo indiano

No

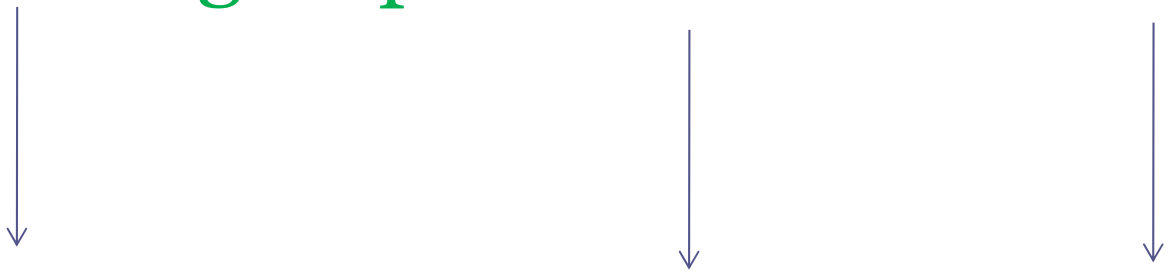
# Translation Process

Non gli piace il cibo indiano

No Indian

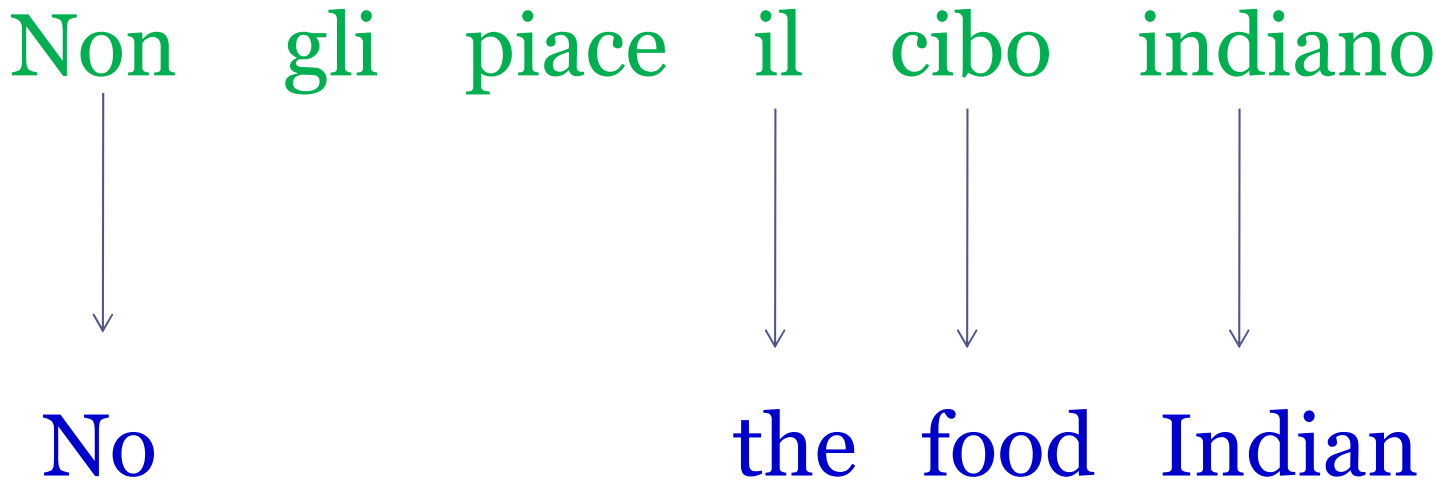
# Translation Process

Non gli piace il cibo indiano

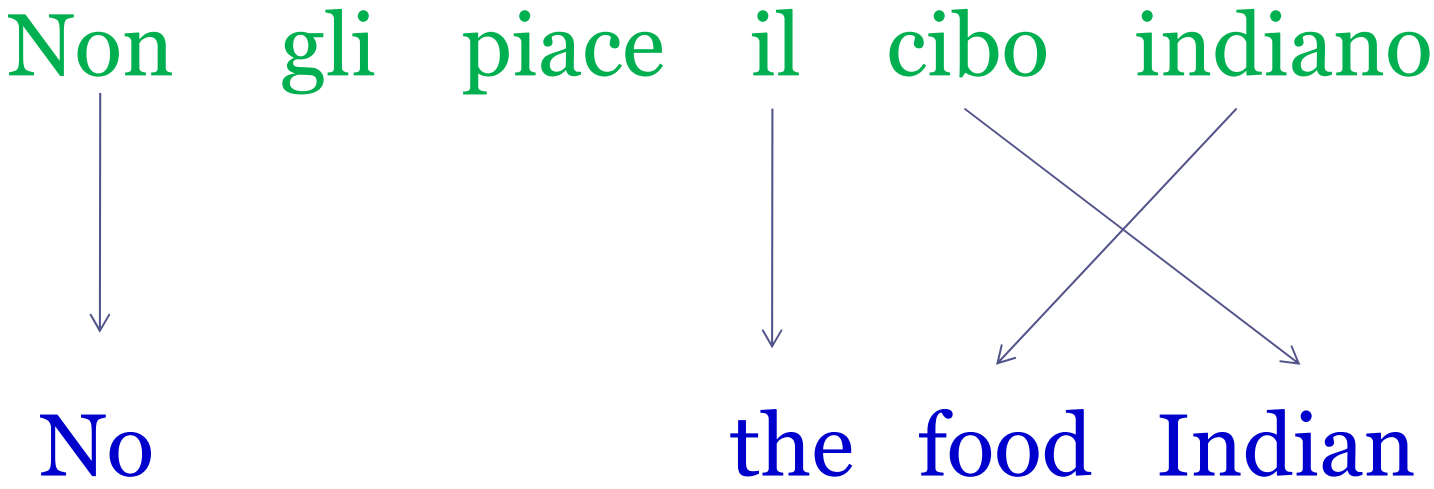


No the Indian

# Translation Process

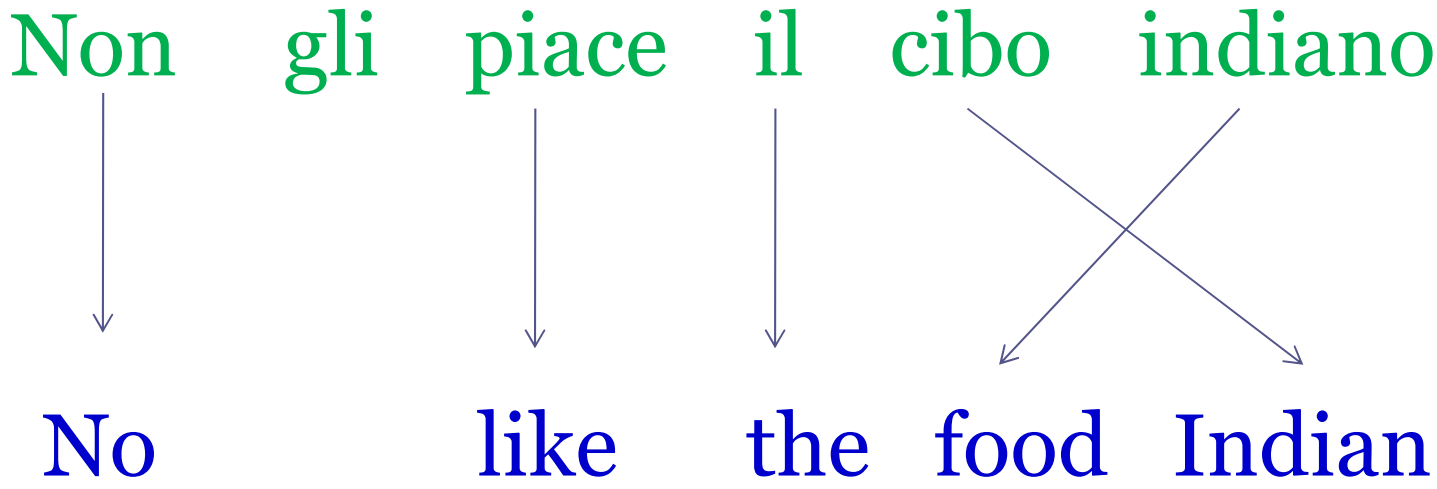


# Translation Process

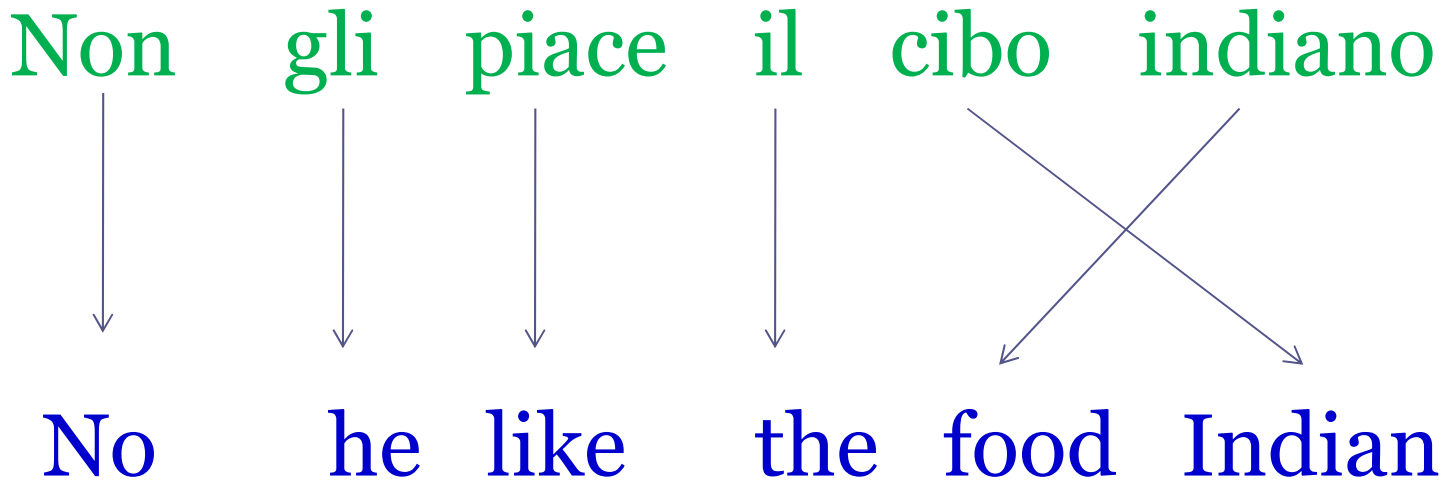



Apply Reordering

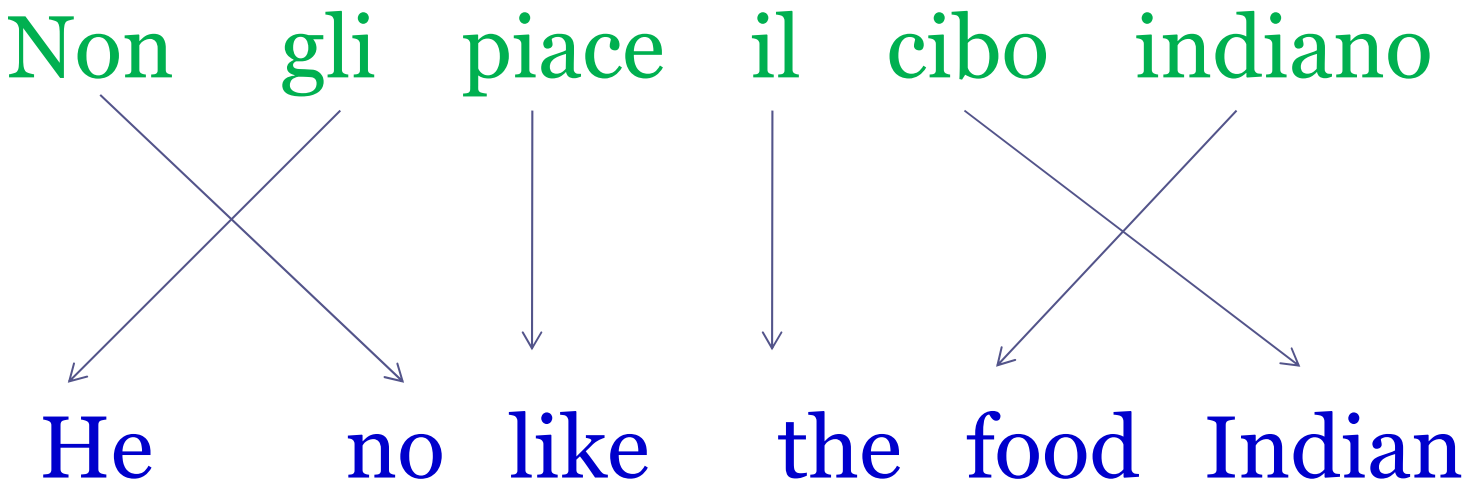
# Translation Process



# Translation Process

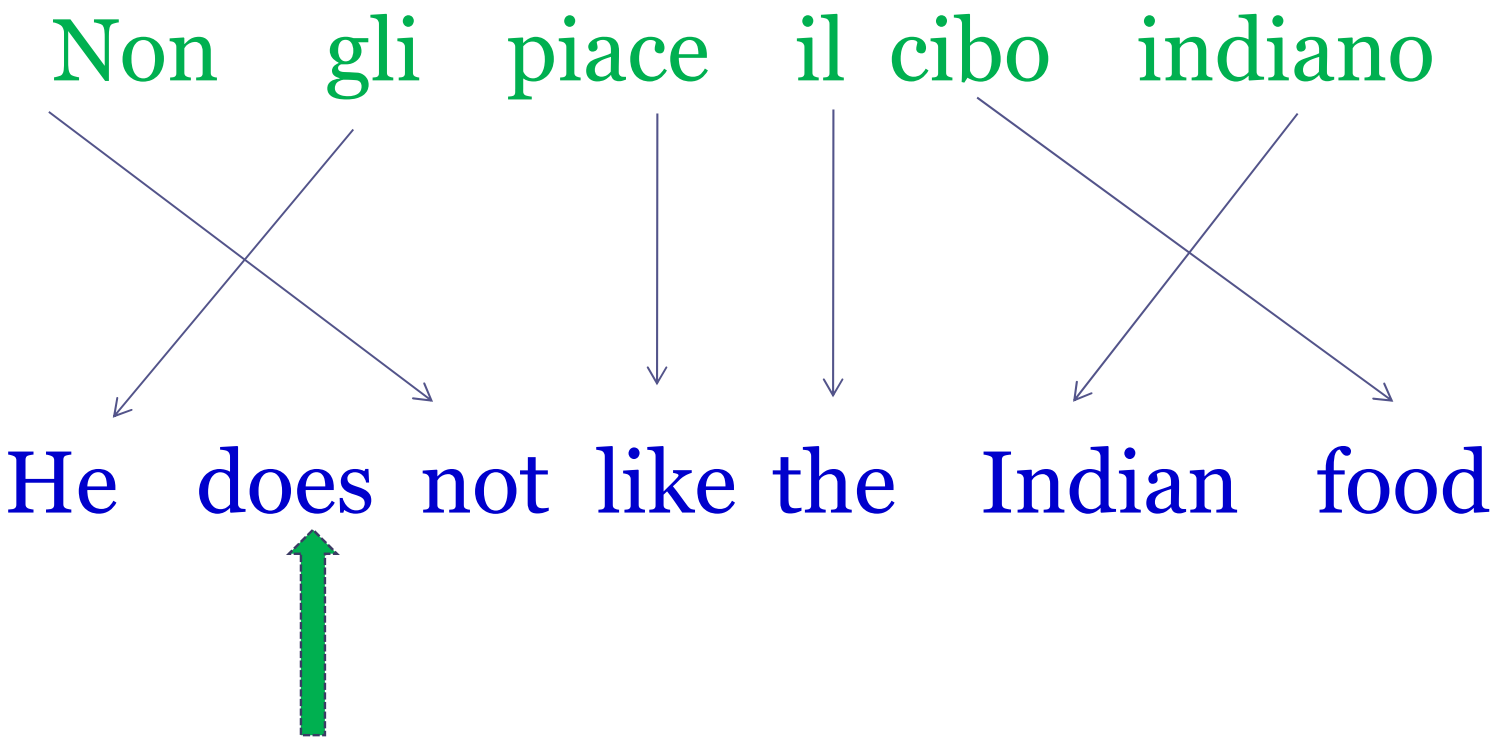


# Translation Process



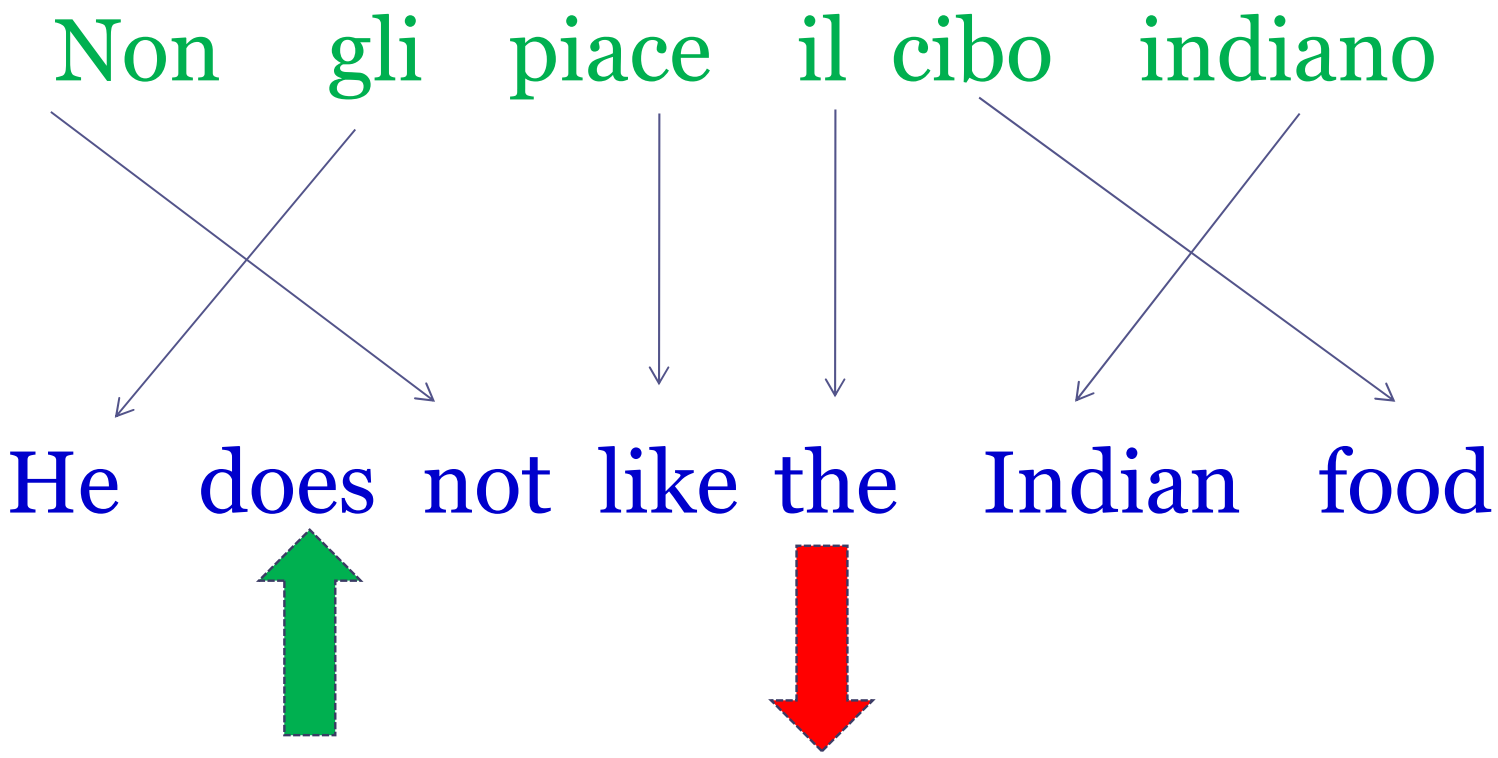

Apply Reordering

# Translation Process




Apply Language Modeling

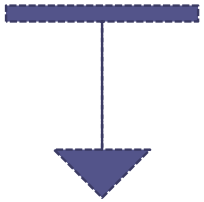
# Translation Process




Apply Language Modeling

# Translation Process

Non gli piace il cibo indiano



He does not like Indian food

Got the final Translation!!!

# Comments

This example is done at a word level.

But suppose we have Phrase Translation tables, which Gives the following translations directly:

Il cibo indian >> Indian food
Non gli piace >> he does not like

Then the whole translation process will be easier.

Hence Phrase-Based Methods are pursued.

# Difficulties

- In each case we may have different options

E.G non >> not, none
gli >> the, him, it
piace >> like
cibo >> food, grub, viands, meat, fare, scoff, cheer
indiano >> Indian, Hindu

So the task is NOT as easy as we thought!!

# Difficulties

If we look at from phrases we can get multiple translations:
E.g.

non gli piace >> not like (got it from google)
dislikes,
does not like,
is not fond of

Hence most our actions will be probabilistic.

The task therefore boils down to three Important tasks:

- Design Reordering Model
- Decoding.

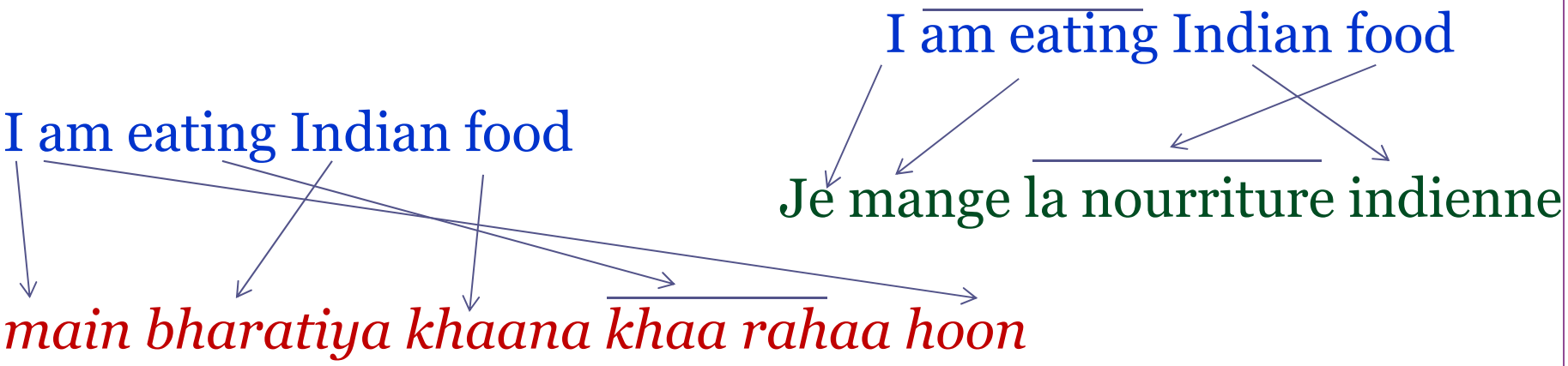
# Reordering Model

Reordering seems to be the most difficult of the tasks.
As most of the reordering are based on language pairs

It is difficult to design a general reordering model –

E.g. Adj NN (English) >> NN Adj (Fr)

Aux V Main V (English) >> Main V Aux V (Hindi)




Consequently total reordering requires much more information!!

# Reordering Model

The reordering model we discussed is based on the movement distance $d_i$.

Does not take care of the underlying word (or its class).

Hence there is a lot a scope to improve reordering:

- e.g *hierarchical*, *reordering with syntactic knowledge*

We discuss one important Reordering Model viz. Lexical Reordering.

# Reordering Model (Lexical)

Here we observe that some phrases switch positions more often than others:

Studente di dottorato (It) >> Doctoral student
Traduzione automatica >> Machine translation
Stati Uniti d'America >> United States of America
European Parliament >> Parlamento europeo

Parlement Européen (Fr) >> European Parliament
Bombe atomique >> Atom bomb

America yuktarashtra (B) >> United States of America

# Lexical Reordering

Reordering is done on the basis of the Actual phrases.

Here we take statistics of three possible directions of reordering:

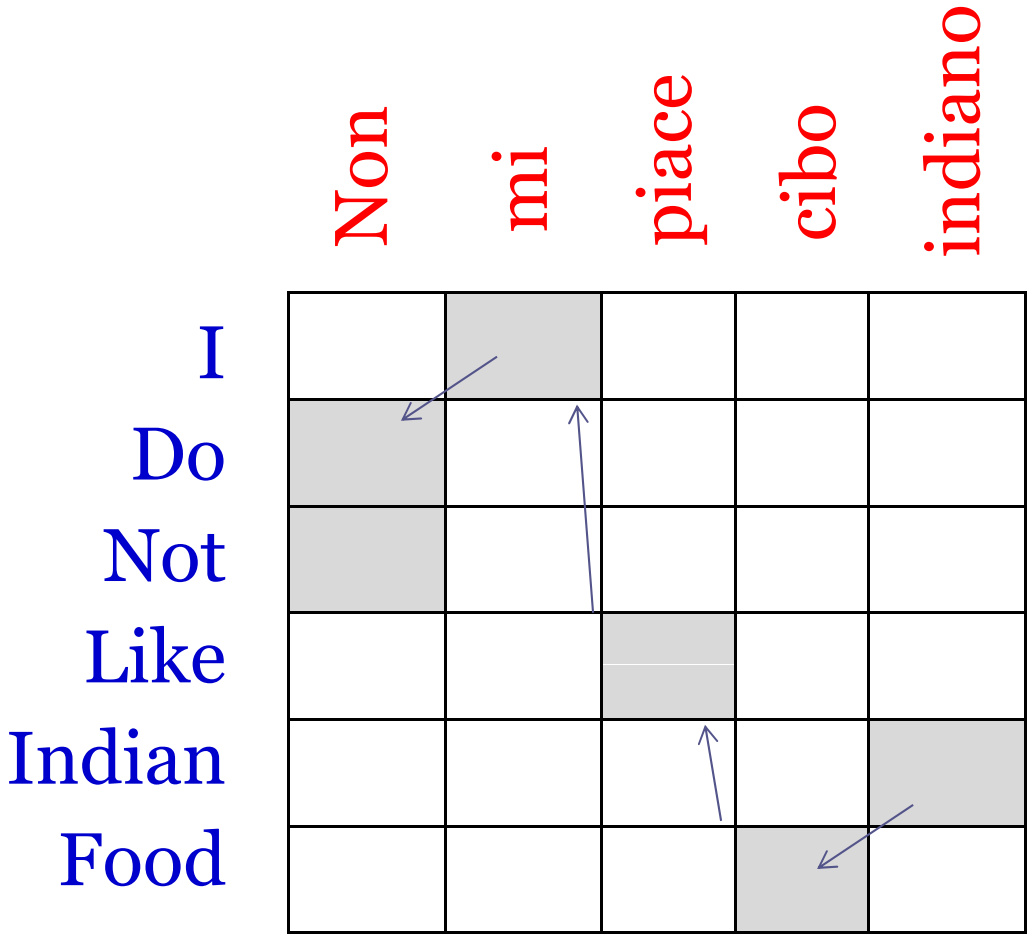
-monotone (m) : two successive phrases in **f** translate into successive phrases in **e**.

-swap (s) : two successive phrases in **f** ($\overline{f}_j, \overline{f}_{j+1}$) translate into two successive phrases ($e_i, e_{i+1}$) of **e** but word alignment is in reverse order.

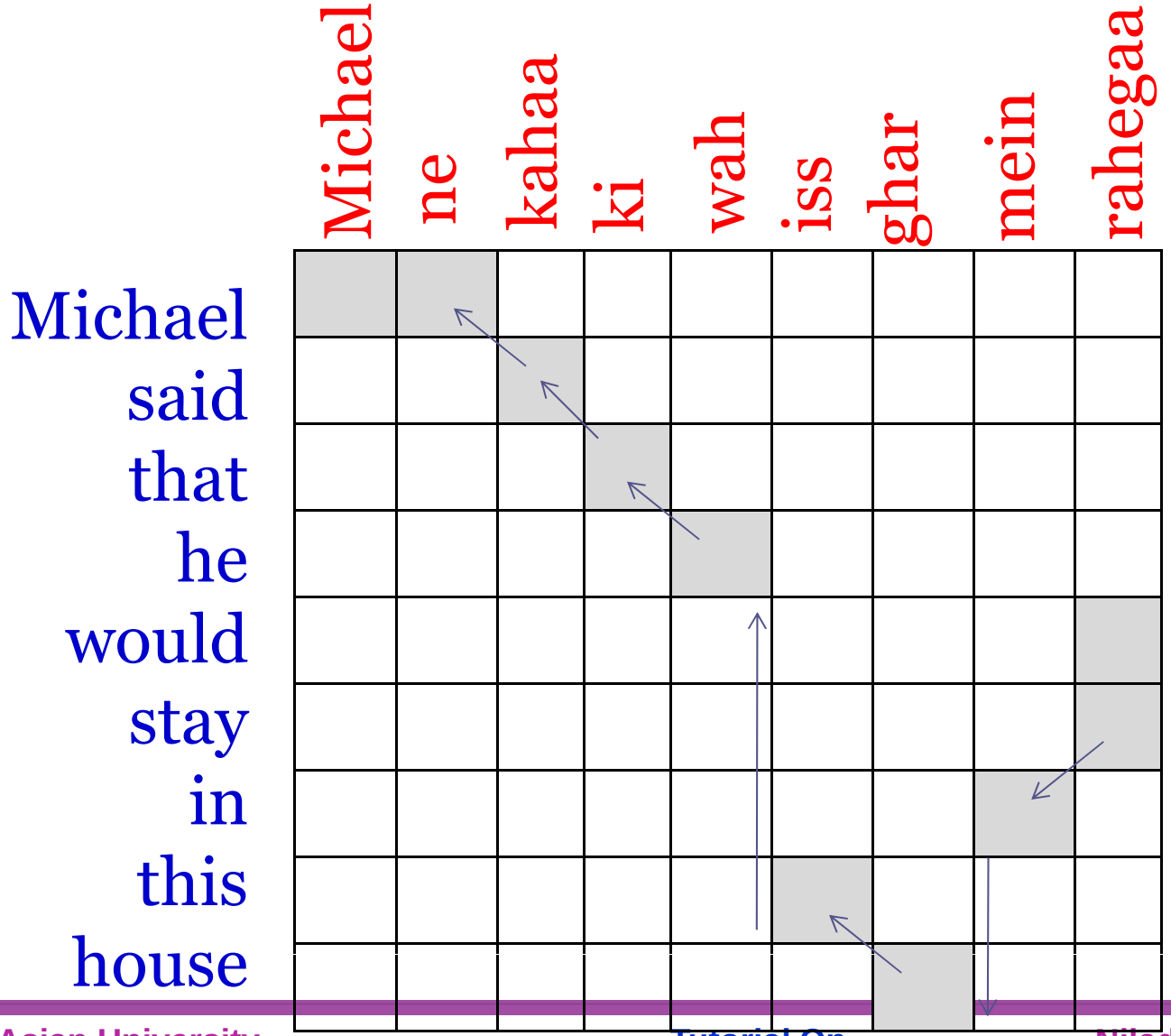
-discontinuous (d): translations of two successive phrases in *f* are not successive in *e*.

*How to obtain ?*

# Lexical Ordering



# Lexical Ordering



# Lexical Reordering

Note:

a) If there is an alignment point at top left of the cell then it is monotone.

b) If there is an alignment point at the bottom left then it is swap.

c) Otherwise it is a discontinuous.

While doing phrase alignment we take the statistics on *Orientation*.

$$p_o(x \mid \overline{f}, \overline{e}) = \frac{count(x, \overline{f}, \overline{e})}{\sum_o count(o, \overline{f}, \overline{e})}, o, x \in \{m, s, d\}$$

Because of data sparseness often some modified Formula/ techniques are also used.

# Lexical Reordering

For example, one can make use of the overall probability of some particular orientation:

Let

$$p_o(x) = \frac{\sum_{\bar{f}}\sum_{\bar{e}} count(x, \bar{f}, \bar{e})}{\sum_{o}\sum_{\bar{f}}\sum_{\bar{e}} count(o, \bar{f}, \bar{e})}, o, x \in \{m, s, d\}$$

Then for a given constant $\alpha$, one can use the following:

$$p_o(x \mid \bar{f}, \bar{e}) = \frac{count(x, \bar{f}, \bar{e}) + \alpha * p_o(x)}{\sum_{o} count(o, \bar{f}, \bar{e}) + \alpha}, o, x \in \{m, s, d\}$$

# Other issues of Reordering

Reordering needs to be dealt with more judiciously:

-It has been noticed that in reordering some groups of words move together

-This happens depending upon the role of the phrase

- There may be *local* reordering and *global*

  *E.g.* intra phrase and inter-phrase

  * hence distance is not the best measurement
  * as it penalizes large movements
  * but some languages demand it ( e.g. SOV vs. SVO)

- Typically reordering is guided by the *Language Model*
- Question is: can a typical 3-gram / 4-gram model can guide long distance reordering?

# *Decoder*

# What is decoding

Mathematical models (word based, phrase based)
Assign probabilistic scores to different possible Translations.

The task of Decoder is to find the best sentence.

But the problem of exhaustive search is NP-complete. [Kevin Knight]

Heuristic search techniques are employed.

Thus though the best solution is not always guaranteed, We expect one very close to that.

# Why Search

Consider: er geht ja nicht nach hause (Gr)

Top 3 translation options of the words (Koehn)

| SL words | Tran 1 | Tran 2 | Tran 3 |
|---|---|---|---|
| Er | he | it | , it |
| Geht | is | are | goes |
| Ja | yes | is | of course |
| Nicht | not | do not | does not |
| Nach | after | to | according to |
| Hause | house | home | chamber |

The actual translation: He does not go home

# Why Search

Therefore a monotone translation with the most probable meanings do not work.

Consider 2-word phrases ( 3 most probables)

| 2-gram phrases | Trans -1 | Trans-2 | Trans-3 |
|---|---|---|---|
| Er geht | it is | he will be | it goes |
| Geht ja | in | are | is after all |
| Ja nicht | not | is not | does not |
| Nicht nach | to | following | not after |
| Nach hause | home | under house | return home |

# Why Search

Similarly one can think of phrases of bigger size and their probabilities.

The message is: there are plenty of choices.

A system trained on the Europarl corpus may have as many as 2727 options for this sentence (Philip Koehn).

The solution space grows at a very fast rate as the Sentence length increase.

Consequently heuristic searching is needed.

# *Decoding Algorithms*

# Hypothesis Expansion

Hypothesis is built Left to Right, one phrase at a time.

Each partial translation made is called a Hypothesis

The process start with Null Hypothesis.

Expansion: Picking a translation option, and construct a new hypothesis.

Hence the search space is progressive.

Also, the above scheme gives the tree-like structure.

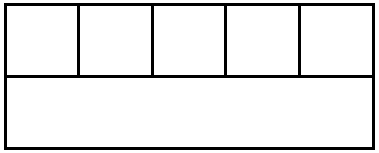
# Hypothesis Expansion

## Illustration:

Hypotheses: A record like structure

Contains information about the current translation and the progress made so far.

Empty hypothesis:



The algorithm starts with Empty Hypothesis

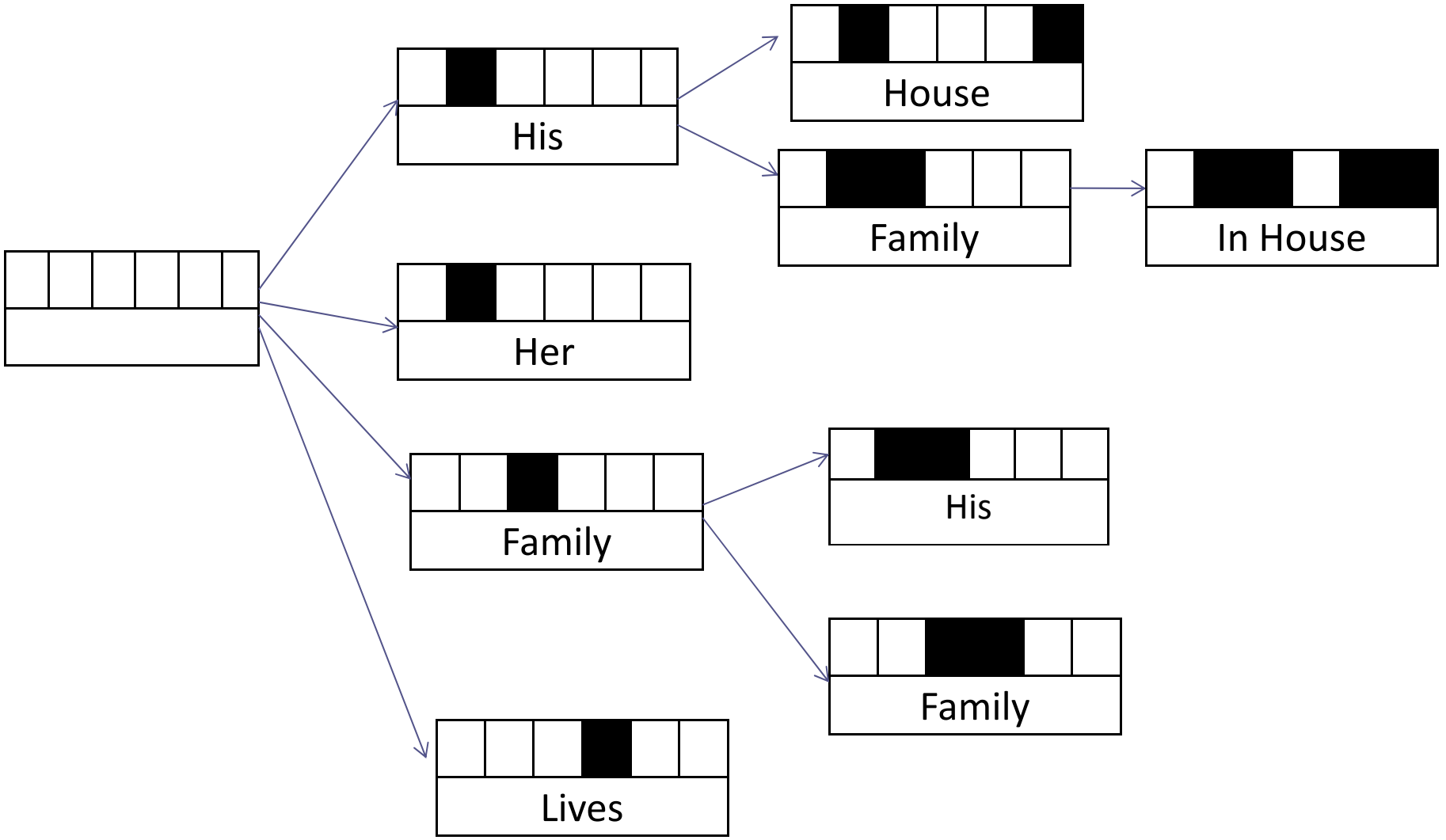
# Illustration

Consider: la sua famiglia resta nella casa

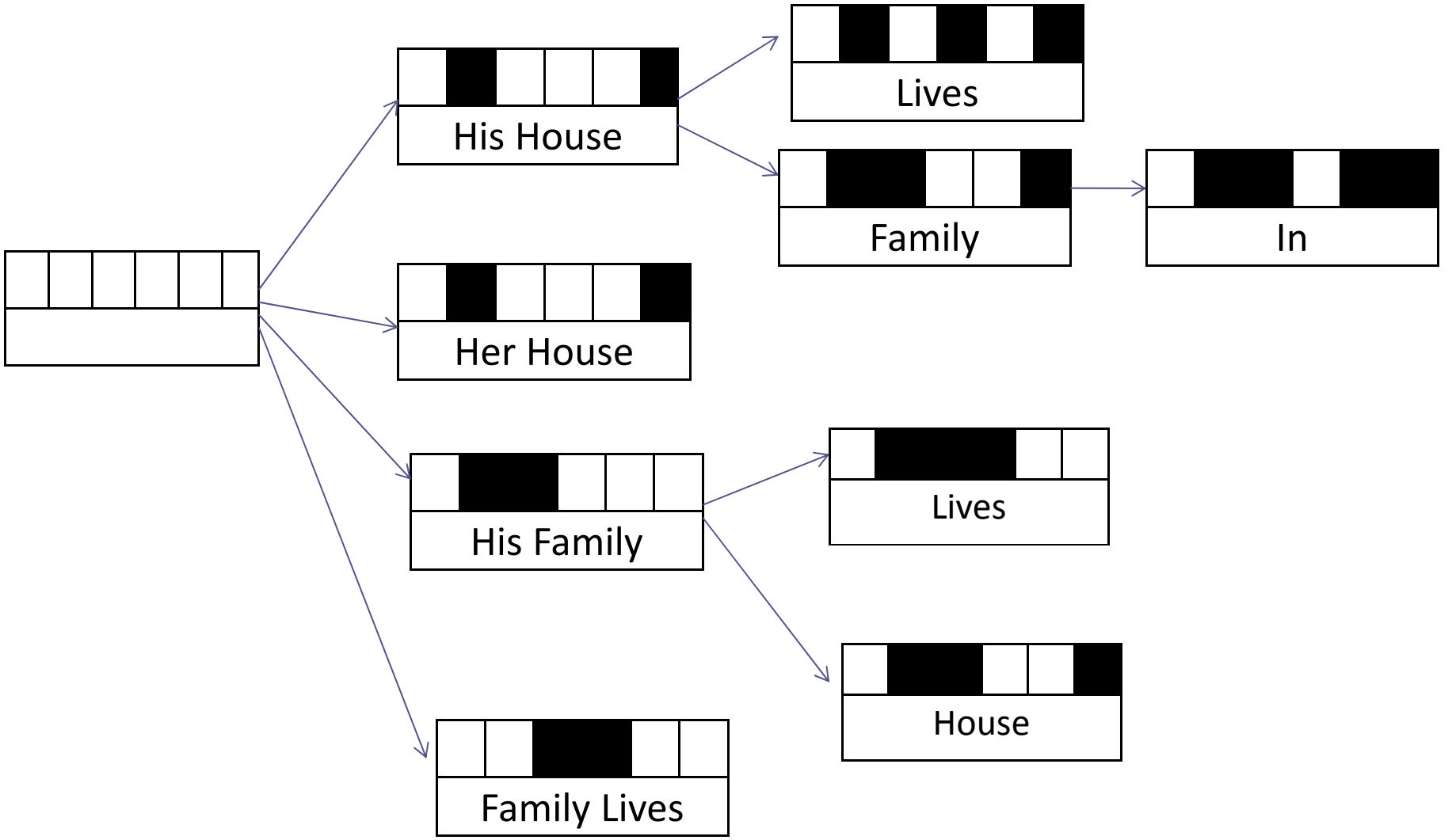
Sua - his her here its
Famiglia - family household home people stock
Resta - remains lives
Casa - house home flat family household

One can pick a word in many ways:
which leads to exponential growth.

# Hypothesis Expansion



# Hypothesis Expansion



# Hypothesis Expansion

Note: Each time one phrase translation is added the score is updated using the three factors:

- Translation Model: Collect the Phrase Translation $\phi$ Probability from the Phrase Translation Table.

- Distortion model: Get the distortion probability $d$ from the start position of the current phrase and end position of the previous phrase translated.

- Language Model: The bi-gram, tri-gram probabilities are used to score the translation generated so far.

Thus at each step partial scores are maintained.

# Hypothesis Expansion

The expansion comes to an end point when there is no More phrase/word to expand.

Thus there will be a large number of endpoints.

The number depends upon : Length of the sentence
No. of options

Different endpoints will have different probabilities.

The probability will depend upon : Language Model
Translation Probs.

Typically, the highest prob. will be chosen.

# Computational Complexity

The best solution can be obtained after building the whole search Space

This is exponential in the size of the input sentence.

This problem is NP-Complete (K. Knight).

Two step reduction of complexity:

- Hypothesis reduction
- Pruning

# Hypothesis Reduction

The same hypothesis can be reached through different Paths: E.g. His Family (In the above example)

Let $x_f >> x_e$ $y_f >> y_e$ and $z_f >> z_e$

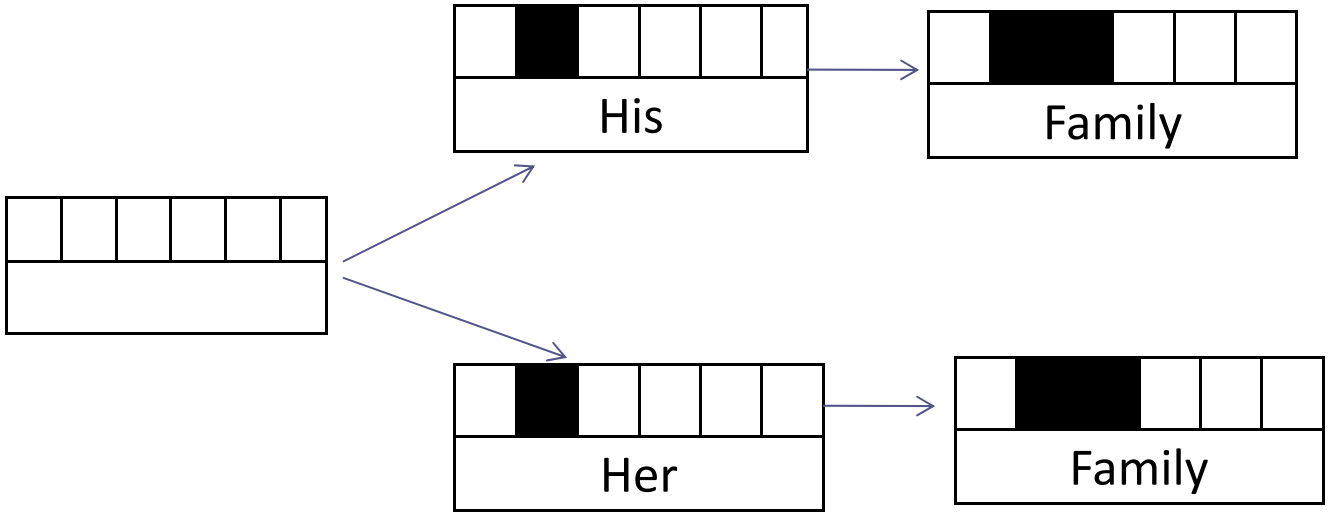
Then the translation decision $[x_f\ y_f\ z_f] >> [x_e\ y_e\ z_e]$
Can be reached in 6 + 3*2 + 1 = 13 possible ways.

Do we need to preserve all?

We can drop the worst hypothesis. It will never be Part of the solution.

# Hypothesis Reduction

Also consider the following situation:



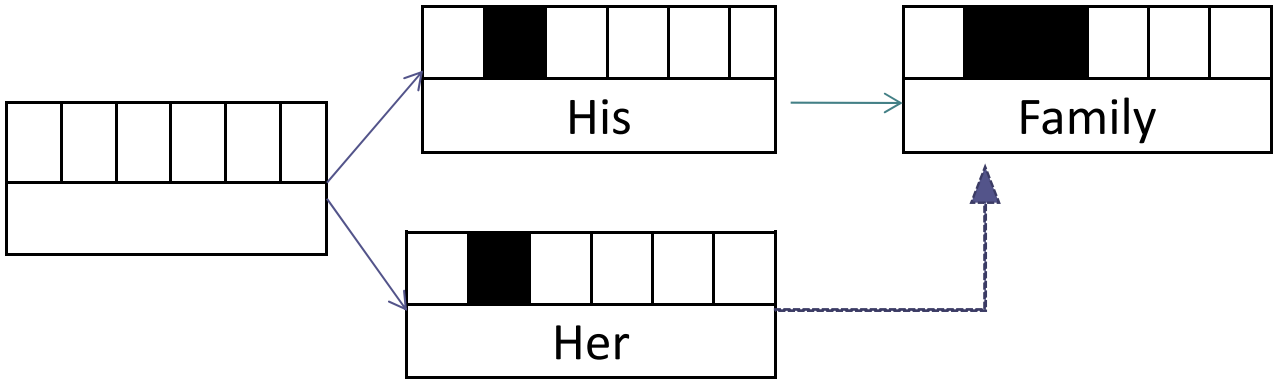

The last production is same for both the path.

Hence both cannot be part of the solution.

Here too we can retain only the better one!!

# Hypothesis Reduction

Typically instead of complete deleting a Link is maintained.




This helps in getting second/third/ … best Solutions.

# Hypothesis Reduction

-Translation model: has no effect on reduction

- Reordering model.

  Requires the start and end positions of the current phrase and last phrase.

  Nodes where they differ cannot be recombined

-Language model.

Nodes with different histories cannot be recombined.

Note: It gives a smarter structure.

Does not reduce exponential complexity.

# *Search Space Reduction*

# The Philosophy

Instead growing the full search space, can we prune some parts?

If a branch does not appear to be probable it can be eliminated. ( Of course has its own risk.)

There will be a search error – if we happen to prune A branch that might yield the final solution.

But exhaustive search is too expensive.

Depends upon the heuristic search technique applied

# Search Space Reduction

Various heuristic search techniques exist:

- Depth First ( exhaustive )
- Breadth First (exhaustive)
- Best First (expands the most promising node chosen according to a specified rule.)
- Hill Climbing (starts with an initial translation. It is recursively improved by changing it in steps)
- Branch& Bound (a branch is stopped from expansion if its max probability is certain to be less than one that is already found )

# Search Space Reduction

However the one that is mostly used is Beam Search.

It is an optimization of best-first search

Best-first search works in the following way:

- orders all partial solutions (states) according to some heuristic
- which attempts to predict how close a partial solution is to a complete solution (goal state).
- Uses already travelled cost and an estimate of the remaining cost.
- Typically implemented on a Priority Queue

# Search Space Reduction

## Beam Search is implemented on stacks.

In Beam search, only a predetermined number of best partial solutions are kept as candidates.

The beam width can either be fixed or variable.

- In a fixed beam width, a maximum number of successor states is kept.
- In a variable beam width, a threshold is set around the current best state.

All states that fall outside this threshold are discarded.

# Stack-Based implementation

n+1 stacks are used, where sentence length is n words

The stacks are numbered 0,1, …, n

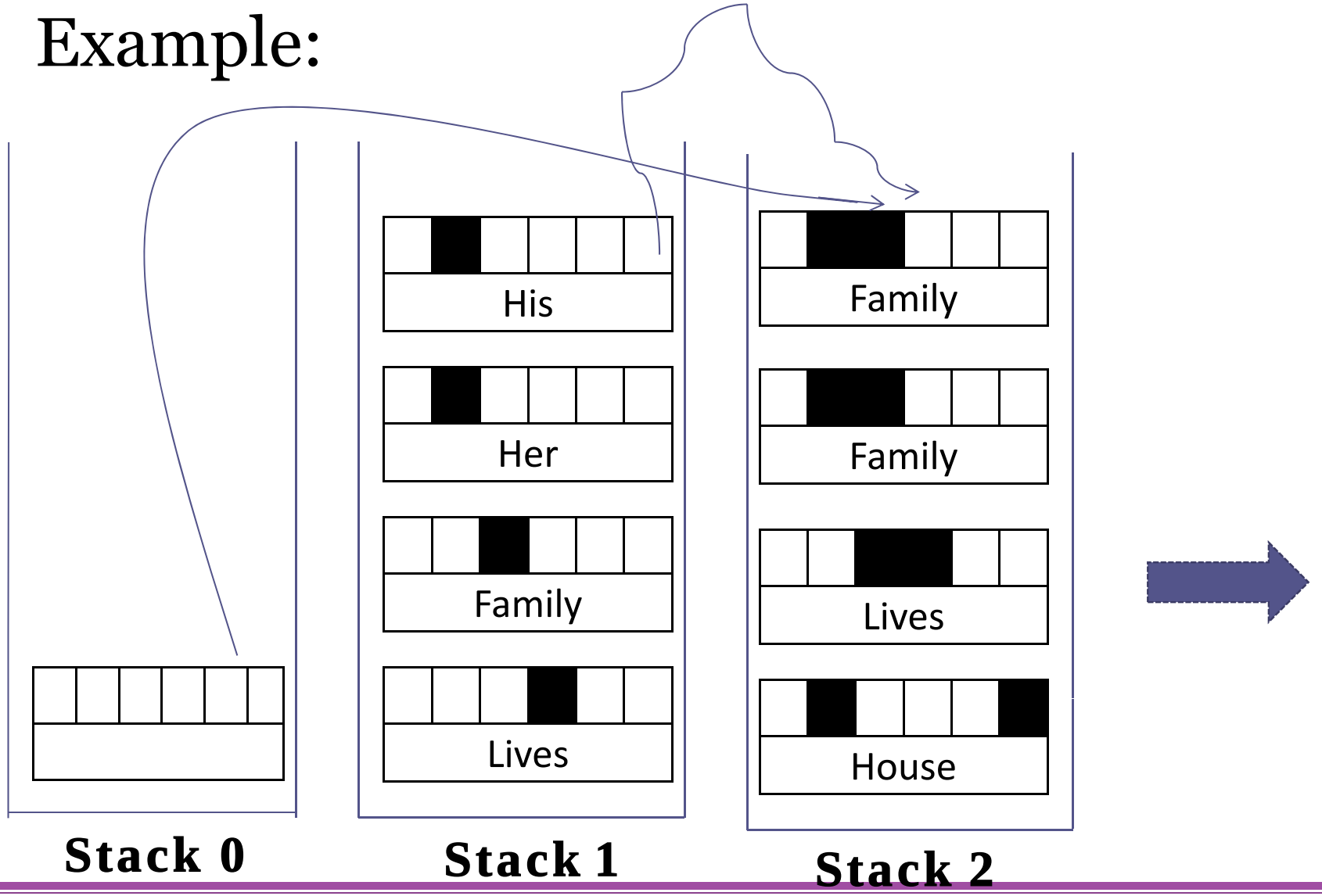
The $i^{th}$ stack contains hypotheses where $i$ original Words have been translated

The progress starts with stack 0, which contains null Hypothesis.

If a stack is full, least probable hypothesis is pruned.

# Stack-Based implementation

Example:



# Stack-Based implementation

The Algorithm

Set stack 0 = null Hypothesis

For i = 0 to n – 1

For j = 0 to $n_i$ // $n_i$ is the no. of hypotheses in stack i

For all translations options

Try option, if applicable

Apply recombination, if applicable

Place hypothesis in appropriate stack

If there is a stack overflow - prune it.

**As each stack has limited capacity only best few retained**

In the lab session we shall see how SMT Can be achieved using

Moses
& Giza++

# MOSES

Moses allows us to automatically train translation models for any language pair.

All that we need is a collection of translated texts (parallel corpus).

Once a trained model is created, an efficient search algorithm quickly finds the highest probability translation among the exponential number of choices.

# *GIZA++* is an extension of the program GIZA - part of Egypt, John Hopkins 1999

The program includes the following extensions to GIZA:

* Model 4; Model 5; Alignment models
* Forward-Backward algorithm, empty word,
* A variant of Model 3 and Model 4
* Various smoothing techniques for fertility,
* Distortion/alignment parameters;
* Significant more efficient training of the fertility models;

# THANK YOU