BITS Pilani

BITS Pilani
Hyderabad Campus

Prof.Aruna Malapati
Department of CSIS

Natural Language Processing (CS F429)

Today's Agenda

- Course Logistics
- Course outline
- Introduction to NLP

General Course Information

Instructor: Prof.Aruna Malapati

Office: H132

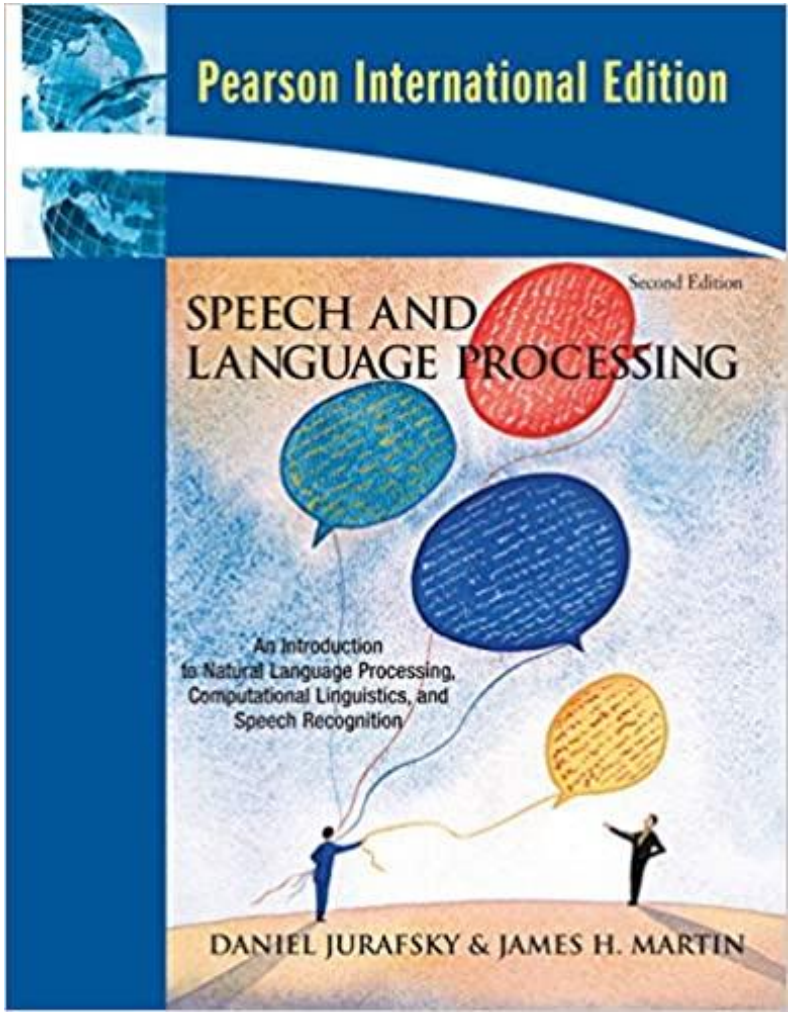
Email: **arunam@hyderabad.bits-pilani.ac.in**

Chamber Consultation : Monday 4:00-5:00pm

Evaluation components

Component	Duration	Weightage	Date & Time	Mode
Project Phase1 evaluation – 10% before mid sem Phase2 evaluation – 15% before compre	Take home	25%	--	Open Book
Mid-Term exam	90 mins	35%		
Comprehensive exam	3 hours	40%	21/12/21	

Text Book



https://web.stanford.edu/~jurafsky/slp3/

Objectives of the course

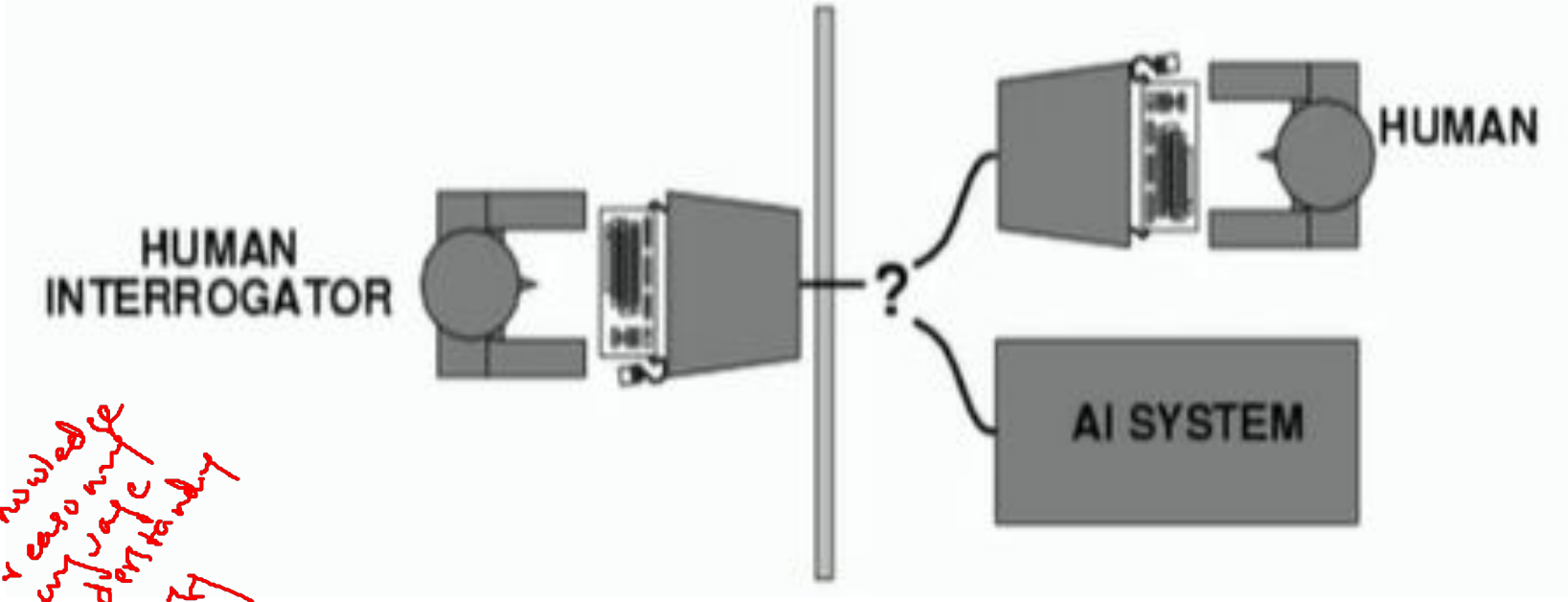
A gentle introduction to NLP problems & solutions including

- Task formulation
- Standard and state of the art approaches
- Applications
- Evaluation measures
- Open issues

Course Outline

- Overview of NLP
- Language models and word representations
- POS Tagging
- Topic Modelling
- Statistical Machine Translation
- Dependency parsing
- NER and question answering systems

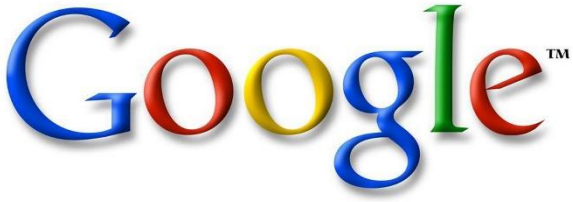
Turing Test



knowledge
reasoning
language
understanding
learning

Can machines think?

Commercial World



Why is NLP hard?

- Ambiguity
- Usage of non standard language
- Segmentation
- Idioms
- New words
- Tricky new entities

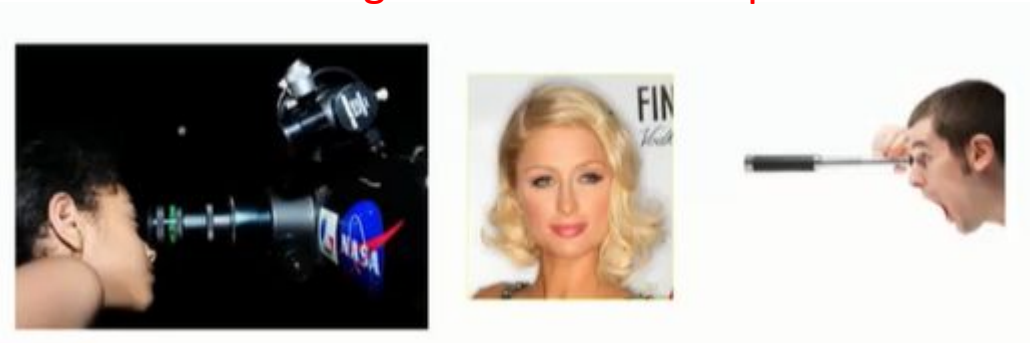
I spilled the beans of our project.

(oov)

Pooja

Why is NLP hard?

I see the girl with the telescope



I drank a cup of chocolate at the bar

Levels of language

- Phonetics/Phonology/Morphology; What words (or sub words) are we dealing with?
- Syntax: What phrases are we dealing with? Which words modify the other?

- Semantics: What's the literal meaning?

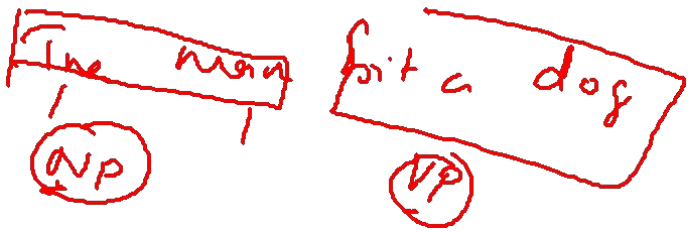
- Pragmatics: What should you conclude from the fact I said something? How to react?

Natural Language Tasks

- Processing natural language text involves many various syntactic, semantic and pragmatic tasks in addition to other problems.

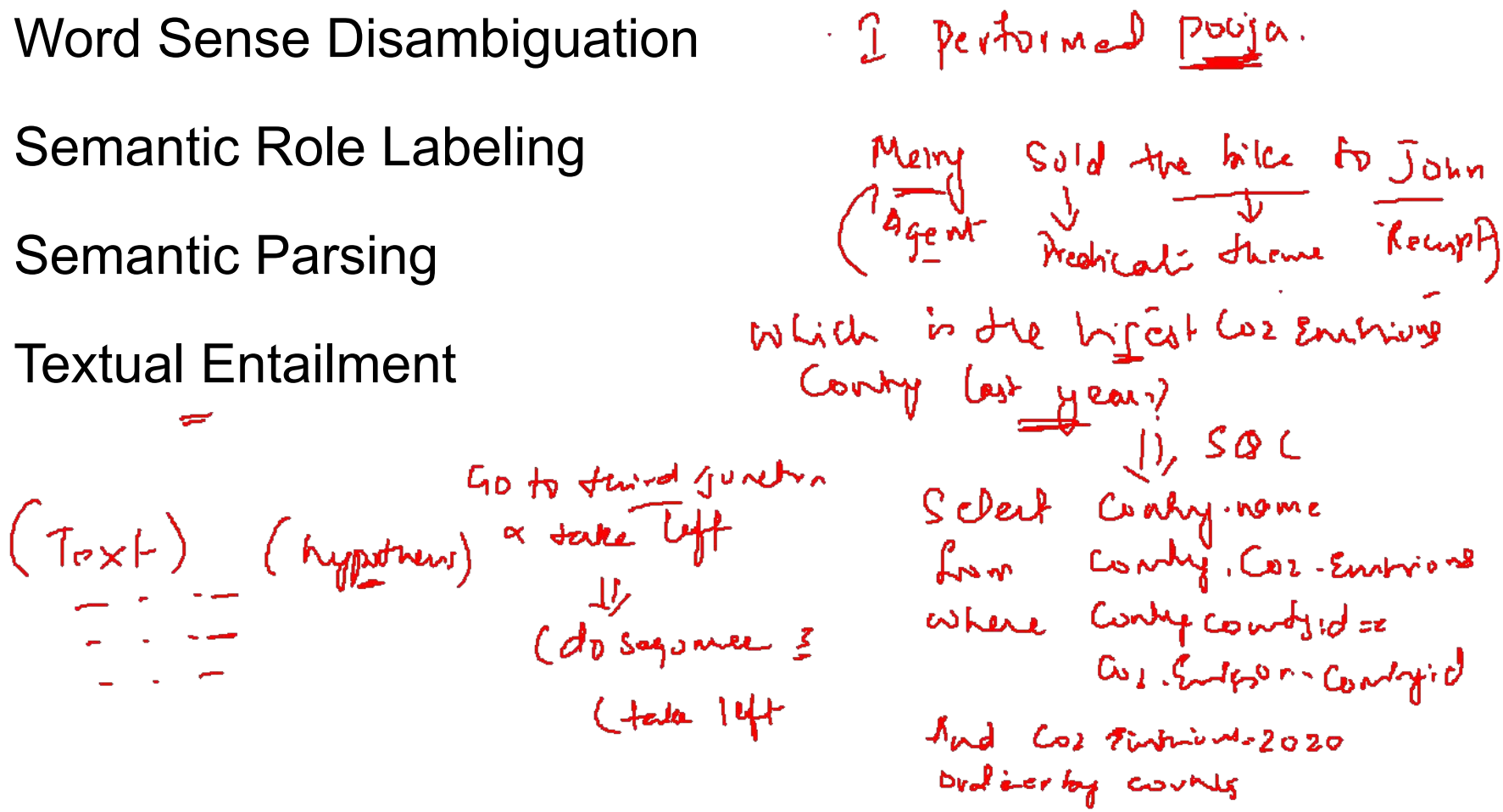
Syntactic Tasks

- Word Segmentation
- Morphological Analysis
- Part Of Speech (POS) Tagging
- Phrase Chunking
- Syntactic Parsing



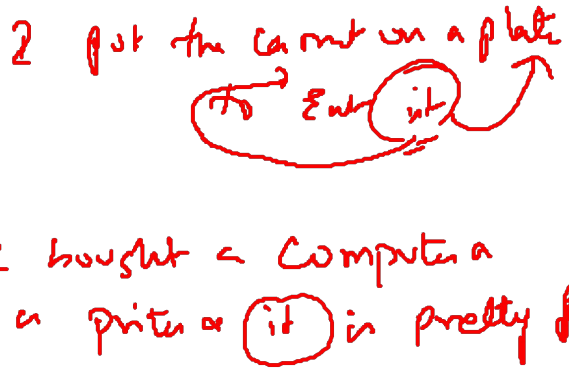
Semantic Tasks

- Word Sense Disambiguation
- Semantic Role Labeling
- Semantic Parsing
- Textual Entailment



Pragmatics/Discourse Tasks

- Anaphora Resolution/Co-Reference
- Ellipsis Resolution



Applications of NLP

- Spelling Correction, grammar checking
- Better search engines
- Speech-to-text and vice versa
- Dialogue systems
- Question Answering
- Machine Translation
- Summarization
- Reading comprehension

SQUAD

Relevant Scientific Conferences

- Association for Computational Linguistics (ACL)
- North American Association for Computational Linguistics (NAACL)
- International Conference on Computational Linguistics (COLING)
- Empirical Methods in Natural Language Processing (EMNLP)
- Conference on Computational Natural Language Learning (CoNLL)
- 18th International Conference on Natural Language Processing December 16-19 2021

BITS Pilani

BITS Pilani
Hyderabad Campus

Prof.Aruna Malapati
Department of CSIS

innovate achieve lead

BITS Pilani
Hyderabad Campus

Language Models

Today's Agenda

- **Language Models**

Slides based on the Stanford "Natural Language Processing" course by Dan Jurafsky and Christopher Manning

Probabilistic Language Models

The goal: assign a probability to a sentence

- Machine Translation:
  - » P(**high** winds tonite) > P(**large** winds tonite)
- Spelling Correction
  - » The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - » P(I saw a van) >> P(eyes awe of an)
- + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:
  - $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$
- Related task: probability of an upcoming word:
  - $P(w_5|w_1,w_2,w_3,w_4)$
- A model that computes either of these:
  - $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.
- Better: **the grammar** But **language model** or **LM** is standard

How to compute P(W)

- How to compute this joint probability:

  – P(its, water, is, so, transparent, that)

- Intuition: let's rely on the Chain Rule of Probability

The Chain Rule: General

- The definition of conditional probabilities

  $P(A \mid B) = P(A, B) / P(B)$

  Rewriting: $P(A, B) = P(A \mid B) P(B)$

- More variables:
  - $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$
- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule: joint probability in sentence

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water) ×

P(so|its water is) × P(transparent|its water is so)

How to estimate these probabilities?

Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

No! Too many possible sentences!

We'll never see enough data for estimating these

Markov Assumption

Andrei Markov

Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

Simplest case: Unigram model

$$P(w_1 w_2 \square\ w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

QUESTION 1

Which is assigned higher probability by a unigram language model for English?

- P(I like ice cream)
- P(the the the the)
- P(Go to class daily)
- P(class daily go to)

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \square\ w_{i-1}) \approx P(w_i \mid w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:
  - "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models

Estimating bigram probabilities

The Maximum Likelihood Estimate (MLE)

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

An Example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

QUESTION 2

If we estimate a bigram language model from the following corpus, **what is P(not|do)?**

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

More examples: Berkeley Restaurant Project sentences

can you tell me about any good cantonese restaurants close by

mid priced thai food is what i'm looking for

tell me about chez panisse

can you give me a listing of the kinds of food that are available

i'm looking for a good place to eat breakfast

when is caffe venezia open during the day

Raw bigram counts (absolute measure)

Out of 9222 sentences

w_{i-1}	i	want	to	eat	chinese	food	lunch	spend
w_i i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Raw bigram probabilities (relative measure)

Normalize by unigrams:

Result:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram estimates of sentence probabilities

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

What kinds of knowledge?

P(english|want) = .0011
P(chinese|want) = .0065
P(to|want) = .66
P(eat | to) = .28
P(food | to) = 0
P(want | spend) = 0
P (i | <s>) = .25

world

grammar

grammar (contingent zero)
grammar (structural zero)

Practical Issues

We do everything in log space

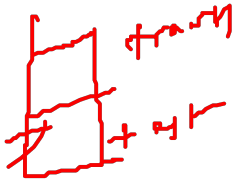
- Avoid underflow: multiplying extremely small numbers
- Adding is faster than multiplying

$$p_1 \times p_2 \times p_3 \times p_4 \Rightarrow \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Evaluation: How good is our model?

Does our language model prefer good sentences to bad ones?

- Assign higher probability to “real” or “frequently observed” sentences
  - Than “ungrammatical” or “rarely observed” sentences?

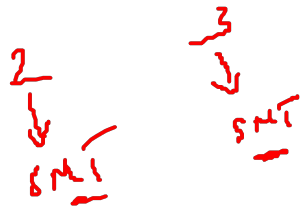


We train parameters of our model on a **training set**.

We test the model’s performance on data we haven’t seen.

- A **test set** is an unseen dataset that is different from our training set, totally unused.
- An **evaluation metric** tells us how well our model does on the test set.

Extrinsic evaluation of N-gram models



Best evaluation for comparing models A and B

- Put each model in a task
  - spelling corrector, speech recognizer, machine translation system
- Run the task, get an accuracy for A and for B
  - How many misspelled words corrected properly
  - How many words translated correctly
- Compare accuracy for A and B

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

Extrinsic evaluation

- Time-consuming; can take days or weeks

So instead

- Sometimes use **intrinsic** evaluation: **perplexity**
- Bad approximation
  - unless the test data looks **just** like the training data
  - So **generally only useful in pilot experiments**
- But is helpful to think about.

Intuition of Perplexity

How well can we predict the next word?

I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

- mushrooms 0.1
- pepperoni 0.1
- anchovies 0.01
- ….
- fried rice 0.0001
- ….
- and 1e-100

– Unigrams are terrible at this game. (Why?)

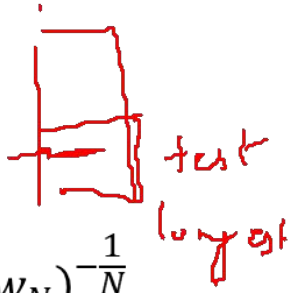
A better model

– is one which assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)



Perplexity is the probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

QUESTION 3

A traffic signal has three colors: green, yellow, and red, which appear with the following probabilities. Using a unigram model, **what is the perplexity of the sequence (green, yellow, red)?**

P(green) = 2/5

P(yellow) = 1/5

P(red) = 2/5

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

Lower perplexity = better model

Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Pretrained LMS

SRILM:

www.speech.sri.com/projects/srilm

Google N-Gram Release, August 2006, dataset details:

Over a trillion words

Over a billion 5-grams (c >= 40)

Over 13 million unique words (c >= 200)

Google Books n-gram viewer:

http://ngrams.googlelabs.com

Summary

- Importance of Language models
- How to define a probabilistic LM?
- LMs can be measured using perplexity

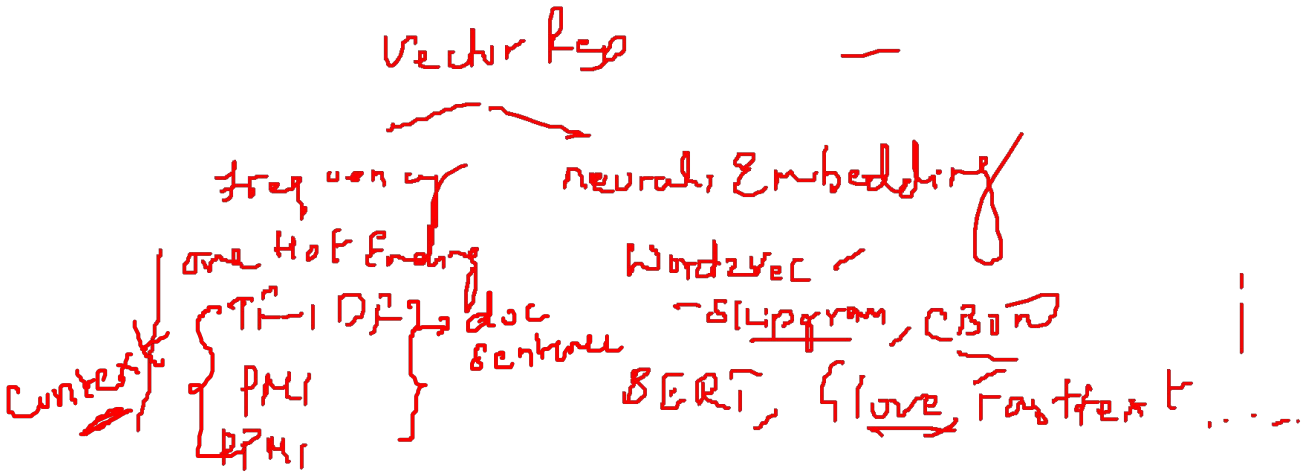
Word Representations

Today's Agenda

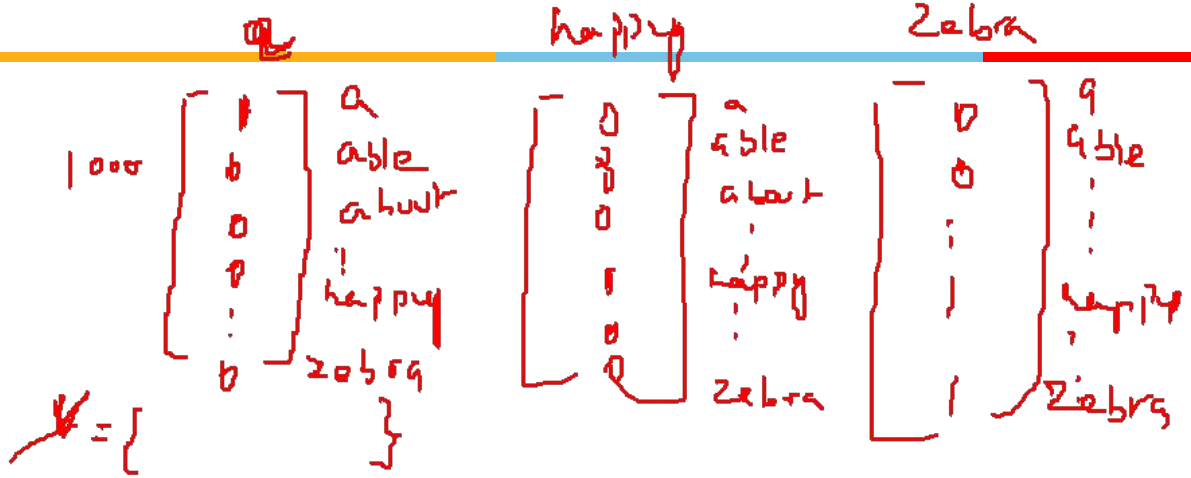
- Representing words as vectors
- One hot vector
- TF-IDF Vectors
- Co-occurance vectors

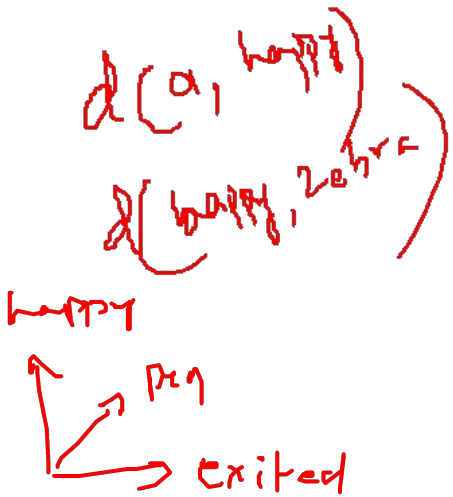
Why convert word to vectors?

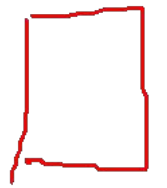
- Most of the Machine Learning algorithms use numbers as inputs.
- In the NLP world we are dealing with characters / Text hence we methods to convert the Text into numbers.



One hot encoding







TF-IDF Vectors

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

tf-idf weighting for document and queries

Example :D1: The restaurants are in the city.
D2:The resorts are in the outskirts.
V={are, city, in outskirts, resorts, restaurants, the}
Q= {resorts,outskirts}

$$Score(q,d) = \sum_{t \in q \cap d} tf.idf_{t,d}$$

	TF-D1	IDF-D1	TF-IDF(D1)	TF-D2	IDF-D2	TF-IDF(D2)	TF-Query	IDF-Query	TF-IDF Query
are	1+log 1=1	log2/2=0	0	1+log 1=1	log2/2=0	0	0	log2/2=0	0
city	1+log 1=1	log2/1=0.3	0.3	0	log2/1=0.3	0	0	log2/1=0.3	0
in	1+log 1=1	log2/2=0	0	1+log 1=1	log2/2=0	0	0	log2/2=0	0
outskirts	0	log2/1=0.3	0	1+log 1=1	log2/1=0.3	0.3	1+log 1=1	log2/1=0.3	0.3
resorts	0	log2/1=0.3	0	1+log 1=1	log2/1=0.3	0.3	1+log 1=1	log2/1=0.3	0.3
restaurants	1+log 1=1	log2/1=0.3	0.3	0	log2/1=0.3	0	0	log2/1=0.3	0
the	1+log 2=1.3	log2/2=0	0	1+log 2=1.3	log2/2=0	0	0	log2/2=0	0

Pointwise Mutual Information (PMI)

Pointwise mutual information: Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989) Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Word-Word Matrix

Context: ± 7 words

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

Resulting word-word matrix:

f(w, c) = how often does word w appear in context c:
"information" appeared six times in the context of "data"

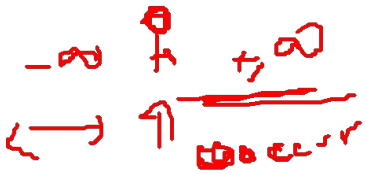
	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

PMI Vectors

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

- pmi(information,data) = $\log_2$ (.32 / (.37*.58)) = .57



PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

What are simple tests for understanding meaning?

Which of the following means same as pulchritude:
(a) Truth (b) Disgust (c) Anger (d) Beauty?

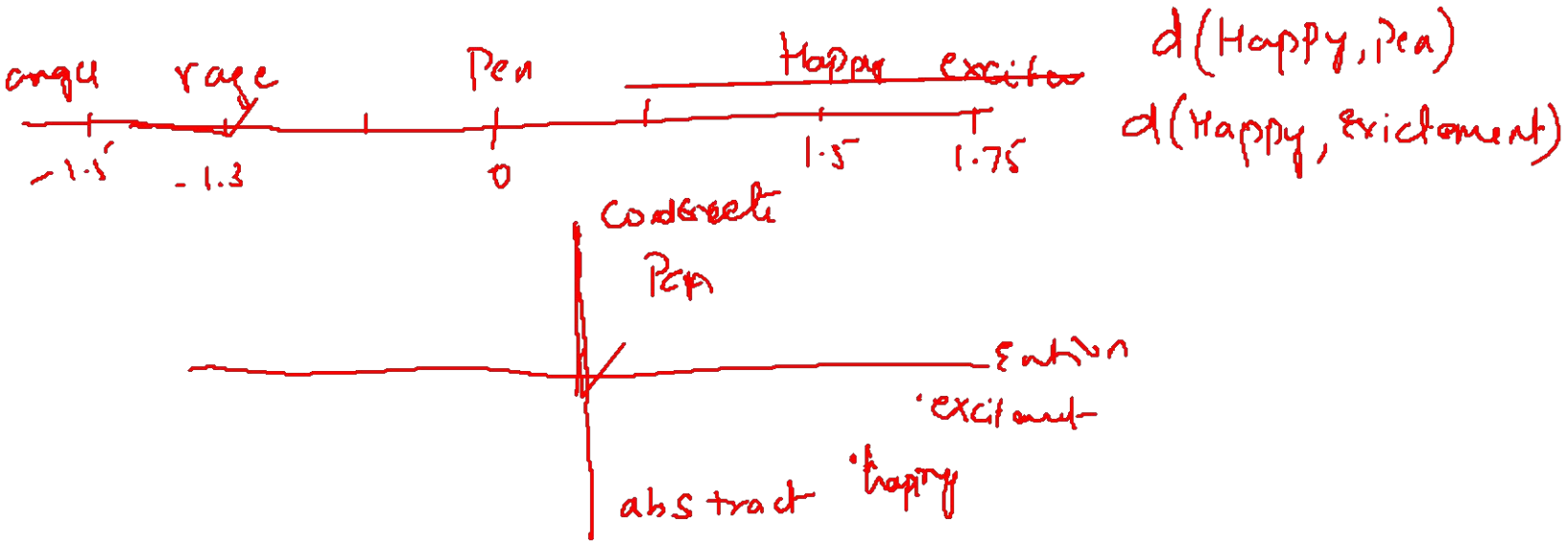
Analogy questions:
Man : Woman :: King : ??

"How can we teach a computer the notion of word similarity?"

Neural Word Embeddings

- A word embedding is a numerical representation of a word.

- These vectors carry the meaning of the word or embed meaning and are of low dimensions.



Two ingredients for generating word embeddings

Training

Corpus

Generic Wikipedia

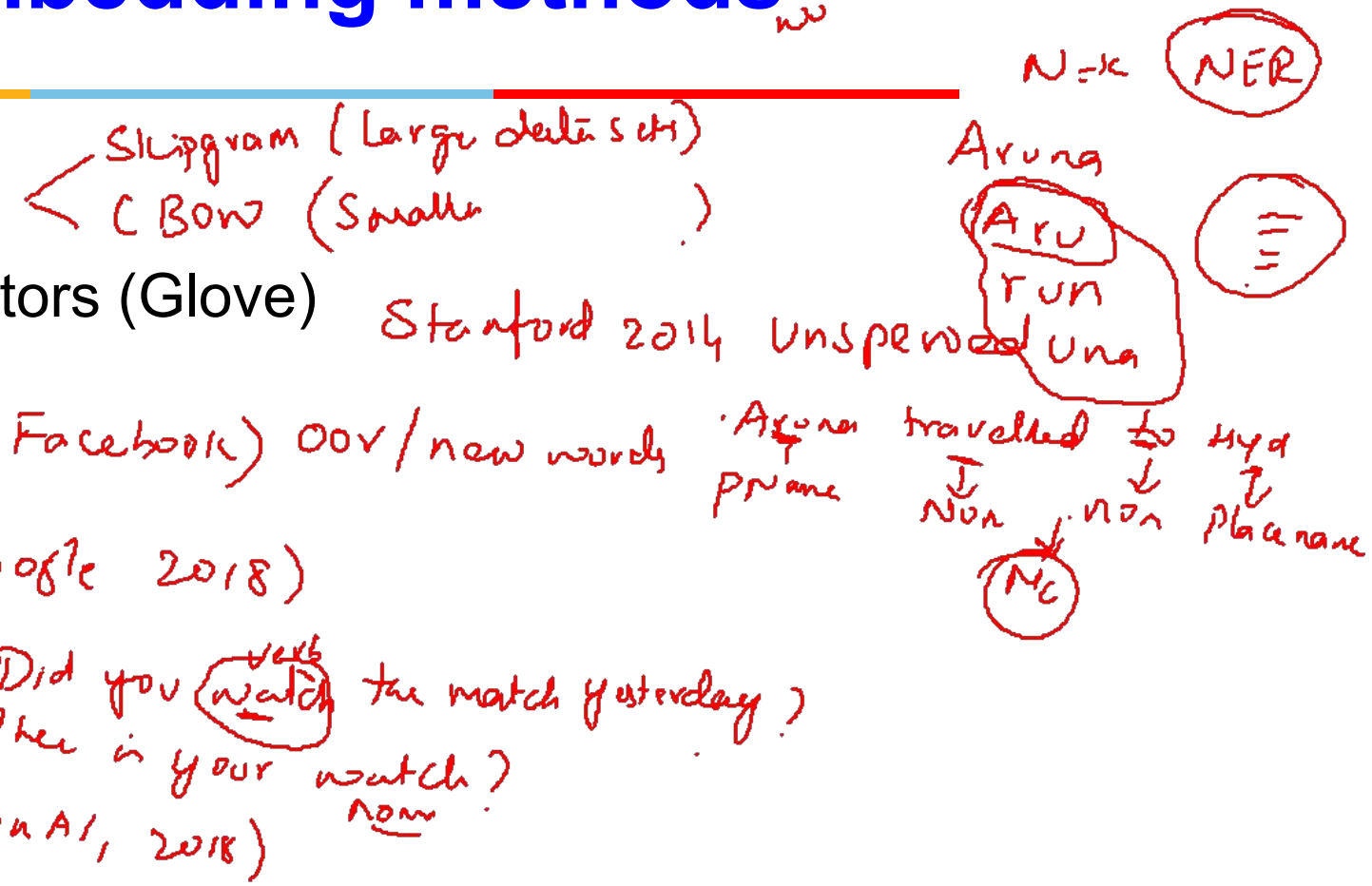
Domain – Tourism
Medical
Legal

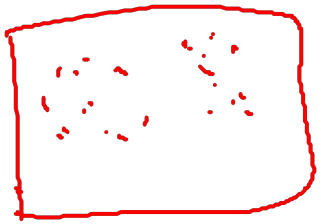
Embedding Method

ML / Neural Networks

Word embedding methods

- Word2Vec
- Global Vectors (Glove)
- FastText
- BERT
- ELMO
- GPT-2



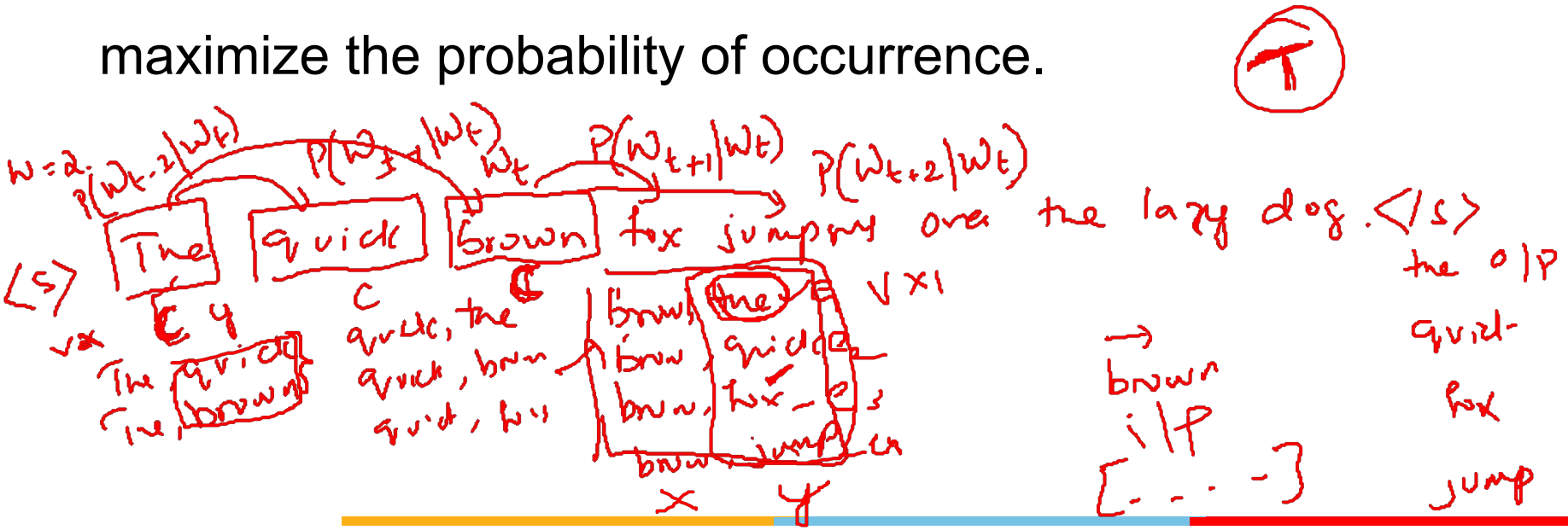


The word representations computed using neural networks are very interesting because the learned vectors explicitly encode many linguistic regularities and patterns. Somewhat surprisingly, many of these patterns can be represented as linear translations.
For example, the result of a vector calculation
vec(“Madrid”) - vec(“Spain”) + vec(“France”) is closer to vec(“Paris”) than to any other word vector [9, 8].

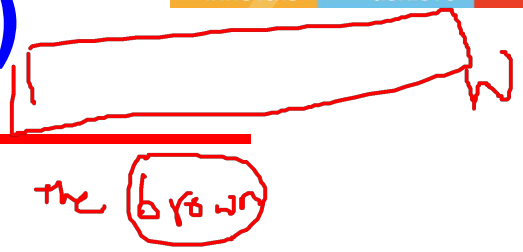
https://arxiv.org/pdf/1310.4546.pdf

Skipgram Model

- A word can be used to predict its surrounding words in a text corpus
- Given a center word it tries to predict the conditional probability of the neighboring words and further maximize the probability of occurrence.



Skipgram Model (Contd..)



$P(\text{the}, \text{quick}, \text{jumps}, \text{fox} \mid \text{brown}) =$

$P(\text{the} \mid \text{brown}) \times P(\text{quick} \mid \text{brown}) \times P(\text{fox} \mid \text{brown}) \times P(\text{jumps} \mid \text{brown})$

$$L(\theta) = \prod_{i=1}^{T} \prod_{-m \le j \le m} P(w_{t+j} \mid w_t)$$

Loss function / objective function

Each word in the corpus

$j \neq 0$: each word within the window of size m

$\theta : MP$

$\theta \in R^{2d|V|}$

$$\theta = \begin{bmatrix} V_{apple} \\ \vdots \\ V_{200} \\ U_{apple} \\ \vdots \\ U_{zebra} \end{bmatrix}$$

$MLE = \text{argmax}\, L(\theta, x)$

$\text{argmax}_x(x) \approx \text{argmin}_x(-x)$

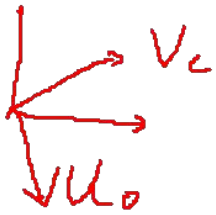
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t ; \theta)$$

$$\frac{\partial J(\theta)}{\partial w} \qquad \frac{\partial J(\theta)}{\partial w'}$$

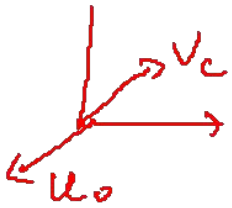
$P(w_{t+j} \mid w_t)$?

u_o v_c

$v_c \cdot u_o \sim 1$ $\quad u_o^T v = \sum_{i=1}^{n} u_i v_i$ $\quad v_c \cdot u_o \in [-1, 1]$



$v_c \cdot u_o = 0$



$v_c \cdot u_o \sim -1$

$0 - 1$

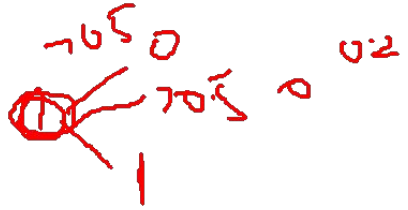
$$P(u_o|v_c) = \frac{\exp(v_c \cdot u_o)}{\sum \exp(v_c \cdot u_o)}$$

$w_t = v_c$

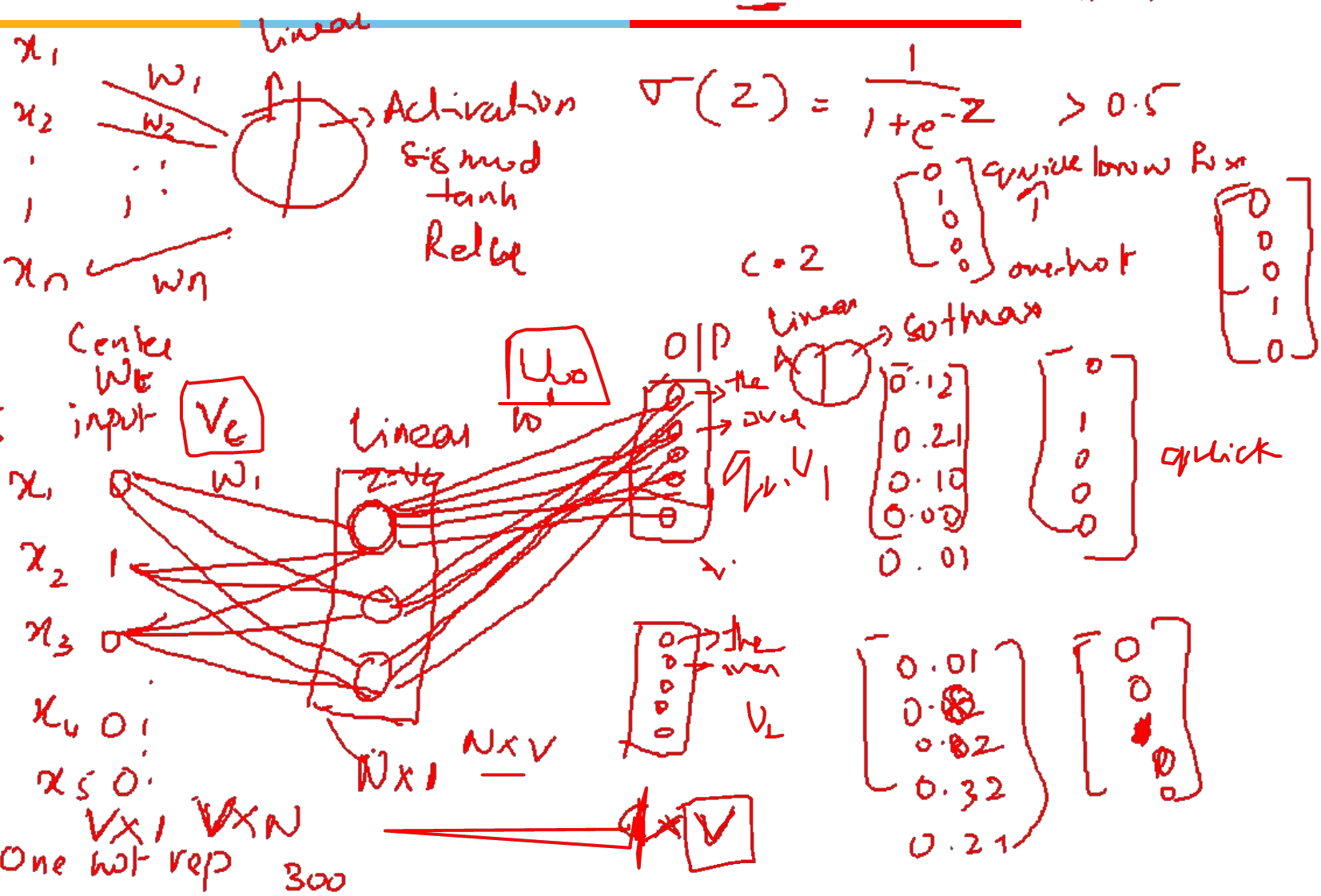
$w_{t+j} = u_o$

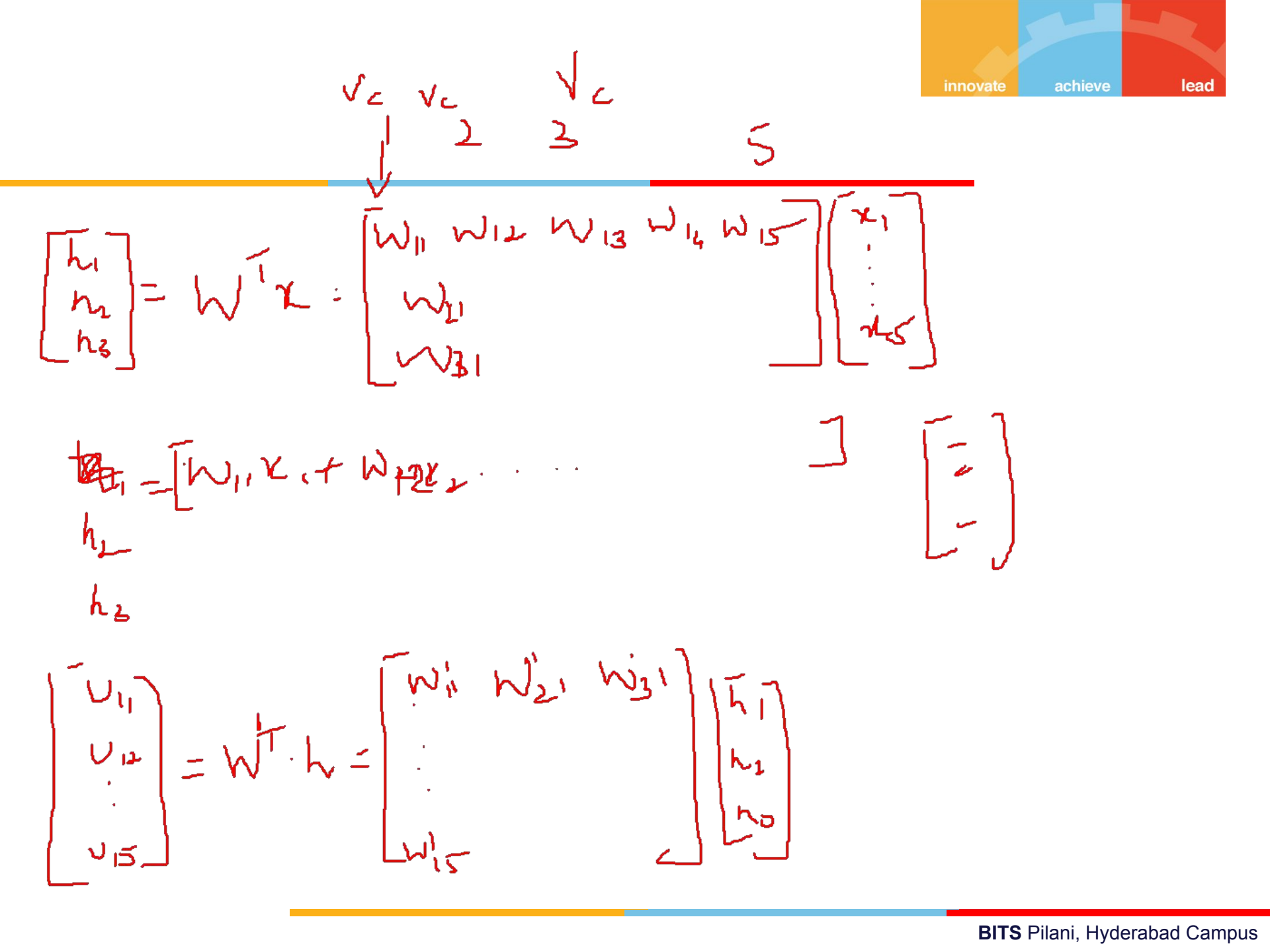
$+, -, 0$ (>1) → Softmax → $[0, 1]$

$$\sigma(\vec{z}_i) = \frac{e^{z_j}}{\sum_{j=k} e^{z_j}}$$



$$z = w_1x_1 + w_2x_2 + w_3x_3 \ldots w_nx_n$$




v_c v_c v_c
1 2 3 5
$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = W^T x = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & & & & \\ w_{31} & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_5 \end{bmatrix}$
$h_1 = [w_{11}x_1 + w_{12}x_2 \cdots]$
h_2
h_3
$\begin{bmatrix} v_{11} \\ v_{12} \\ \vdots \\ v_{15} \end{bmatrix} = W^T \cdot h = \begin{bmatrix} w'_{11} & w'_{21} & w'_{31} \\ \vdots & & \\ w'_{15} & & \end{bmatrix} \begin{bmatrix} h_1 \\ h_1 \\ h_0 \end{bmatrix}$

Example – forward propagation

Input word one hot encoding

x1	the	0
x2	quick	1
x3	brown	0
x4	fox	0
x5	jumps	0
x6	over	0
x7	lazy	0
x8	dog	0

X

Initial weight matrix for center words

-0.78	0.018	0.033
0.068	0.17	-0.109
-0.158	-0.081	-0.151
0.15	0.064	0.145
-0.097	-0.055	0.188
0.036	0.071	0.059
0.168	-0.06	-0.58
0.098	0.015	0.096

=

Hidden Layer

0.068
0.17
-0.109

Hidden Layer

0.068
0.17
-0.109

X

Initial weight matrix for context words

0.192	0.176	0.012
0.07	0.061	-0.046
-0.066	0.117	0.083
0.014	0.006	-0.044
-0.012	0.067	0.147
0.013	0.111	-0.097
0.016	0.175	-0.198
-0.028	-0.016	0.148

=

0.042
0.2
0.006
0.007
-0.005
0.03
0.052
-0.021

Softmax

0.128
0.125
0.124
0.124
0.122
0.127
0.13
0.12
Sum=1

Example-Backward Propagation: Sum of Prediction Errors

C different prediction errors are computed, then summed up. For simplicity let the window size be 1. In which case we are trying to predict two words.

Y-Pred	Y-true		Error_the
0.128	the	1	-0.872
0.125	quick	0	0.125
0.124	brown	0	0.124
0.124	fox	0	0.124
0.122	jumps	0	0.122
0.127	over	0	0.127
0.130	lazy	0	0.130
0.120	dog	0	0.120

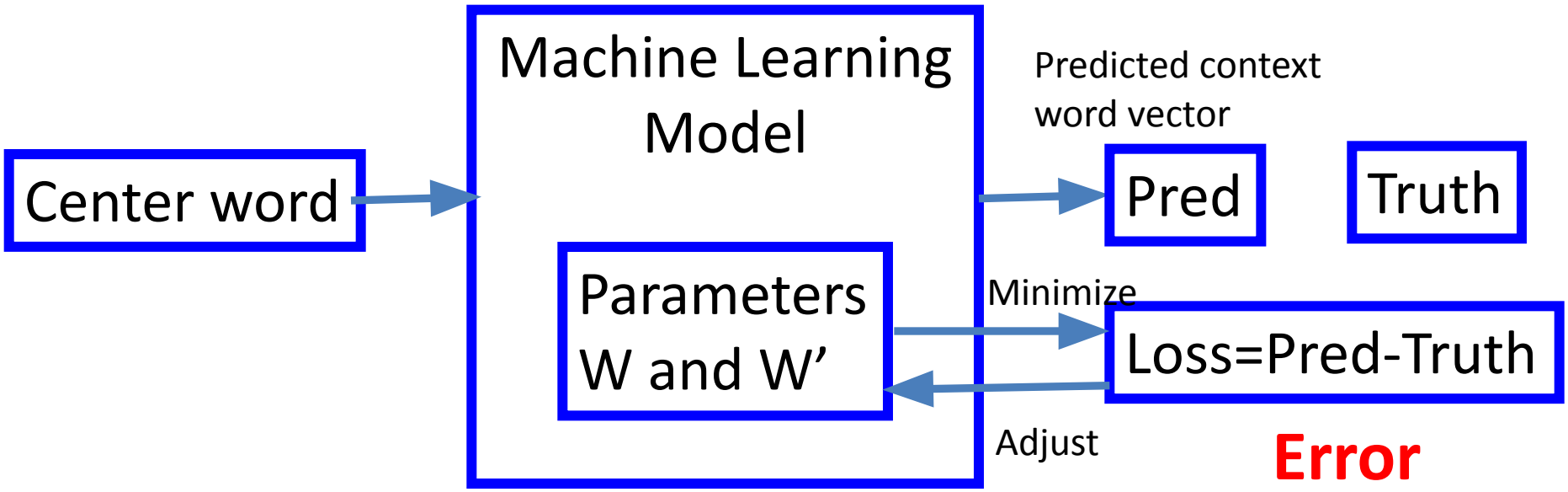
+

Y-Pred	Y-true		Error_brown
0.128	the	0	0.128
0.125	quick	0	0.125
0.124	brown	1	-0.876
0.124	fox	0	0.124
0.122	jumps	0	0.122
0.127	over	0	0.127
0.130	lazy	0	0.130
0.120	dog	0	0.120

=

Error_Sum
-0.744
0.25
-0.752
0.248
0.244
0.254
0.26
0.24

Loss


Machine Learning Model
Center word
Parameters W and W'
Predicted context word vector
Pred
Truth
Minimize
Loss=Pred-Truth
Adjust
Error

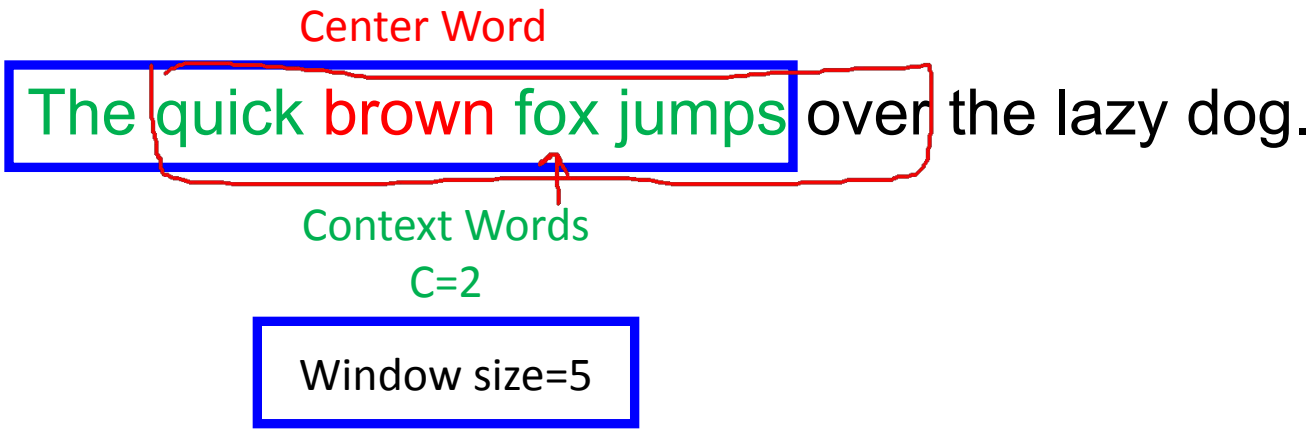
Hyper parameters

- Dimensionality: dimensions of the word vectors
- Window Size: how big of a window you want around the center word
- Minimum Count: how many times a word has to appear in the corpus for it to be assigned a vector (if a word happens too few times, it is difficult to assign it a good vector)
- ModelType: skip-gram or continuous bag-of-words
- Number of Iterations: number of iterations (epochs) over the corpus

Continuous Bag Of Words(CBOW)

- The objective of this model is to predict a missing word based on the surrounding words.
- The idea is that if two unique words are frequently surrounded by similar set of words in various sentences then the two words are semantically similar.
- For example “The little ____ is barking”

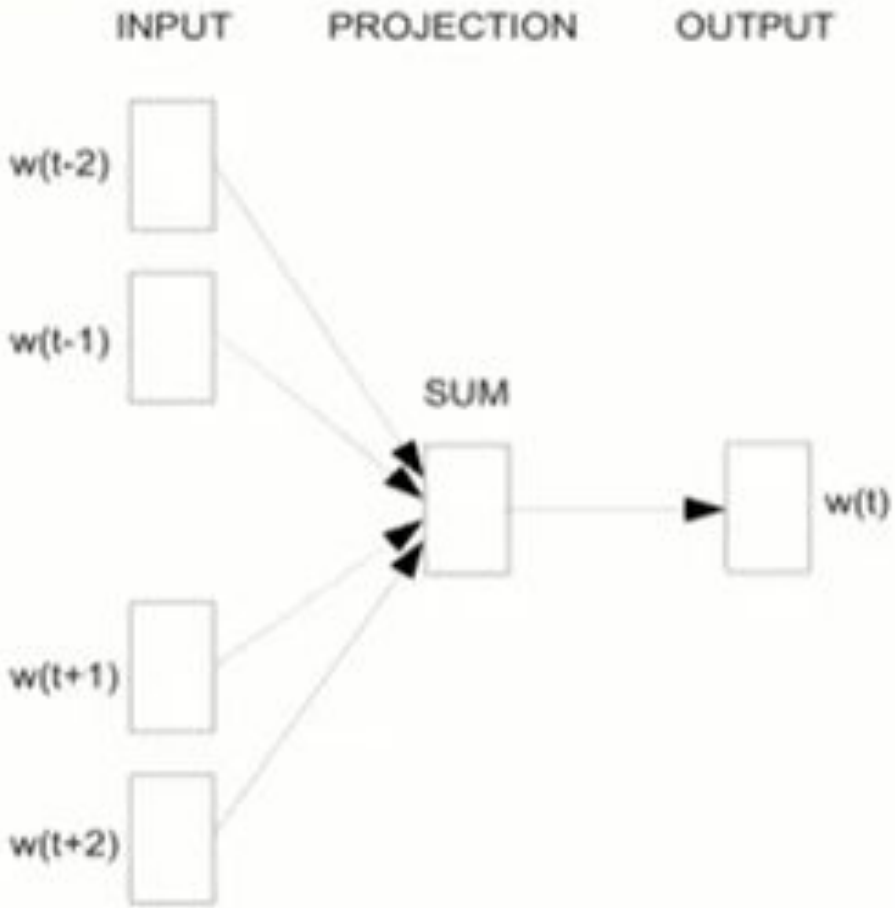
Training data



Training Examples

Context words (Input)	Center Word(output)
The quick fox jumps	brown
quick brown jumps over	fox
brown fox over the	jumps
fox jumps the lazy	over

Continuous Bag Of Words(CBOW)



Source: Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space

Converting center words and context words into vectors

The quick brown fox jumps over the lazy dog.

Vocabulary = brown,dog,fox,jumps,lazy,over,quick,the}

Center words will be one hot encoded

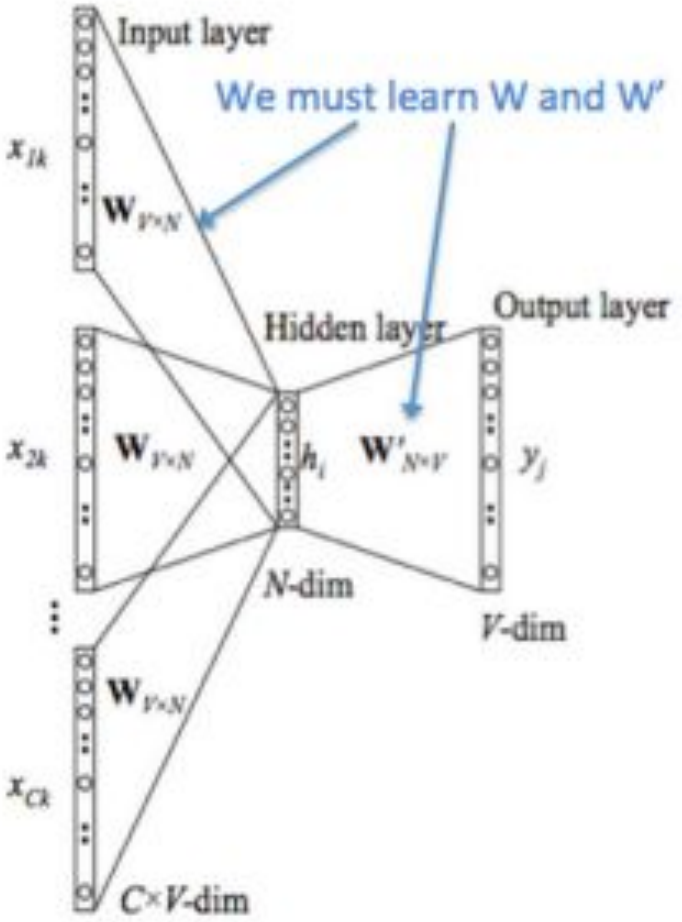
brown	1
dog	0
fox	0
jumps	0
lazy	0
over	0
quick	0
the	0

Context words will be average of one hot vectors

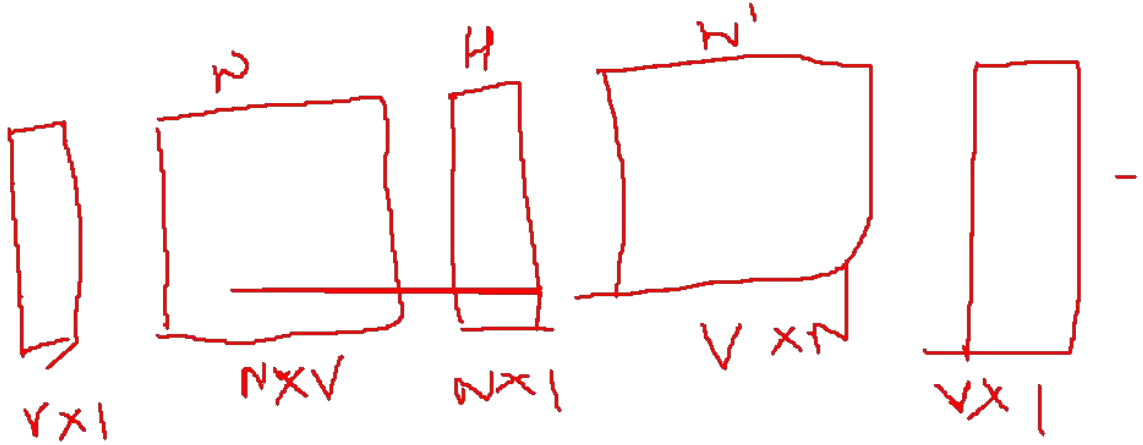
brown	1	brown	0	brown	0	brown	0		brown	0.25
dog	0	dog	0	dog	0	dog	0		dog	0
fox	0	fox	0	fox	1	fox	0		fox	0.25
jumps	0	jumps	0	jumps	0	jumps	1		jumps	0.25
lazy	0	lazy	0	lazy	0	lazy	0		lazy	0
over	0	over	0	over	0	over	0		over	0
quick	0	quick	1	quick	0	quick	0		quick	0.25
the	1	the	0	the	0	the	0		the	0.25

The quick fox jumps

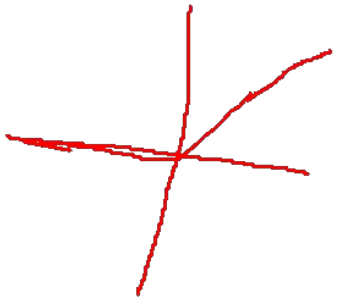
Architecture of CBOW



Dimensions of the vectors and Matrices



$$Relu(x) = Max(0, x)$$

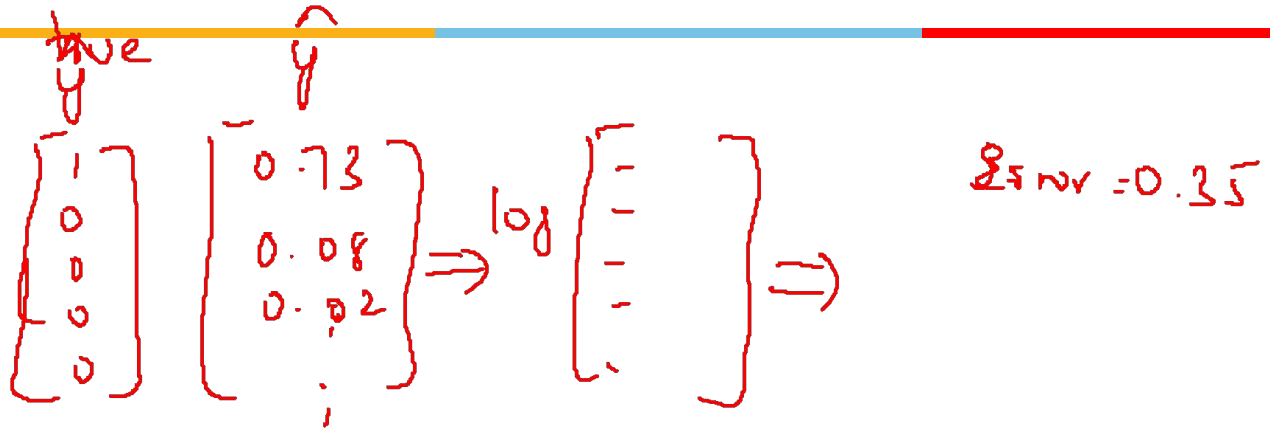


$$\begin{bmatrix} 0.14 \\ -0.23 \\ 0.53 \\ -0.43 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0 \\ 0.53 \\ 0 \end{bmatrix}$$

Cross entropy loss function

$$H(\hat{y}, y) = -y_i \log(\hat{y}_i)$$

$$\begin{aligned}
\text{minimize } J &= -\log P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c | \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}$$


true
ŷ
0.73
0.08
0.02
log
Error = 0.35

Other Work on Word Embeddings

- Active research area
- Using sub word information (e.g. characters) in the word embeddings
- Tailoring embeddings for different NLP tasks

BITS Pilani

BITS Pilani
Hyderabad Campus

Prof.Aruna Malapati
Department of CSIS

Glove Word Representations

Today's Agenda

- Representing words as vectors
- One hot vector
- TF-IDF Vectors
- Co-occurance vectors

The GloVe Method

- Unsupervised algorithms make use of the statistical information present in the word occurrences in the corpus.
- GloVe considers global context
  - Utilizes a co-occurrence matrix to capture global statistics

Matrix of word-word co-occurrence counts

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	i	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

How can we generate meaning from these statistics?

Notations used

X - matrix of word-word co-occurrence counts.

X_{ij} - number of times word j occurs in the context of word i.

$X_i = \sum_k X_{ik}$ - number of times any word appears in the context of the word i.

$P_{ij} = P(j|i) = X_{ij}/X_i$ - the probability that word j appear in the context of word i.

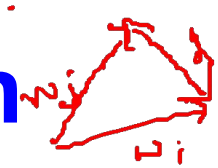
Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Table: Co-occurrence probabilities for target words and context words.

Cost function

- The table suggests that the appropriate starting point for word vector learning should be with ratios of co-occurrence probabilities rather than the probabilities themselves.

GloVe cost function



- Set a function F that represents ratios of co-occurrence probabilities rather than t he probabilities themselves.

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}, \qquad (1)$$

Note that *wi* and *w'k* are vectors from different vector-spaces

- We would like F to encode the information present the ratio P_{ik}/P_{jk} in the word vector space. Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences.

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}. \qquad (2)$$

GloVe cost function

- We note that the arguments of F in Eqn. (2) are vectors while the right-hand side is a scalar.
- While F could be taken to be a complicated function parameterized by, e.g., a neural network, doing so would obfuscate the linear structure we are trying to capture.
- To avoid this issue, we can first take the dot product of the arguments, which prevents F from mixing the vector dimensions in undesirable ways.

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}, \qquad (3)$$

GloVe cost function

- For word-word co-occurrence matrices, the distinction between a word and a context word is arbitrary and that we are free to exchange the two roles.
- The symmetry can be restored in two steps.
- First, we require that F be a homomorphism between the groups $(\mathbb{R},+)$ and $(\mathbb{R}>0, \times)$, i.e.,

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}, \quad (4)$$

GloVe cost function

which, by Eqn. (3), is solved by

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i} . \qquad (5)$$

The solution to Eqn. (4) is $F = \exp$

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) . \qquad (6)$$

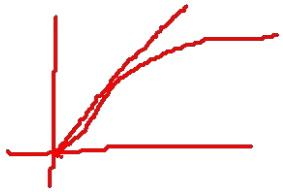
- Next, we note that Eqn. (6) would exhibit the exchange symmetry if not for the $\log(Xi)$ on the right-hand side.
- However, this term is independent of k so it can be absorbed into a bias bi for wi .
- Finally, adding an additional bias $b'i$ for $w'i$ restores the symmetry.

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) . \qquad (7)$$

GloVe cost function : weighting function

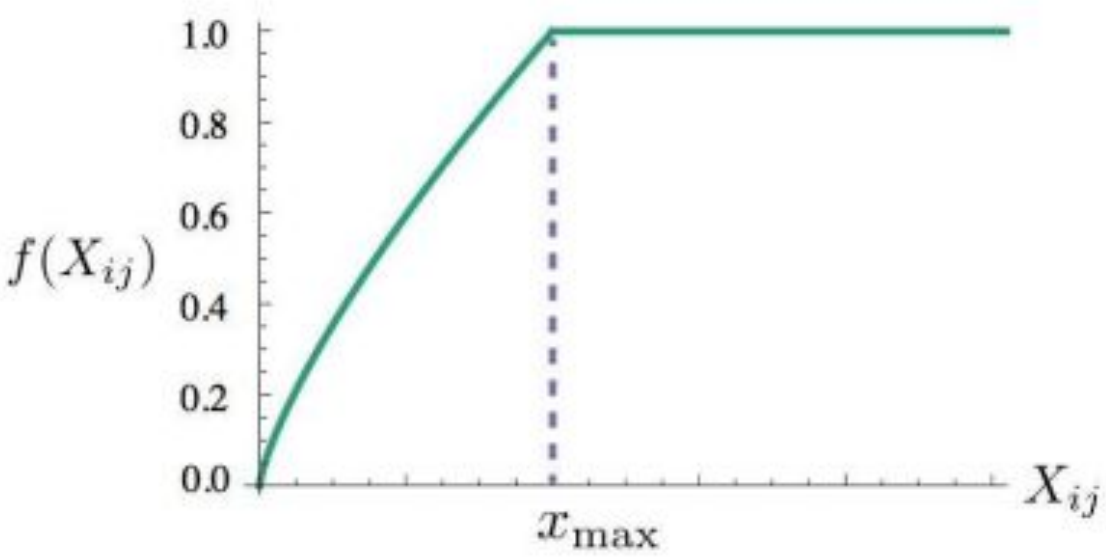
$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 , \quad (8)$$

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x) \log^2 x$ is finite.
2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.
3. $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.



GloVe cost function : weighting function

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} . \qquad (9)$$



Weighting function f with $\alpha = 3/4$.

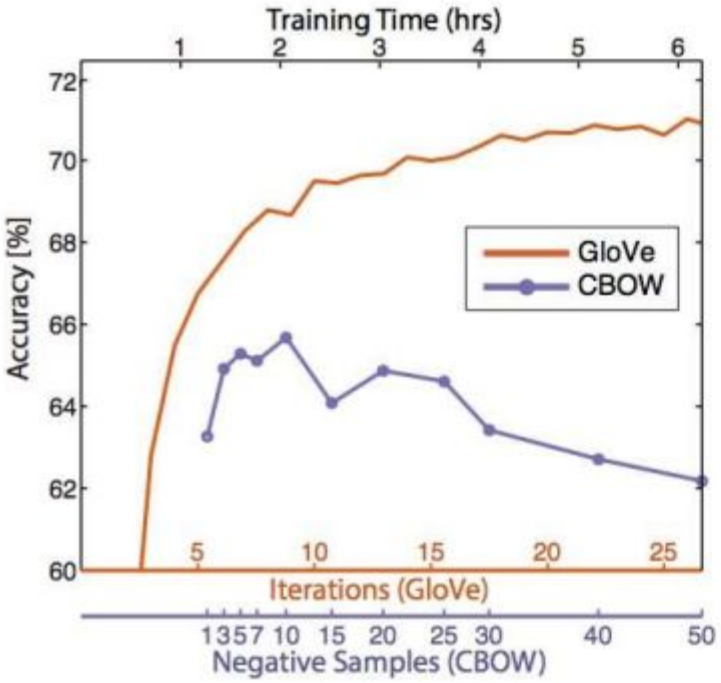
Evaluation of Glove embeddings

- Word analogy
- Word Similarity
- Named Entity Recognition(NER)

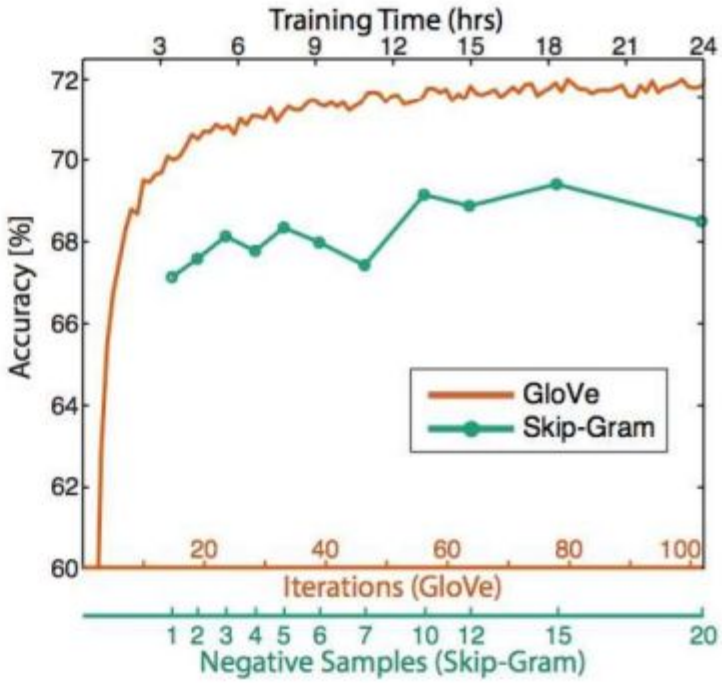
Word analogy

- The word analogy task consists of questions like, “a is to b as c is to ___?”
- The semantic questions, like “Athens is to Greece as Berlin is to ___?”.
- The syntactic questions like, “dance is to dancing as fly is to ___?

Experiments : analogy task



(a) GloVe vs CBOW



(b) GloVe vs Skip-Gram

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to Parts of Speech Tagging

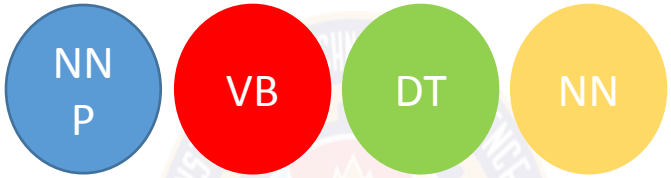
Prof.Aruna Malapati

Learning objectives

- Define the problem of Parts Of Speech tagging (POS)
- Challenges while POS tagging
- Hidden Markov Model

Parts Of Speech Tagging (POS)

Aruna saw the saw.



- Annotate each word in a sentence with a part of speech.

Sequence Data

- Till now, we assumed that the data instances are classified independently.
  - More precisely, we assumed that the data is iid (identically and independently distributed)
- In many applications, the data arrives sequentially and the classes are correlated – E.g., weather prediction, speech

What is the challenge in PoS Tagging?

- Tag ambiguous words
  - Solve the lexical ambiguities
  - The/DT wind/NN was/VB too/ADV strong/ADJ to/PRP wind/VB the/DT sail/NN.
- Tag unknown words
  - The/DT rural/JJ Babbitt/??? who/WP bloviates/??? about/IN progress/NN and/CC growth/NN

Category of classes

- Open: vast number of new members
  - Nouns, Verbs, Adjectives, Adverbs
- Closed: small set of words
  - Determiners: a, an, the
  - Pronouns: she, he, i, you, we
  - Prepositions: on, under, over, near, by,…

Choosing a tagset

- Need to choose a standard set of tags to do POS tagging.
- Could pick very coarse tagset – N, V, Adj, Adv, Prep.
- More commonly used set is finer-grained.
  - Penn TreeBank II tagset has 36 word tags
  - PRP, PRP$, VBG,VBD, JJR, JJS …

Penn TreeBank PoS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

POS Tagging performance evaluation

- Percentage of tags predicted correctly.
- Baseline approach: Tag every word with its most common tag and rest of the words are tagged as Noun.

Applications of POS tagging

- Text to speech conversion
- Useful as a preprocessing step of parsing

Two Methods for PoS Tagging

- Rule-based systems
- Statistical sequence models
  - Hidden Markov Models

Discriminative classification models

- Given a set of training examples $<x_i, y_i>$ for i=1..m where each x_i is the input vector and y_i is the class label.
- The modeling task as to learn a function f mapping the input x to labels f(x).
- The classification models you have learnt till now are discriminative models where each sample input was considered independent of each other.

Generative model for sequence labelling

- Given a set of training examples $<x_i, y_i>$ for i=1..m where each x_i is the input vector and y_i is the class label.
- In the POS tagging problem we have

  x_1= the dog barks, y_1=DT NN VB

  x_2= the boy smiles, y_2=DT NN VB

  .....
- The task is to learn a function f that maps

Generative model for sequence labelling

Using Bayes rule

$$P(t_1, \ldots, t_n \mid w_1, \ldots, w_n) = \frac{P(t_1, \ldots, t_n) P(w_1, \ldots, w_n \mid t_1, \ldots, t_n)}{P(w_1, \ldots, w_n)}$$

Submodels:

1. Prior: $P(t_1,...,t_n)$
2. Likelihood: $P(w_1,...,w_n \mid t_1,...,t_n)$
3. Marginal: $P(w_1,...,w_n)$ – can be ignored in argmax search

Markov Assumption

- Context model (prior)

$$P(t_1, \ldots, t_n) = \prod_{i=1}^{n} P(t_i | t_{i-k} \ldots, t_{i-1})$$

- Lexical model (likelihood)

$$P(w_1, \ldots, w_n \mid t_1, \ldots, t_n) = \prod_{i=1}^{n} P(w_i | t_i)$$

Model Parameters

- Contextual probabilities : $P(t_i|t_{i-k},...,t_{i-1})$
- Lexical probabilities : $P(w_i|t_i)$
- We can estimate these probabilities from a tagged corpus:

$$\hat{P}_{\text{MLE}}(w_i|t_i) = \frac{c(w_i, t_i)}{c(t_i)} \qquad \hat{P}_{\text{MLE}}(t_i|t_{i-k}, \ldots, t_{i-1}) = \frac{c(t_{i-k}, \ldots, t_{i-1}, t_i)}{c(t_{i-k}, \ldots, t_{i-1})}$$

Computing Probabilities

- The probability of a tagging:

$$P(t_1, \ldots, t_n, w_1, \ldots, w_n) = \prod_{i=1}^{n} P(t_i | t_{i-k}, \ldots, t_{i-1}) P(w_i | t_i)$$

- Finding the most probable tagging:

$$\underset{t_1, \ldots, t_n}{\text{argmax}} \prod_{i=1}^{n} P(t_i | t_{i-k}, \ldots, t_{i-1}) P(w_i | t_i)$$

Two fundamental problems in HMM

- Decoding:
  - How do we compute the best tag sequence given parameters?
- Learning:
  - How do we estimate the parameters?

Example

- Given a sentence of length 3, * * the dog barks STOP and the tag sequence * * DT NN VB * then

- P(w1 w2 w3,y1,y2,y3) = T(DT|*,*) X T(NN|*,DT) X T(VB|DT NN) X T(STOP|NN VB) X E(*|*) X E(*|*) E(the|DT) X E(dog|NN) X E(barks|VB)X E(STOP|*)

- We can also define y_{-1}=* and y_0=* as special symbols.

Markov Chains

- A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set.
- A Markov Model is a finite state machine with probabilistic state transitions.
- Markov assumption that next state only depends on the current state and independent of previous history.

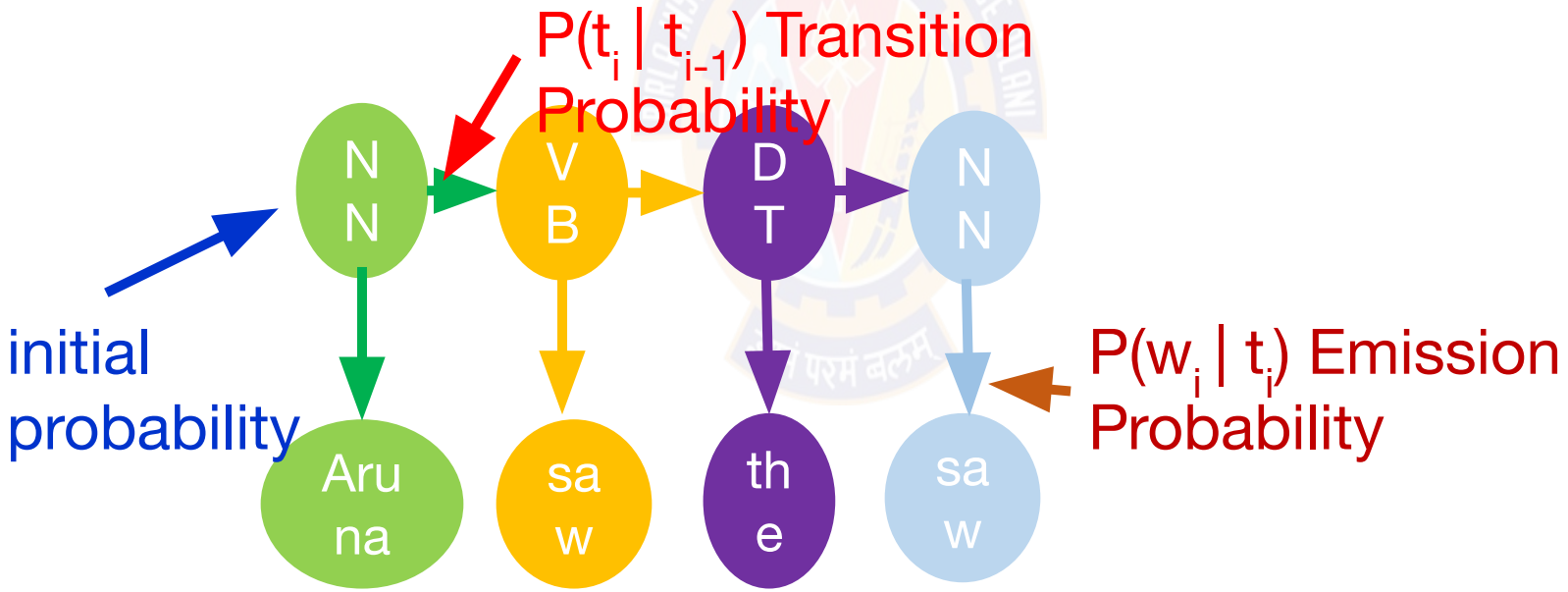
Markov Chain

- Formally, a Markov chain is specified by the following components:

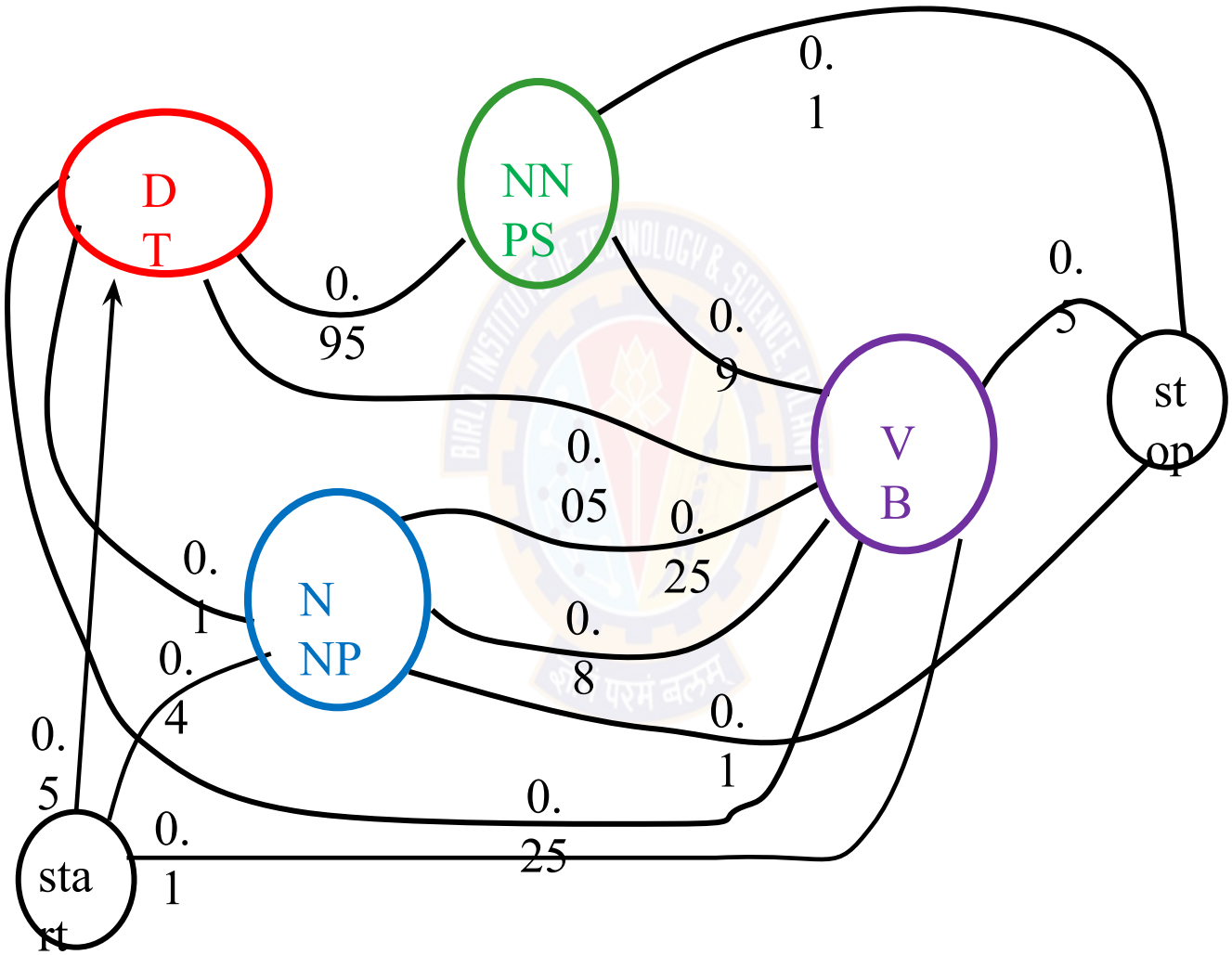
$Q = q_1 q_2 \ldots q_N$ a set of N states	A set of N states
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	A transition probability matrix A, each a_{ij} representing the probability of moving from state i to state j,
$\pi\pi = \pi_1, \pi_2, \pi_3, \ldots, \pi\pi_n$	An initial probability distribution over states. pi is the probability that the Markov chain will start in state i. Some states j may have pi =0,

The Hidden Markov Model

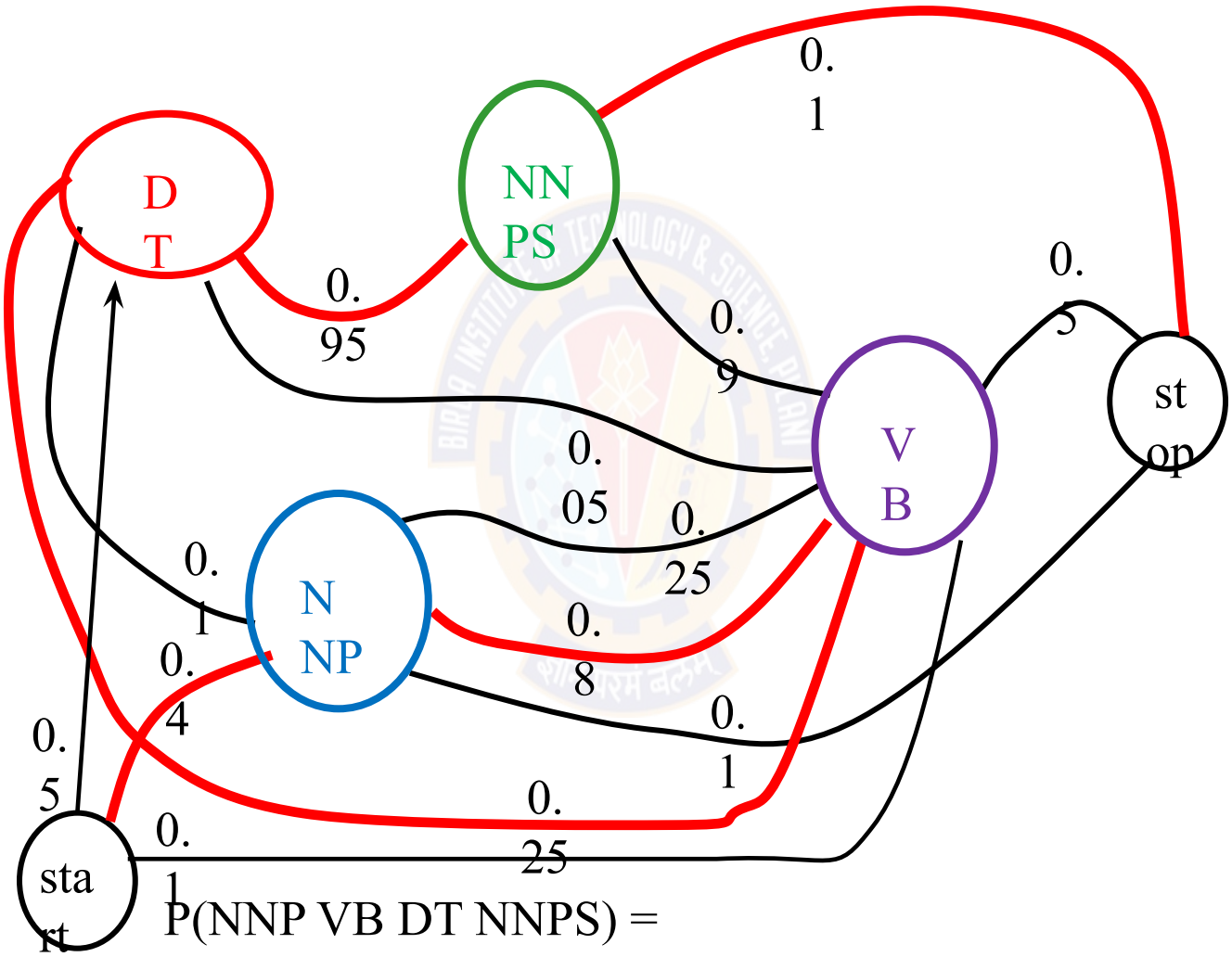
- In many cases the events we are interested in are hidden: we don't observe them directly.



HMMs as probabilistic FSA

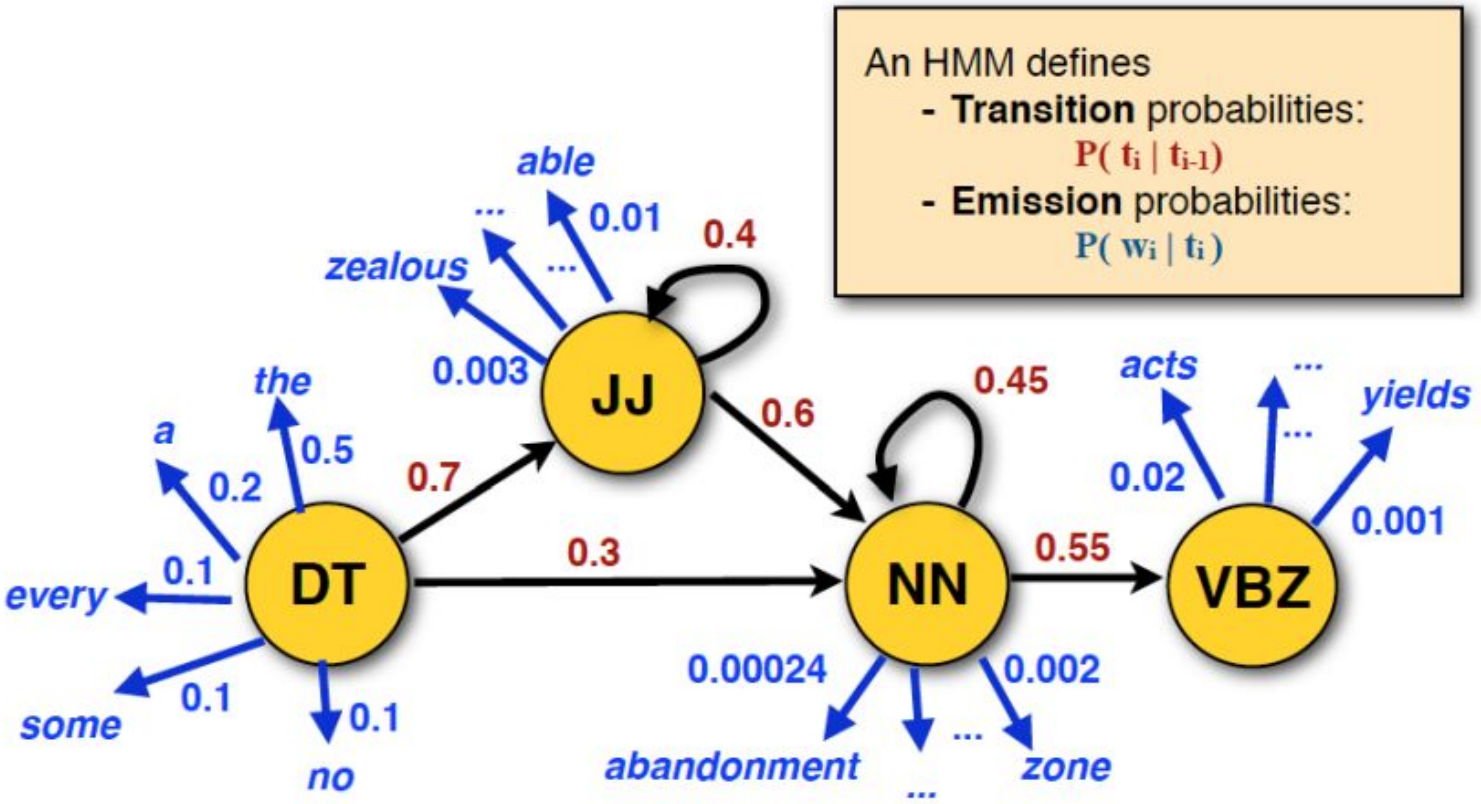


HMMs as probabilistic FSA



P(NNP VB DT NNPS) =
0.4*0.8*0.25*0.95*0.1=0.0076

HMMs as probabilistic FSA



Julia Hockenmaier:
Intro to NLP

Hidden Markov Models (formal)

- States T = $t_1, t_2 \ldots t_N$;
- Observations W= $w_1, w_2 \ldots w_N$;
  - Each observation is a symbol from a vocabulary V = $\{v_1, v_2, \ldots v_V\}$
- Transition probabilities
  - Transition probability matrix A = $\{a_{ij}\}$

$$a_{ij} = P(t_i = j \mid t_{i-1} = i) \quad 1 \leq i, j \leq N$$

- Observation likelihoods
  - Output probability matrix B=$\{b_i(k)\}$

$$b_i(k) = P(w_i = v_k \mid t_i = i)$$

- Special initial probability vector $\pi_i = P(t_1 = i) \quad 1 \leq i \leq N$

HMM tagging as decoding

- HMM model contains hidden variables, the task of determining the hidden variables sequence corresponding to the sequence of observations decoding is called decoding.
- Find our best estimate of the sequence that maximizes F $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$

HMM tagging as decoding (Contd…)

Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$P(t_1^n \mid w_1^n) = \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

Drop denominator since is the same for all tags we consider

HMM tagging as decoding (Contd…)

- A1:P(w) depends only on its own POS

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

- A2:P(t) depends only on P(t-1)

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

HMM tagging as decoding (Contd…)

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

Now we have two probabilities to calculate:

- Probability of a word occurring given its tag
- Probability of a tag occurring given a previous tag
- We can calculate each of these from a

Example

Emission

	*	DT	NNS	VB	NN	IN	STOP
*	1						
the		3/4					
employees			3/4				
pass				2/4			
an		1/4					
exam					1		
wait				1/4			
for						1	
employers			1/4				

Tag Translation Matrix

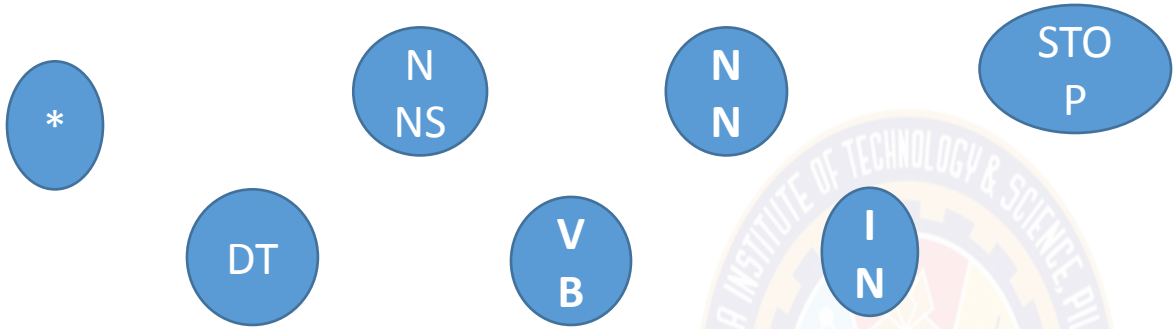
FIRST TAG / SECOND TAG

	*	DT	NNS	VB	NN	IN	STOP
*		2/3	1/3				
DT			2/4	1/4	1/4		
NNS				3/4			1/4
VB		1/4	1/4			1/4	1/4
NN							1
IN		1					
STOP							

S2: the employees wait for the pass .

T2: DT NNS VB IN DT

Transition diagram



Tag Translation Matrix

FIRST TAG	*	DT	NNS	VB	NN	IN	STOP
*		2/3	1/3				
DT			2/4	1/4	1/4		
NNS				3/4			1/4
VB		1/4	1/4			1/4	1/4

The intuition behind Viterbi

- Finding the most probable tagging sequence for a test sentence <w_1 w_2 w_3..,w_n>
- The $\operatorname{argmax}_{t_1,\ldots,t_n} \prod_{i=1}^{n} P(t_i|t_{i-k},\ldots,t_{i-1})P(w_i|t_i)$. $y_1,y_2,\ldots,y_n$ such that y_i Є S for i= 1,2,…n and y_{n+1}=STOP
- Lets assume that out tagging model is a bigram model

Brute force search over tag sequences

- Input sentence <the employees wait for an exam>
- S ={ DT NNS VB IN NN}
- All possible tag sequences

DT DT DT DT DT DT STOP

DT DT DT DT DT NNS STOP

DT DT DT DT DT VB STOP

….

Viterbi algorithm

- Create a table V with N+2 rows and T columns:
  - N – the number of states/tags
  - T – the length of the sequence/sentence
- Initialise the first column
- For each tag t in the tagset compute:

$$V[t,1] = P(t|start)P(w_1|t)$$

- For each column j = 2 to T in the table V:
  - For $V[t,j] = \max_{t'} V[t',j-1]P(t|t')P(w_j|t)$ pute:

Example

Transition matrix:

$P(t_i \mid t_{i-1})$	NOUN	Verb	Det	Prep	ADV	STOP
<S>	.3	.1	.3	.2	.1	0
Noun	.2	.4	.01	.3	.04	.05
Verb	.3	.05	.3	.2	.1	.05
Det	.9	.01	.01	.01	.07	0
Prep	.4	.05	.4	.1	.05	0

Emission matrix:

$P(w_i \mid t_i)$	a	cat	doctor	in	is	the	very
Noun	0	.5	.4	0	0.1	0	0
Verb	0	0	.1	0	.9	0	0
Det	.3	0	0	0	0	.7	0
Prep	0	0	0	1.0	0	0	0
Adv	0	0	0	.1	0	0	.9

$$V[t,1] = P(t|start)P(w_1|t)$$

	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0				
Verb	0				
Det	.21				
Prep	0				
Adv	0				

V(Noun,the) = P(Noun|<S>)P(the|Noun) = .3 X 0=0

V(Verb,the) = P(Verb|<S>)P(the|Verb) = .1 X 0=0

V(Det,the) = P(Det|<S>)P(the|Det) = .3 X .7=.21

V(Prep,the) = P(Prep|<S>)P(the|Prep)

Example (Contd..)

$V(\text{Noun, doctor}) = \max_{t'} V(t',\text{the}) \times P(\text{Noun}|t') \times P(\text{doctor}|\text{Noun})$

$= \max\{0, 0, .21\ (.9 \times .4), 0, 0\} = .0756$

$V(\text{Verb, doctor}) = \max_{t'} V(t',\text{the}) \times P(\text{Verb}|t') \times P(\text{doctor}|\text{Verb})$

$= \max\{0, 0, .21(.01 \times .1)), 0, 0\}$

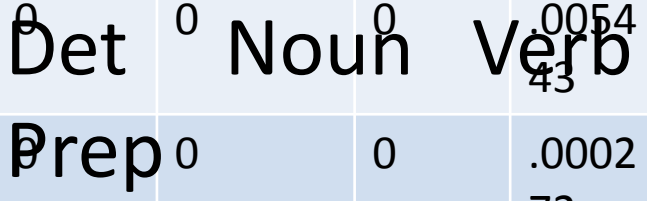
	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756			
Verb	0	.00021			
Det	.21	0			
Prep	0	0			
Adv	0	0			

Completed Viterbi matrix

	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756	.001512	0	
Verb	0	.00021	.027216	0	.0000272
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

Backtracking the Viterbi Matrix

	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756	.001512	0	
Verb	0	.00021	.027216	0	.0000272
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	



Thank You!

BITS Pilani

BITS Pilani
Hyderabad Campus

Prof.Aruna Malapati
Department of CSIS

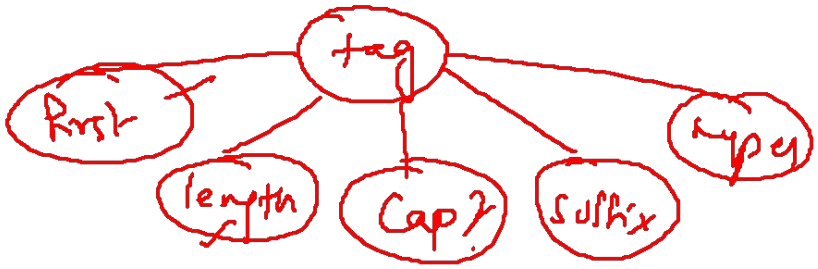
Maximum Entropy Markov Model for POS Tagging

Discriminative Models

$$\underset{T}{\operatorname{argmax}}\, P(T|W) \approx \underset{T}{\operatorname{argmax}}\, \bar{P}(T,W)$$

$P(T|W)$

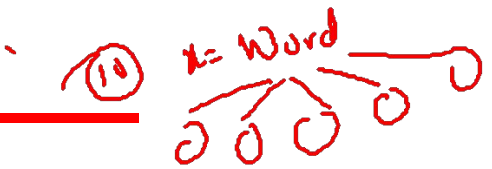
word



Maximum Entropy Model (MEM)

- MEM/MaxEnt is another name for multinomial logistic regression that belong to the family of classifiers known as the exponential or log-linear classifiers.
  - Extract some set of features from the input.
  - Combine them linearly(i.e multiply the weight and ass them up)
  - Use this sum as an exponent
- Given the features and weights, out goal is to choose a class(For example a part of speech) for the word.

Maximum Entropy Model (MEM)



- MaxEnt does this by choosing the most probable tag: the probability of a particular class (tag) c given the observation x described using a set of features f.

$$P(c|\vec{f}) = \frac{1}{Z}\exp(\sum_i w_i f_i)$$

where Z normalizes, w_i is a weight and f_i is a numerically valued feature.

- actually w and f depend on class
- discriminative rather than generative

Discriminative classifiers

In order to understand MaxEnt we need a few basics of

- Linear Regression
- Logistic Regression
- MaxEnt(Multinomial Regression)
- MaxEnt for sequence classification

Discriminative Models

Linear Regression

$y = w_0 + w_1 x$

slope

Real value

intercept

no of words

Cost

15

2

Multiple Linear regression

$$y = w_0 + \sum_{i=1}^{N} w_i \times f_i$$

Where w are weights and f are features.
Rewritten using an intercept feature, f_0, with value 1:

$$y = \sum_{i=0}^{N} w_i \times f_i$$

Weights chosen to minimize sum of squares of differences between prediction and observation.

Multiple Linear Regression

Abstractly we want (where f is the feature vector associated with observation x):

$$P(y = true|x) = \sum_{i=0}^{N} w_i \times f_i = \vec{w} \cdot \vec{f}$$

but what we're predicting won't be a probability.
Instead, we predict the log of the odds (logit function).

$$ln\left(\frac{P(y = true|x)}{1 - P(y = true|x)}\right) = \vec{w} \cdot \vec{f}$$

Logistic Regression

Abstractly we want (where f is the feature vector associated with observation x):

$$P(y = true|x) = \sum_{i=0}^{N} w_i \times f_i = \vec{w} \cdot \vec{f}$$

but what we're predicting won't be a probability.
Instead, we predict the log of the odds (logit function).

$$ln\left(\frac{P(y = true|x)}{1 - P(y = true|x)}\right) = \vec{w} \cdot \vec{f}$$

$$\frac{P(y = true|x)}{1 - P(y = true|x)} = e^{w \cdot f}$$

Logistic regression (contd…)

Classify observation as 'true' if:

$$P(y = true|x) > P(y = false|x)$$

That is:

$$\frac{P(y = true|x)}{1 - P(y = true|x)} > 1$$

or:

$$\vec{w} \cdot \vec{f} > 0$$

So logistic regression involves learning a hyperplane with true above and false below.

Logistic regression (contd…)

MaxEnt: Multinomial logistic regression

- Logistic regression with more than two classes is Multinomial logistic regression.
- Also known as Maximum Entropy(MaxEnt) classifier

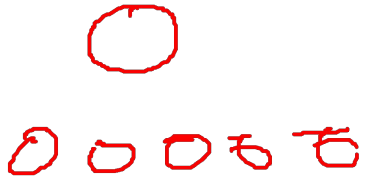
$$P(c|x) = \frac{\exp\left(\sum_{i=0}^{N} w_{ci} f_i\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^{N} w_{c'i} f_i\right)}$$

with numerical-valued features

- The denominator is the normalization factor used to make the score into a proper probability distribution

$$P(c|x) = \frac{1}{Z} \exp\left(\sum_{i=0}^{N} w_{ci} f_i\right)$$

MaxEnt: Multinomial logistic regression



with booean-valued features:

$$P(c|x) = \frac{\exp\left(\sum_{i=0}^{N} w_{ci} f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^{N} w_{c'i} f_i(c',x)\right)}$$

Features include the class:

$$f_1(c,x) = 1 \text{ if } word_i \text{ ends in "ic" \& } c = \text{CJ}$$
$$= 0 \text{ otherwise}$$

Example of non-sequential classification

Secretariat/NNP is/BEZ expected/VBN to/TO race/?? tomorrow/RB

$$f_1(c,x) = \left\{ \begin{array}{l} 1 \text{ si } w_i = \text{"race"} \ \& \ c = \text{NN} \\ 0 \text{ otherwise} \end{array} \right\}$$

$$f_2(c,x) = \left\{ \begin{array}{l} 1 \text{ si } t_{i-1} = \text{TO} \ \& \ c = \text{VB} \\ 0 \text{ otherwise} \end{array} \right\}$$

$$f_3(c,x) = \left\{ \begin{array}{l} 1 \text{ si suffix}(w_i) = \text{"ing"} \ \& \ c = \text{VBG} \\ 0 \text{ otherwise} \end{array} \right\}$$

$$f_4(c,x) = \left\{ \begin{array}{l} 1 \text{ si is\_lower\_case}(w_i) \ \& \ c = \text{VB} \\ 0 \text{ otherwise} \end{array} \right\}$$

$$f_5(c,x) = \left\{ \begin{array}{l} 1 \text{ si } w_i = \text{"race"} \ \& \ c = \text{VB} \\ 0 \text{ otherwise} \end{array} \right\}$$

$$f_6(c,x) = \left\{ \begin{array}{l} 1 \text{ si } t_{i-1} = \text{TO} \ \& \ c = \text{NN} \\ 0 \text{ otherwise} \end{array} \right\}$$

Example of non-sequential classification

Given the current input x ="race" and supposing the following weights:

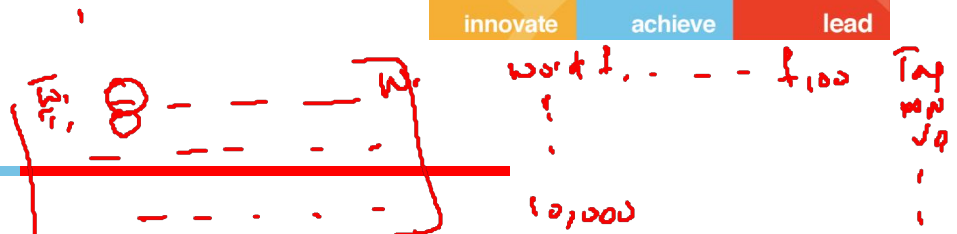
		f1	f2	f3	f4	f5	f6
VB	f	0	1	0	1	1	0
VB	w		.8		.01	.1	
NN	f	1	0	0	0	0	1
NN	w	.8					-1.3

$$P(NN \mid x) = \frac{e^{0.8}e^{-1.3}}{e^{0.8}e^{-1.3} + e^{0.8}e^{0.01}e^{0.1}} = 0.20$$

$$P(VB \mid x) = \frac{e^{0.8}e^{0.01}e^{0.1}}{e^{0.8}e^{-1.3} + e^{0.8}e^{0.01}e^{0.1}} = 0.80$$

When using MaxEnt for classification we obtain a probability distribution on the classes

Complex Features



- Consider proper nouns: a word is likely to be a proper noun if it starts with a capital letter and is not the first word of the sentence.
- We need a conjunction of two primitive features

$$f_{123}(c, x) = \begin{cases} 1 & \text{if } word_{i-1} \neq \langle s \rangle \wedge Capitalized(w_i) \wedge c = NNP \\ 0 & \text{otherwise} \end{cases}$$

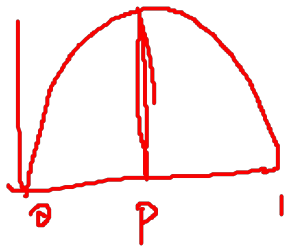
Note: We need to define such feature conjunctions explicitly as opposed to other ML (e.g. SVMs) that can handle this directly.

Why MaxEnt?

- Out of all the possible models, choose the simplest one consistent with the data (Occam's razor)
- Entropy of the distribution of discrete random variable X:

$$H(X) = -\sum_{x} P(X=x) \log_2 P(X=x)$$

- The uniform distribution has the highest entropy.



- Finding maximum entropy distribution in the set C of possible distributions

$$p^* = \underset{p \in C}{\arg\max}\ H(p)$$

MaxEnt: example

- We want to assign a tag to the word zzfish
- Initial model

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	VBG	POS	PRP	CC	CD	…
1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45	1/45

- The set of possible tags for zzfish are NN,JJ,NNS and VB(based on some training data)

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	VBG	POS	PRP	CC	CD	…
1/4	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0

MaxEnt: example

- 8 times out of 10, zzfish was tagged as some sort of common noun either NN or NNS

NN	JJ	NNS	VB	NNP	….
4/10	1/10	4/10	1/10	0	….

P(NN)+P(JJ)+P(NNS)+P(VB)=1
P(word is zzfish and t1=NN or t1=NNS) = 8/10

- In the training data for all English words (not just zzfish) verbs (VB) occur as 1 word in 20.

NN	JJ	NNS	VB
4/10	1/20	4/10	1/20

P(NN)+P(JJ)+P(NNS)+P(VB)=1
P(word is zzfish and t1=NN or t1=NNS) = 8/10
P(VB)=1/20

Maximum Entropy Markov Model

- First Solution: run a local classifier left to right, first making a hard classification of the first word in the sentence, then the second word, and so on.
- Use the output of the classifier from the previous word as a feature to predict the tag of the current word.
- The problem with hard classification on each word before moving on to the next word is that global optimal sequence cannot be achieved.
- Hence we combine Viterbi algorithm with MaxEnt Maximum Entropy Markov Model (MEMM)

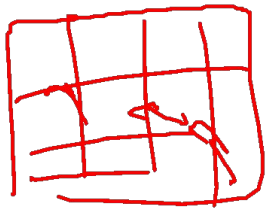
HMMs VS MEMM

HMM

$$\hat{T} = \arg\max_{T} P(T|W)$$

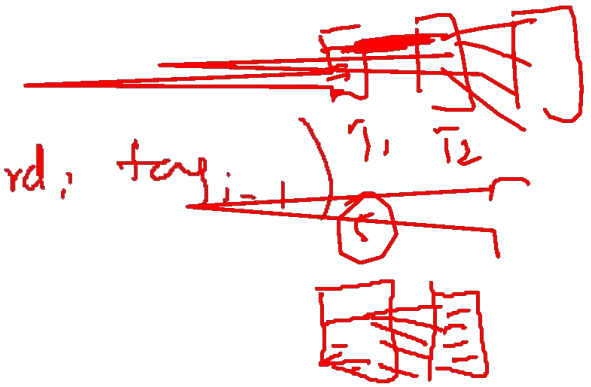
$$= \arg\max_{T} P(W|T)\,P(T)$$

$$= \arg\max_{T} \prod_{i} P(\text{Word}_i|\text{tag}_i) \prod_{i} P(\text{tag}_i|\text{tag}_{i-1})$$



$$\hat{T} = \arg\max_{T} P(T|W)$$

$$= \arg\max_{T} \prod_{i} P(\text{tag}_i|\text{word}_i, \text{tag}_{i-1})$$



Summary

- MaxEnt is a multinomial logistic regression.

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to Topic Modelling

Prof.Aruna Malapati

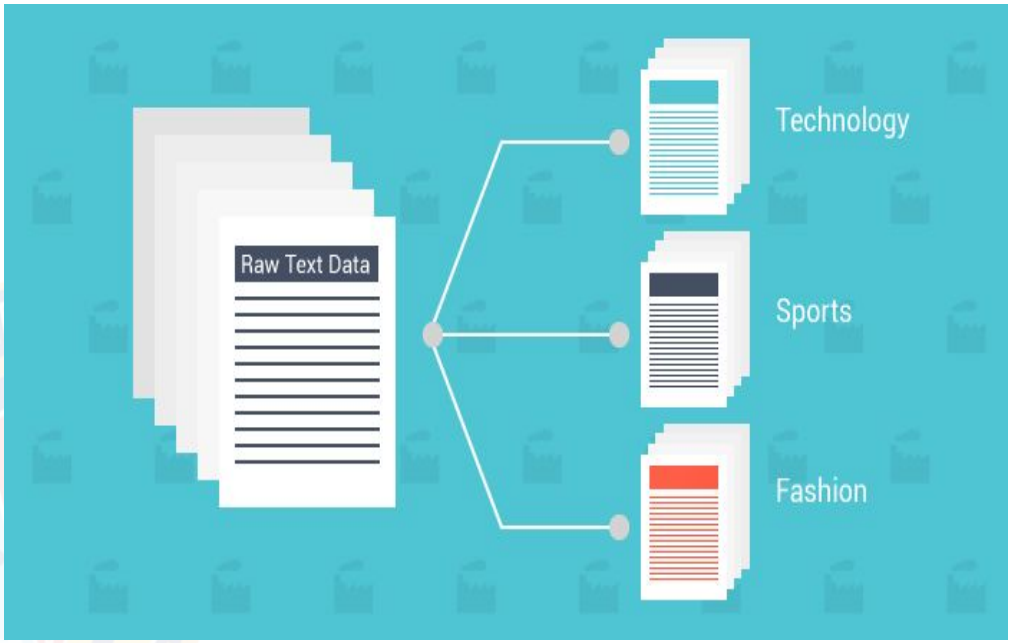
Learning Objectives

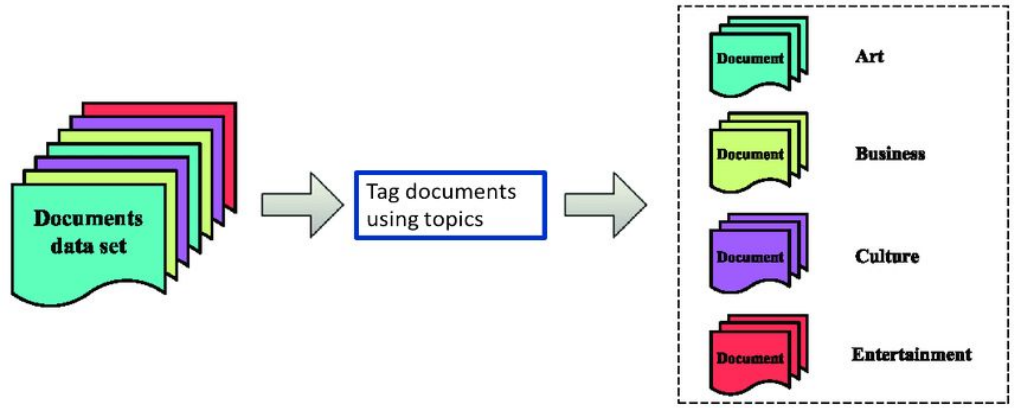
- Motivation for Topic Modelling
- Generative model for Topic Model
- The posterior distribution
- Bernoulli and Beta distributions
- Conjugate prior
- Dirichlet distribution

Motivation for Topic Modelling

Objectives of Topic Modelling

- Use these annotations to organize, summarize and search the documents.





Sample output from the LDA

- Four topics learned from the S&P 500 stock market data
- Goal is to find groups of stocks that tend to move together.

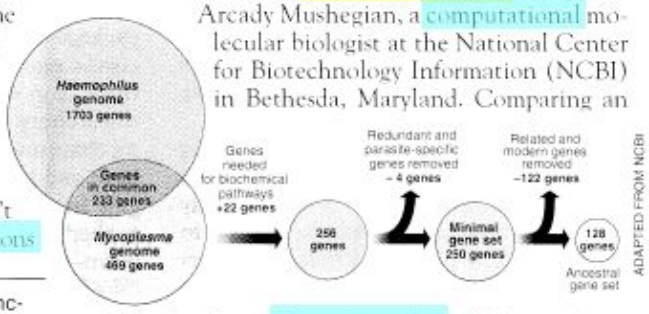
Topic 1	Topic 2	Topic 3	Topic 4
Southwestern Energy	Penneys	Capital One	Simon Property
Range Resources	Macys	BNY Mellon	Kimco Realty
Cabot Oil & Gas	Kohls	Discover	Equity Residential
EOG Resources	Nordstrom	Northern Trust	AvalonBay Communities
Chesapeake Energy	Target	Janus	Apartment Investment
Pioneer Resources	Limited	JPMorgan Chase	Vornado Realty Trust
Devon Energy	Lowes	State Street	Boston Properties
Peabody Energy	Home Depot	Wells Fargo	Public Storage
Anadarko Petroleum	American Express	PPL	Host Hotels
Massey Energy	Abercrombie	T. Rowe Price	HCP Inc.

- The topic model does not provide any label to these group of words.

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Genetics

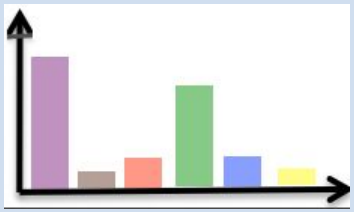
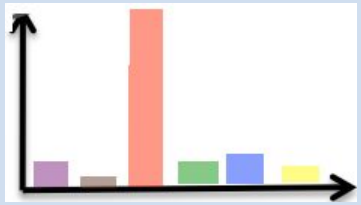
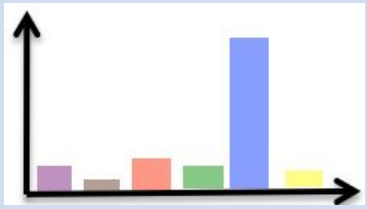
Evolutionary biology

Data Analysis

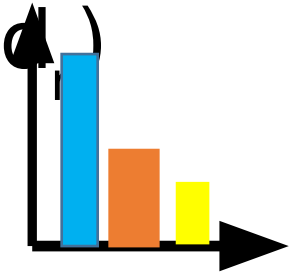
Overall schematic

Corpus

Each topic is defined as a Multinomial distribution over the vocabulary, parameterized by ϕ_k

T_1	T_2	T_3
		

Documents(d_1 ... d_n)

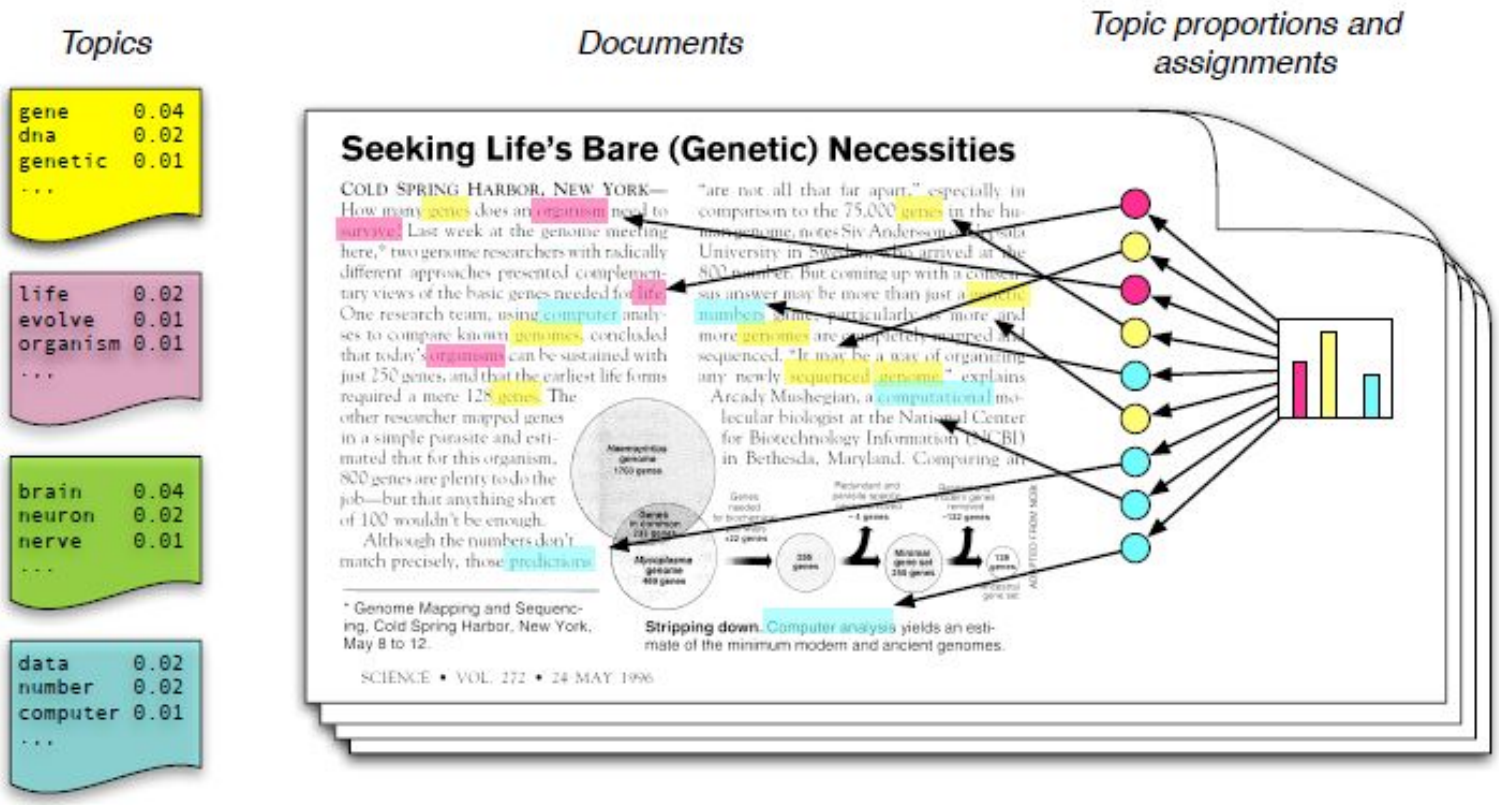


Each document has a distribution over K topics

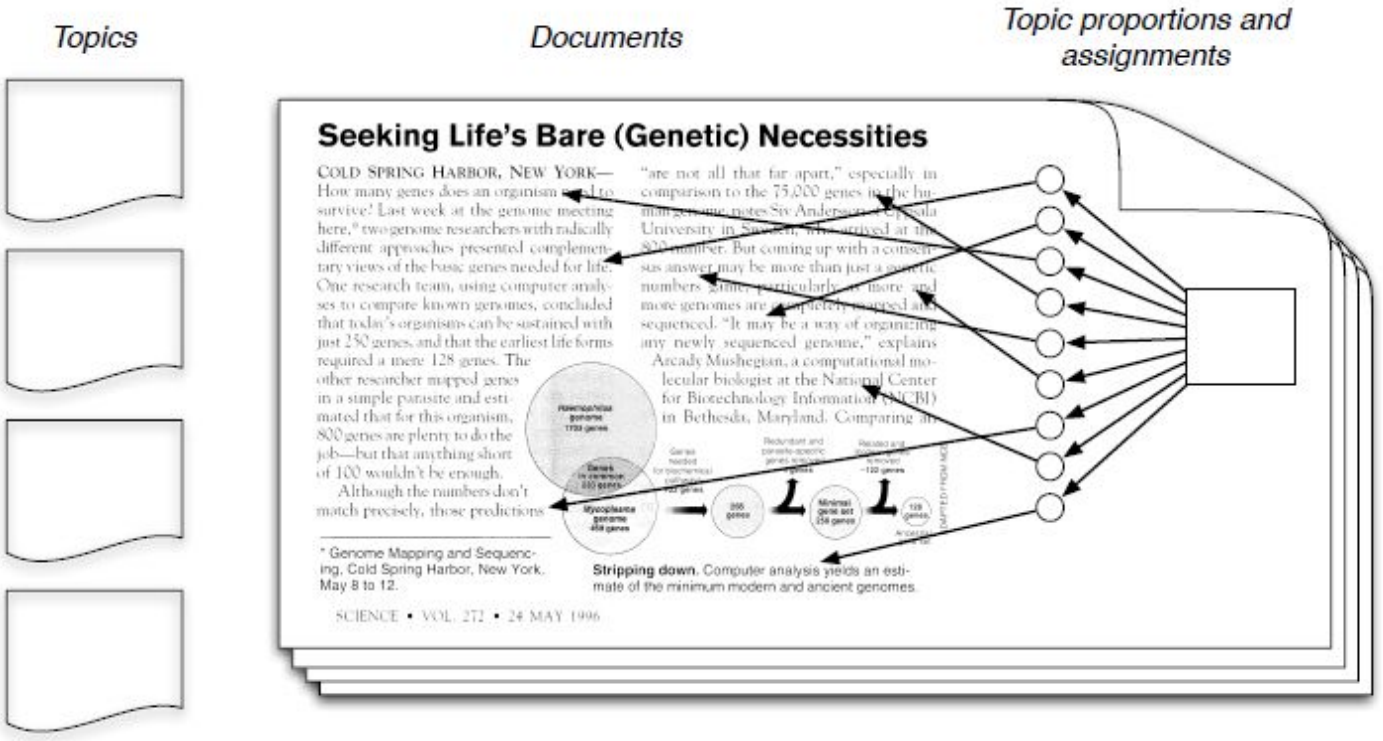
K-Topics (Hyper Parameter)

Vocabulary(W_1

Generative model for Topic Model



The posterior distribution



Bernoulli Trial

- Any single trial with two possible outcomes can be modeled as a Bernoulli trial: team wins/loses, pitch is a strike/ball, coin comes up heads or tails, etc.
- A Bernoulli trial uses Bernoulli distribution to calculate the probability of either outcome.

Bernoulli trial

Bernoulli: A Special Case of the Binomial Distribution

Binomial Trail: Chance of getting
n heads in a row(n=3)

Bernoulli Trail: Chance of getting
a heads on a single flip

Bernoulli - Distribution Notation

- The probability mass function of the Bernoulli distribution is

$$f(x)=P(X=k)=\Theta^{k}(1-\Theta)^{1-k}, \quad k=\{0,1\}$$

- The only parameter of the bernoulli distribution is θ, which defines the probability of success during a bernoulli trial.

Binomial distribution

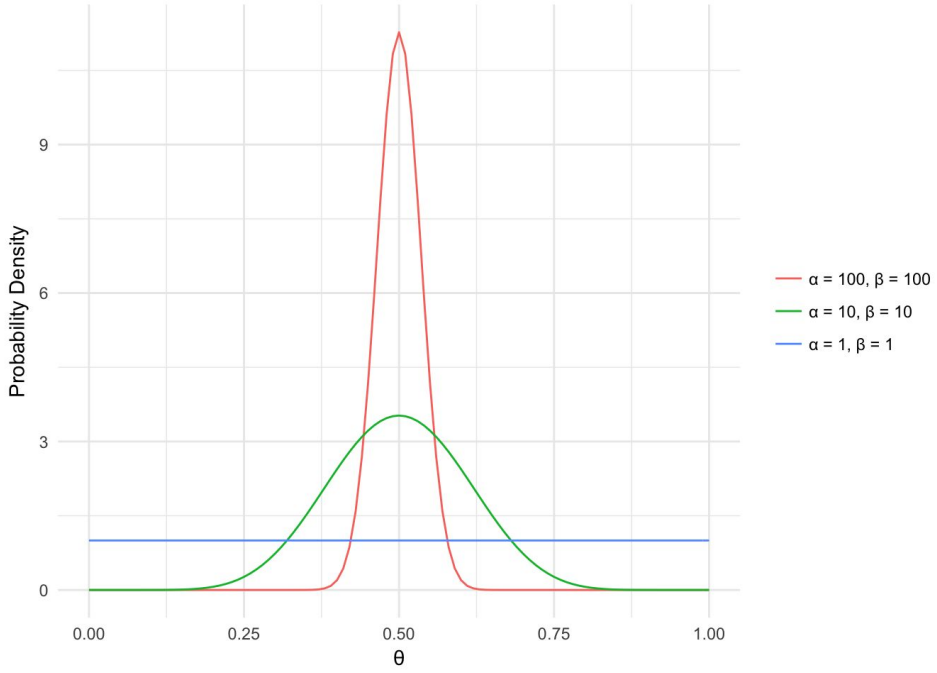
Beta Distribution

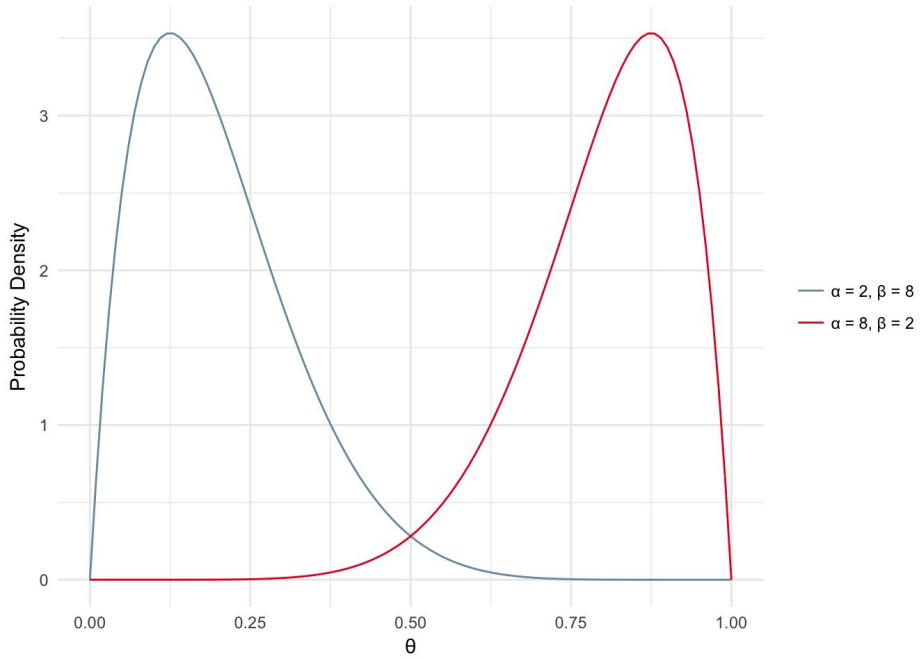
- The probability distribution function for the beta distribution

$$f(\theta;\alpha,\beta) = \frac{\theta^{(\alpha-1)}(1-\theta)^{(\beta-1)}}{B(\alpha,\beta)}$$

Beta Distribution

The beta distribution can be thought of as a probability distribution of probabilities.





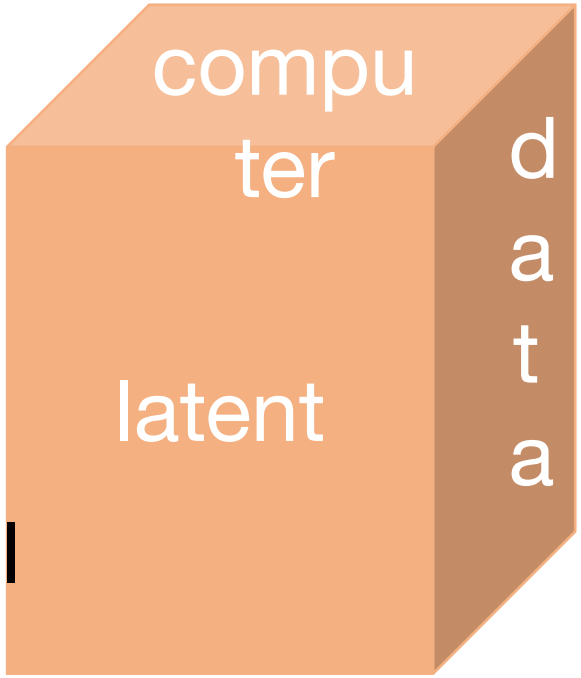
From Dice to words

- Suppose we roll our die of words having k sides(vocabulary) where each side takes probabilities $\Theta_1, \ldots, \Theta_k$ respectively.
- Probability mass function

$$f(x) = \frac{n!}{x_1!x_2!\cdots x_k!}\theta_1^{x_1}\theta_2^{x_2}\cdots\theta_k^{x_k}$$

k - number of sides on the die

n - number of times the die will be rolled



The Building Blocks of inferring the parameters

- Parameter estimation

$$\underbrace{p(\theta|D)}_{posterior} = \frac{\overbrace{p(D|\theta)}^{likelihood}\overbrace{p(\theta)}^{prior}}{\underbrace{p(D)}_{evidence}}$$

- Maximum Likelihood
- Maximum a Posterior (MAP)
- Bayesian Inference

Conjugate Prior

$$\underbrace{p(\theta|D)}_{posterior} = \frac{\overbrace{p(D|\theta)}^{likelihood}\overbrace{p(\theta)}^{prior}}{\underbrace{p(D)}_{evidence}}$$

Dirichlet distributions

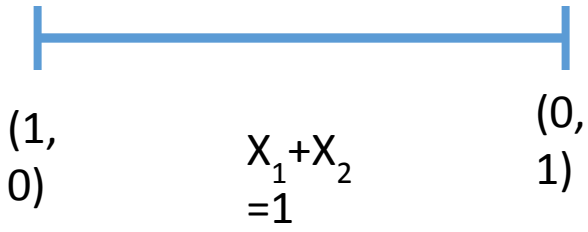
- Dirichlet distributions are probability distributions over multinomial parameter vectors
  - They are called Beta distributions when k = 2
- The Dirichlet probability density function is defined as

$$Dir(\vec{\theta}|\vec{\alpha}) = \frac{\Gamma(\sum_{i=1}^{K}\alpha_i)}{\prod_{i=1}^{K}\Gamma(\alpha_i)}\prod_{i=1}^{K}\theta_i^{\alpha_i-1}$$

$$Dir(\vec{\theta}|\vec{\alpha}) = \frac{1}{B(\alpha)}\prod_{i=1}^{K}\theta_i^{\alpha_i-1} \qquad \text{wh} \quad \frac{1}{B(\alpha)} = \frac{\Gamma(\sum_{i=1}^{K}\alpha_i)}{\prod_{i=1}^{K}\Gamma(\alpha_i)}$$

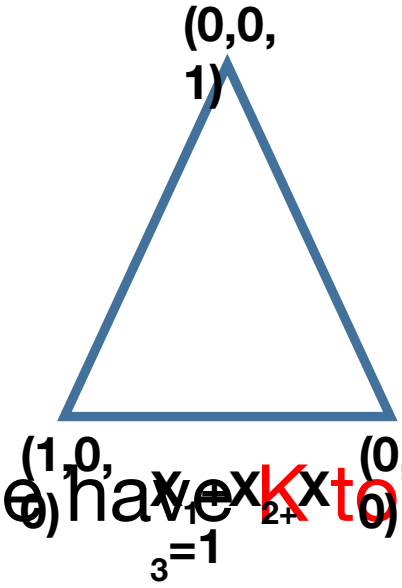
Visualization of the simplex

- This is often referred as simplex and a most convenient way to visualize this is using a certain shapes depending upon the number of topics.
- Suppose K=2 topics which can be modeled as 1-simplex and can be visualized using a line.



Visualization of the simplex(Contd…)

- Suppose K=3 topics which can be modeled as 2-simplex and can be visualized using a triangle.

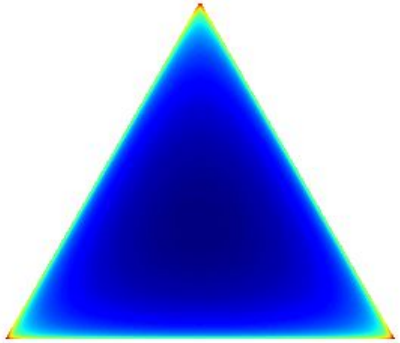


- If we have K topics this can be generated using K-1 simplex.

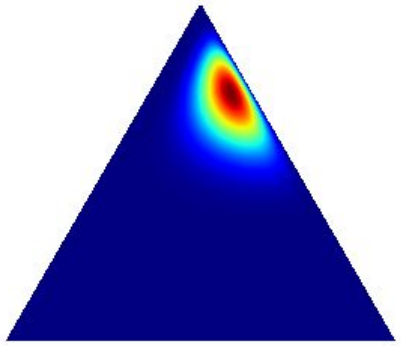
Dirichlet distribution is parametrized by α

Shape of the Dirichlet distribution

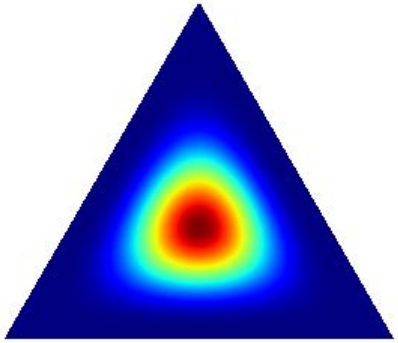
Dirichlet(0.999,



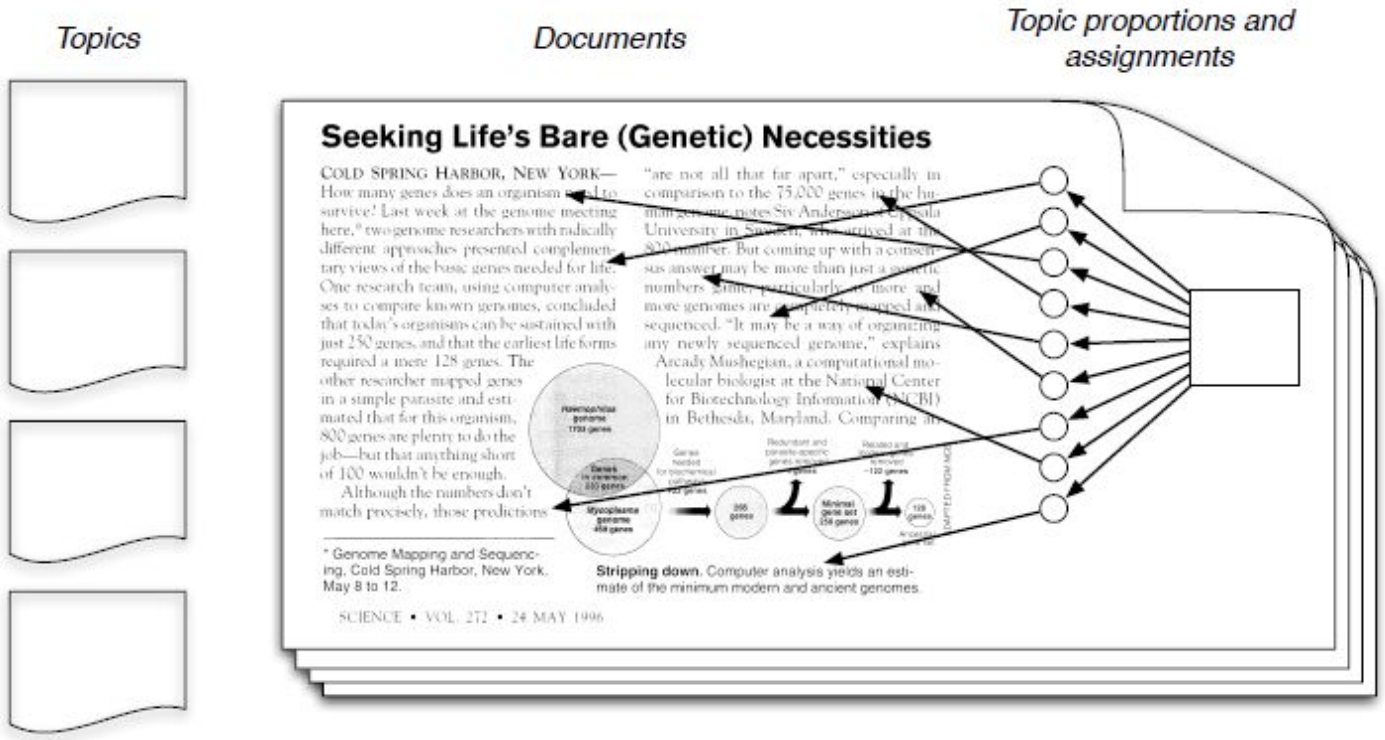
Dirichlet(2,



Dirichlet(5,

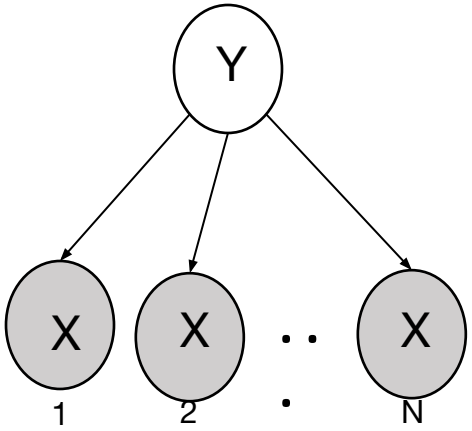


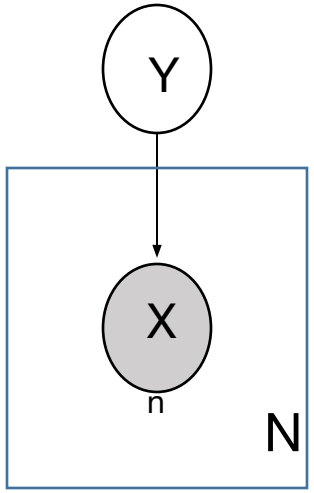
The posterior distribution



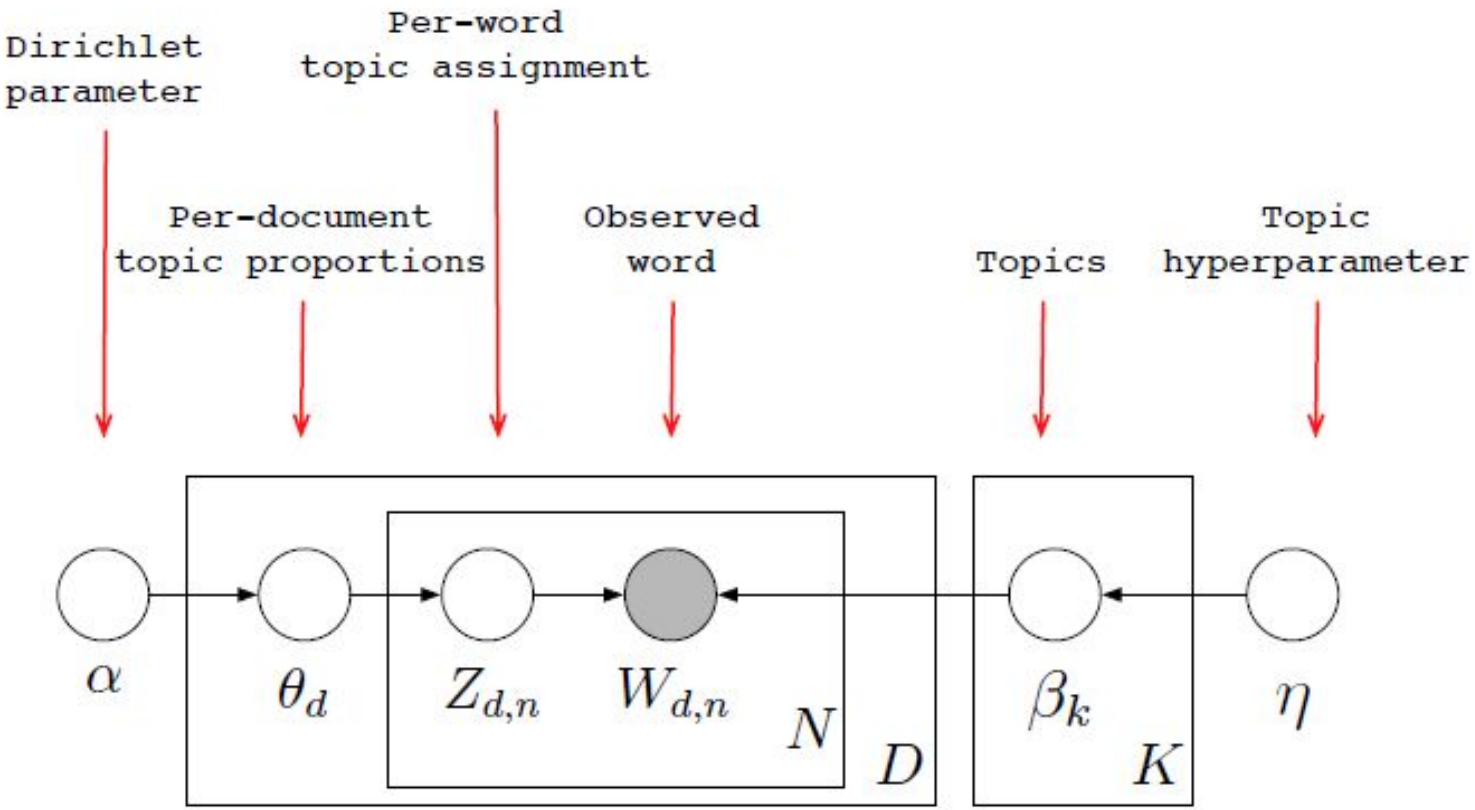
BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
ज्ञानं परमं बलम्

Directed graphical model

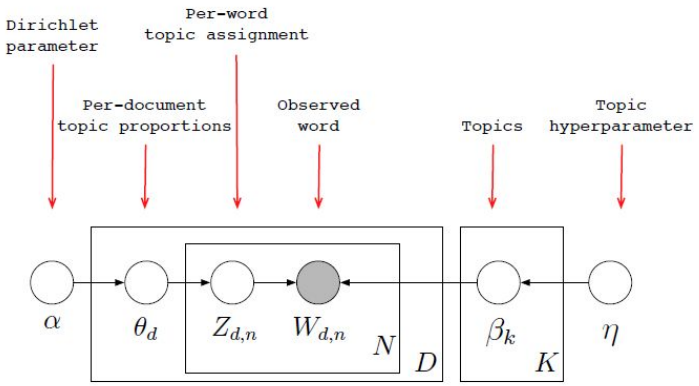




Directed graphical model of LDA



Directed graphical model of LDA (Contd..)



- Draw each topic $\beta_i \sim Dir(\eta)$, for $i \in \{1,2,\ldots K\}$
- For each document
  - Draw each topic proportions $\theta_d \sim Dir(\alpha)$
    - Draw $Z_{d,n} \sim Multinomial(\theta_d)$
    - Draw $W_{d,n} \sim Multinomial(\beta_{z_{d,n}})$

- Draw each topic $\beta_i \sim Dir(\eta)$, for $i \in \{1,2,\ldots K\}$
- For each document
  - Draw each topic proportions $\theta_d \sim Dir(\alpha)$
    - Draw $Z_{d,n} \sim Multinomial(\theta_d)$
    - Draw $W_{d,n} \sim Multinomial(\beta_{z_{d,n}})$

Gibbs Sampling

Gibbs Sampling (Contd…)

Algorithm

- Step1: Assign a random topic [1…T] for each word
- Step2: For each word token, a new topic is sampled as per $P(z_i=j|z_{-l},w_i,d_i)$ and the matrices Cwt and Cdt are updated.
- One iteration over all word token in the document is a Gibbs Sample

Estimate for θ and β

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
ज्ञानं परमं बलम्

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
ज्ञानं परमं बलम्

Thank You!

BITS Pilani

Dr.Aruna Malapati
Asst Professor
Department of CSIS

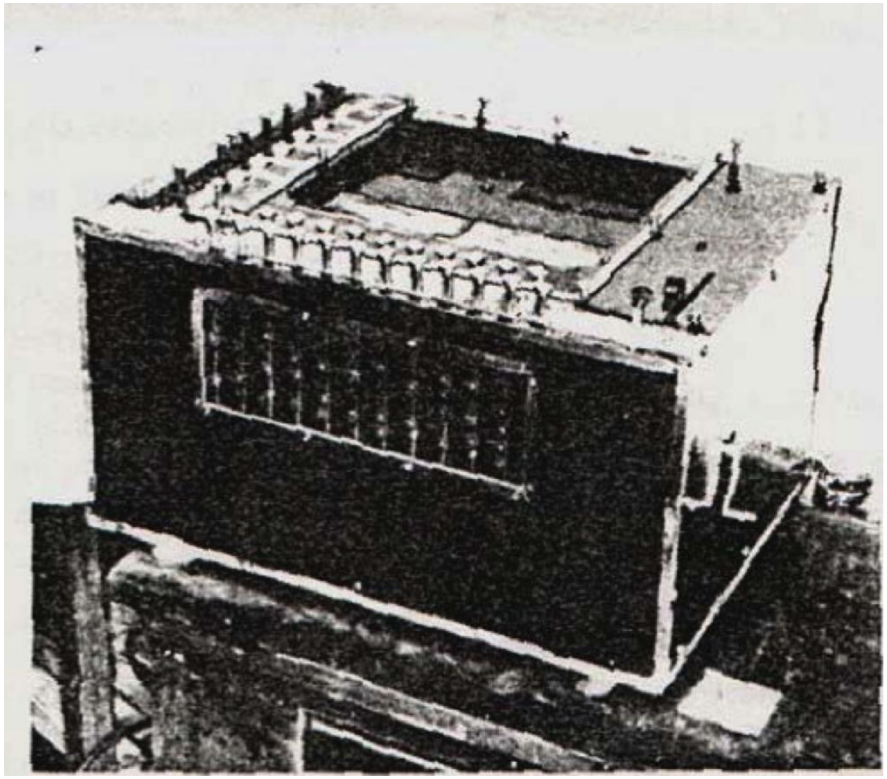
BITS Pilani
Hyderabad Campus

Statistical Machine Translation

Today's Agenda

- Cross Language IR
- Approaches to Machine Translation(MT)
- Classification of MT System
- Statistical MT – IBM model1 word based MT

History



Georges Artrouni's mechanical brain, a translation device patented in France in 1933.

COMIT	LUTE
METAL	KANT
SYSTRAN	CANDIDE
TITUS	SDL
SPANAM	EUROTRA
CUKT	UNITRAN
ARIANE	REWRITE
GETA	PHAROAH
SUSY	MOSES
AMPAR	HIERO
NERPA	SAMT
FRAP	IODHUA
LOGOA	GOOGLE TRANSLATE

Challenges in MT

1. Lexical Ambiguity

   Example

   book the flight -> reservation

   read the book -> library

   Kill the man -> murder

   kill the process

Challenges in MT

2.Differing word ordering

English word ordering: subject verb object

Most Indian languages word ordering : subject object verb

Japanese word ordering: subject object verb

3. Syntactic structure is not preserved

the bottle floated on water

बोतल पानी पर जारी

4. Syntactic ambiguity

john hit the dog with a stick

जॉन एक छड़ी के साथ कुत्ता मारा

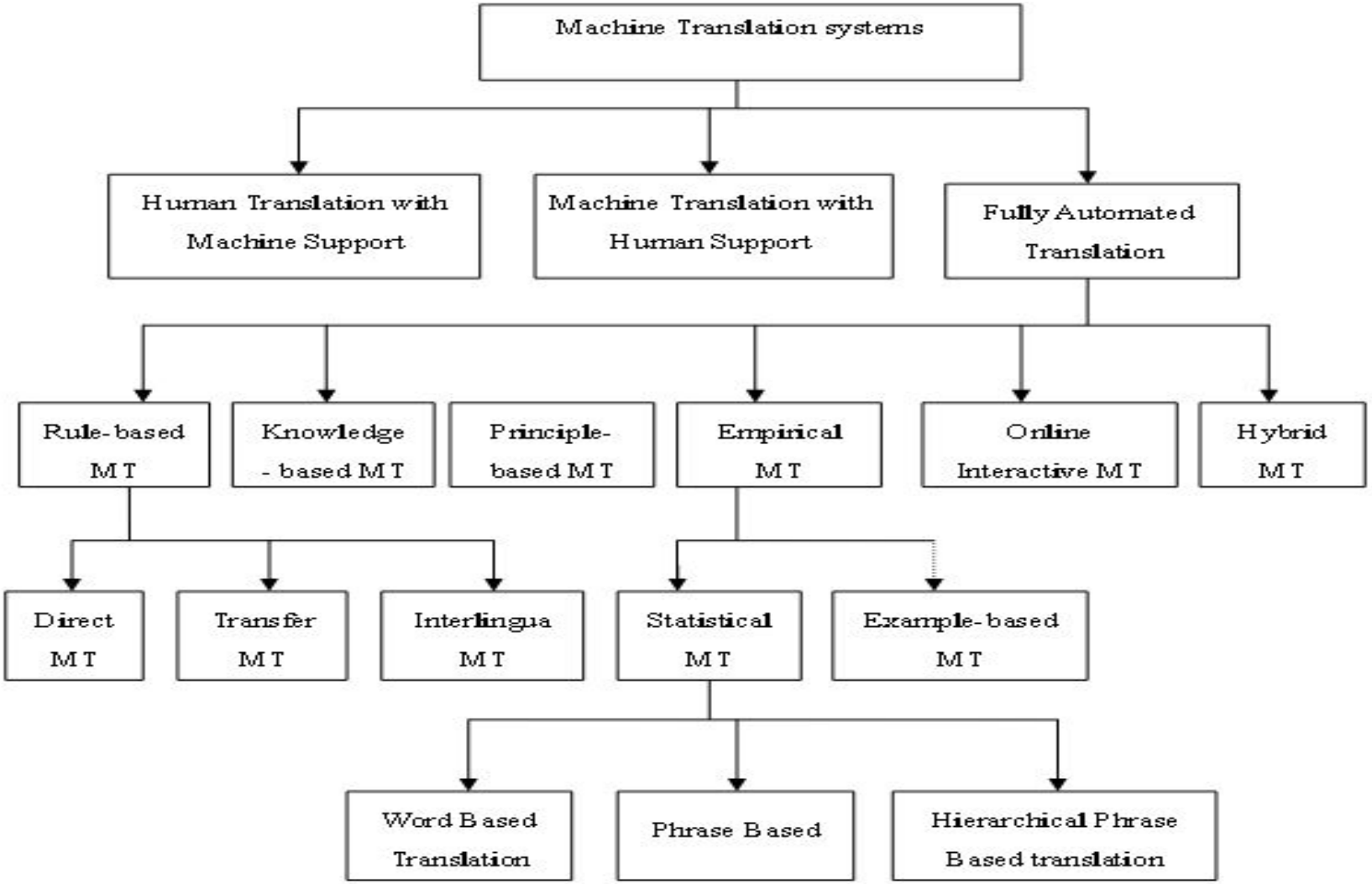
Challenges in MT

5.Pronoun resolution

The computer outputs data, it is fast

कंप्यूटर outputs डेटा, यह तेजी से है

Classification of MT System

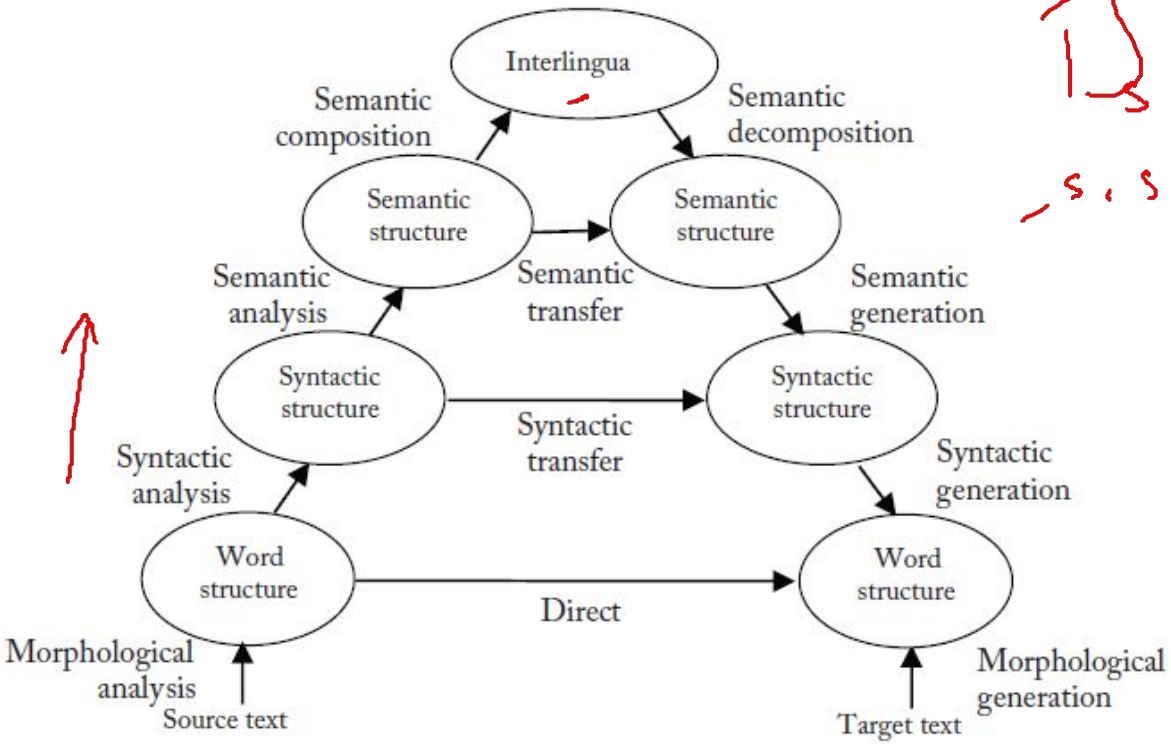


Rule-based approaches

- Expert system-like rewrite systemsInterlingua methods (analyze and generate)
- Lexicons come from humans
- Can be very fast, and can accumulate a lot of knowledge over time (e.g. Systran)

Rule-Based MT

- Traditional rule-based MT approaches usually follow what is called Vauquois triangle (Vauquois 1968)



Systran

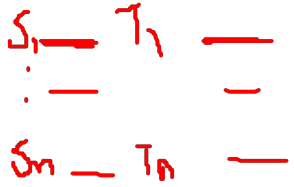
Statistical MT

- Statistical MT relies on translation examples contained in a parallel corpus, i.e., a set of texts translated into another language.

- Such a corpus can be further processed into aligned sentences

- Word-to-word translation
- Phrase-based translation
- Syntax-based translation (tree-to-tree, tree-to-string)
- Trained on parallel corpora
- Usually noisy-channel (at least in spirit)

Basic terminology



- A bitext is a pair of texts such that one is a translation of the other.
- A tritext is a triple of texts such that they are each a translation of the others.
- Parallel corpora include bitexts, tritexts, and any set of N texts, such that each is a translation of the others.
- Parallel corpora tend towards literal translations(word-for-word translation).
- Examples Canadian Hansards, Europarl croups

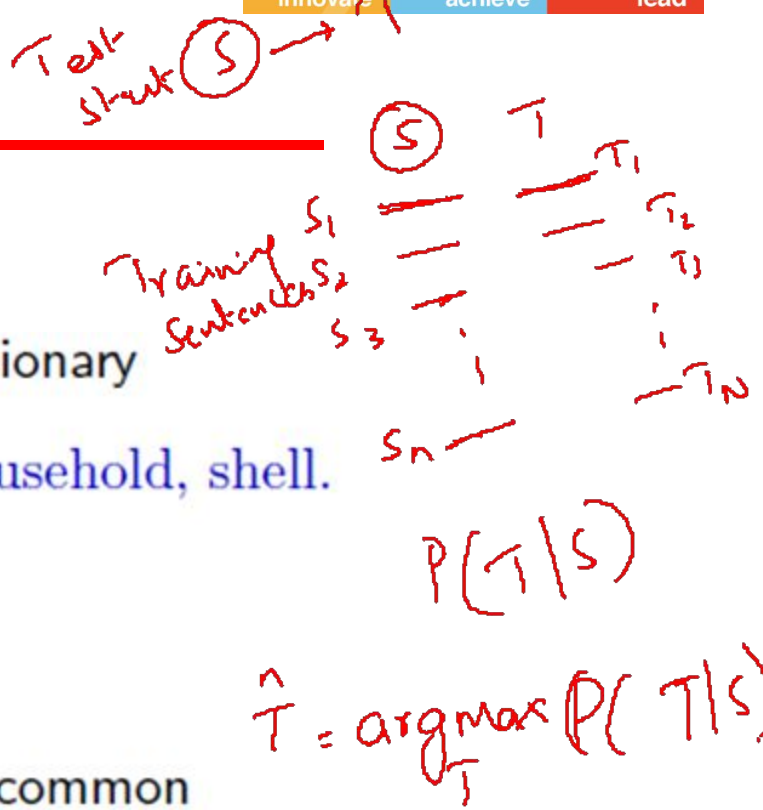
Word-based models

Lexical Translation

- How to translate a word → look up in dictionary

  Haus — house, building, home, household, shell.

- Multiple translations
  - some more frequent than others
  - for instance: house, and building most common
  - special cases: Haus of a snail is its shell



Estimating a Probability Distribution

- We now want to estimate a **lexical translation probability distribution** from these counts.

To put it formally, we want to find a function

$$p_f : e \rightarrow p_f(e)$$

given a foreign word *f* (here *Haus*), returns a probability, for each choice of English translation *e*, that indicates how likely that translation is.

Collect Statistics

Look at a parallel corpus (German text along with English translation)

Translation of *Haus*	Count
house	8,000
building	1,600
home	200
household	150
shell	50

10,000 occurrences of the word Haus

Estimate Translation Probabilities

Maximum likelihood estimation

$$p_f(e) = \begin{cases} 0.8 & \text{if } e = \text{house}, \\ 0.16 & \text{if } e = \text{building}, \\ 0.02 & \text{if } e = \text{home}, \\ 0.015 & \text{if } e = \text{household}, \\ 0.005 & \text{if } e = \text{shell}. \end{cases}$$

Alignment

Lexical translation probability tables for four German words: ***das, Haus, ist, klein***.

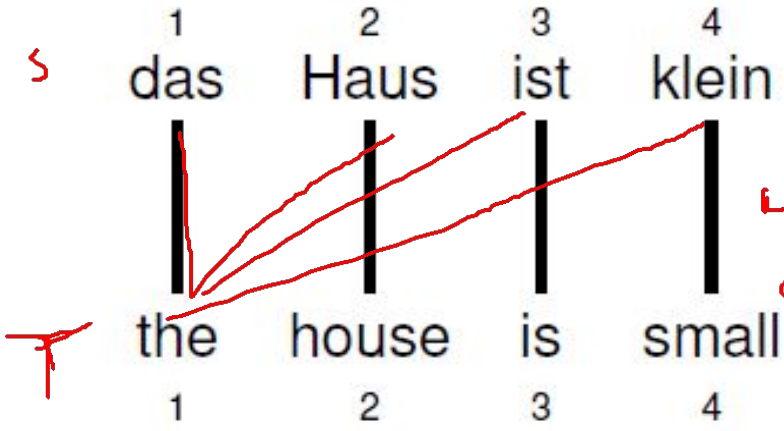
das		Haus		ist		klein	
e	*t*(*e*| *f*)	*e*	*t*(*e*| *f*)	*e*	*t*(*e*| *f*)	*e*	*t*(*e*| *f*)
the	0.7	*house*	0.8	*is*	0.8	*small*	0.4
that	0.15	*building*	0.16	*'s*	0.16	*little*	0.4
which	0.075	*home*	0.02	*exists*	0.02	*short*	0.1
who	0.05	*household*	0.015	*has*	0.015	*minor*	0.06
this	0.025	*shell*	0.005	*are*	0.005	*petty*	0.04

We now denote the probability of translating a foreign word *f* into an English word *e* with the conditional probability function *t*(*e*| *f*), denote this probability distribution as a *translation probability*.

Given the German sentence *das Haus ist klein* using the tables we can translate

Alignment

- In a parallel text (or when we translate), we align words in one language with the words in the other



- Word positions are numbered 1–4

Implicit in this translation is an **alignment**, meaning a mapping from German words to English words.

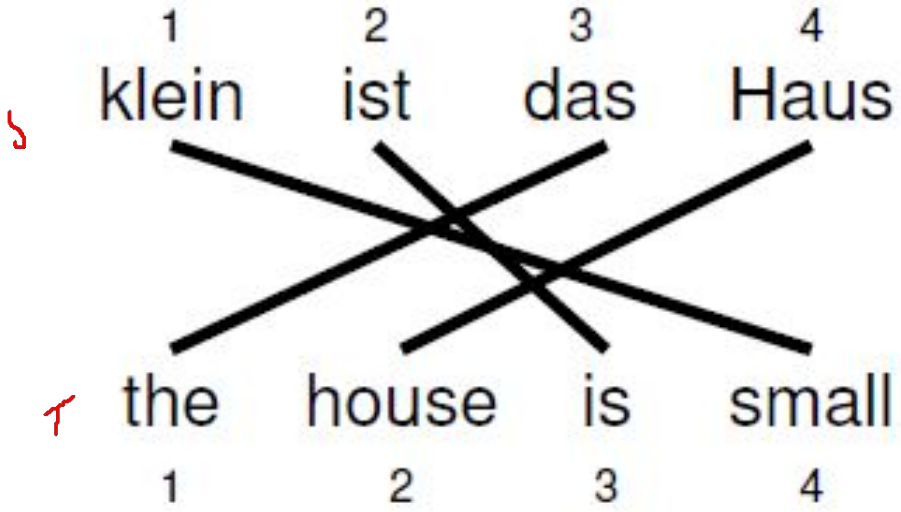
Alignment Function

- Formalizing alignment with an alignment function
- Mapping an English target word at position i to a German source word at position j with a function $a : i \rightarrow j$
- Example

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

Reordering

Words may be reordered during translation



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

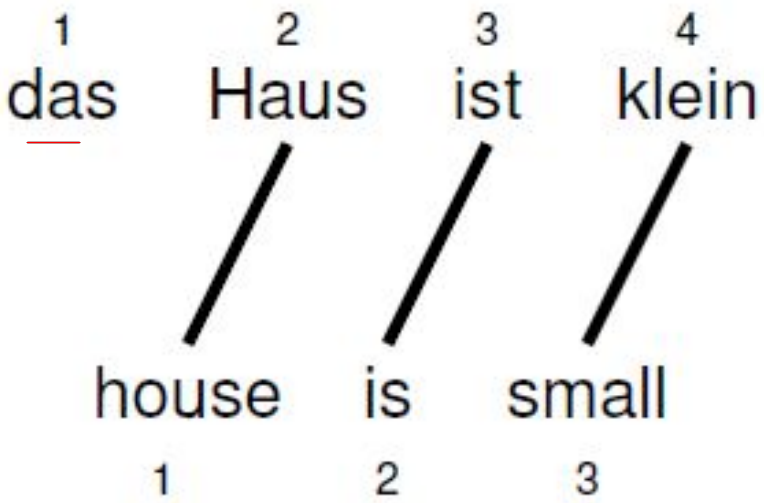
One-to-Many Translation

A source word may translate into multiple target words

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

Dropping Words

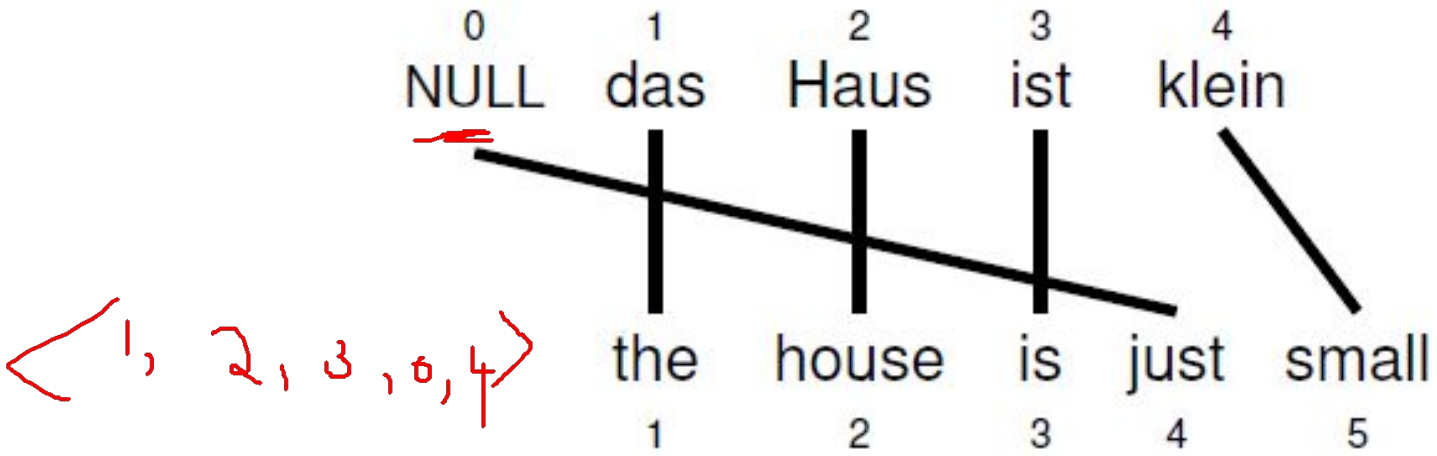
Words may be dropped when translated
(German article das is dropped)



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

Inserting Words

- Words may be added during translation
  - The English just does not have an equivalent in German
  - We still need to map it to something: special NULL token



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

Statistical MT

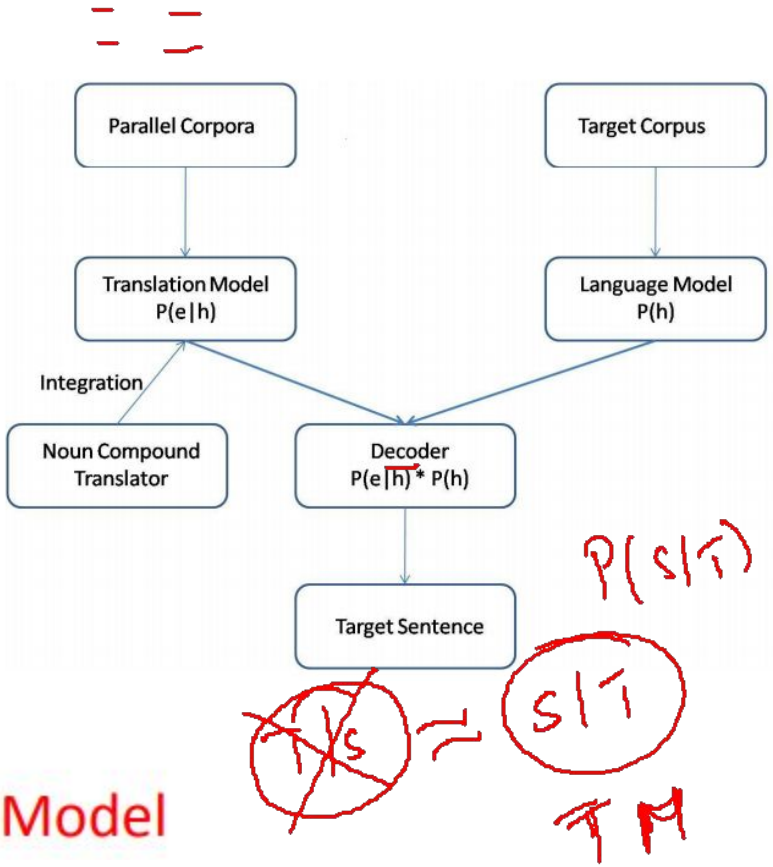
- Given a English sentence E, find a Foreign sentence F.

$$\hat{F} = \text{argmax}\, P(F \mid E)$$

$$= \text{argmax}\, \frac{P(E \mid F)P(F)}{P(E)}$$

$$= \text{argmax}\, P(E \mid F)P(F)$$

Translation Model Language Model



$l_t = 4$

1 2 3 4

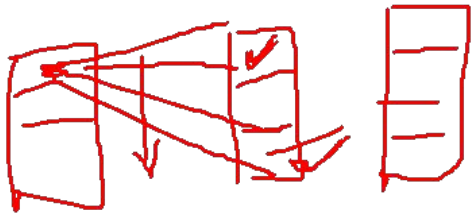
This is a rose – यह गुलाब है

This = यह

is = है

a = एक

rose = गुलाब



यह है गुलाब

है यह गुलाब

गुलाब है यह

} $P(E)$

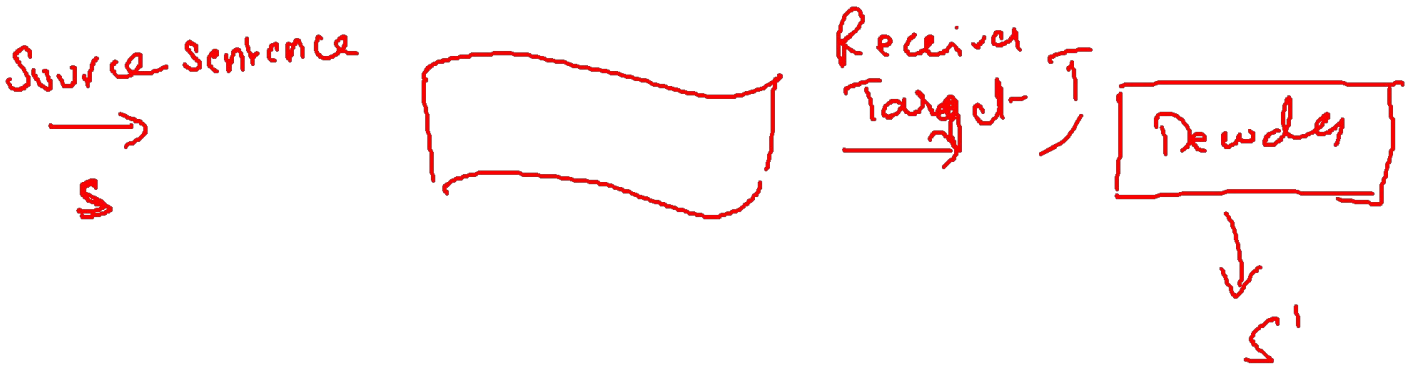
1. Choice of length l_s

2. Choice of words $s_1, s_2 \ldots s_{l_s}$

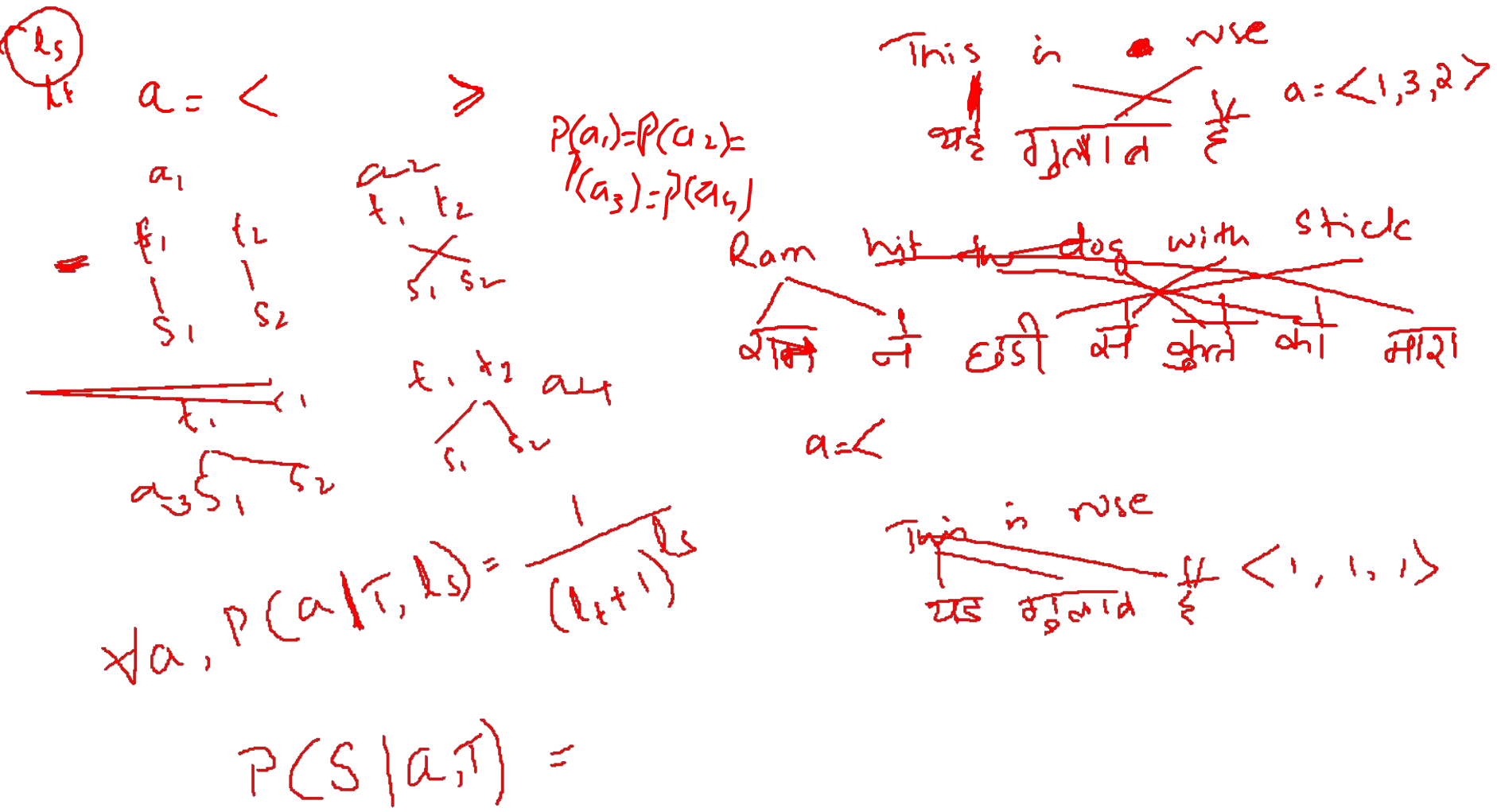
$$P(S|T) = \frac{Count(S,T)}{Count(T)}$$

$$P(s_1 s_2 s_3 \ldots s_{l_s} | t_1 t_2 \ldots t_{l_t}, l_s)$$

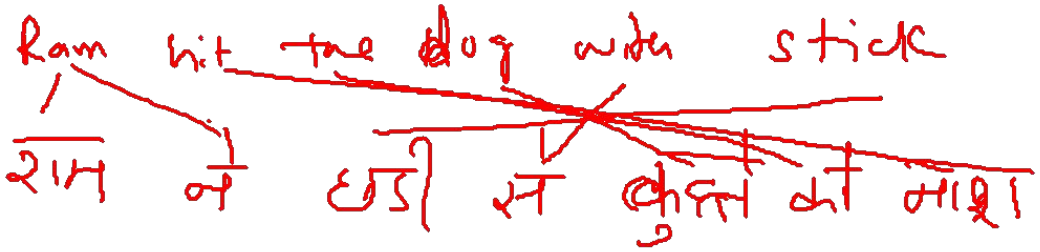
Noisy Channel Model



Alignment


innovate
achieve
lead
⟨y→1, x→3, z→2⟩
a = <
>
P(a1)=P(a2)=
P(a3)=P(a4)
This is wse
a=<1,3,2>
Ram hit dog with stick
a=<
This is wse
a=<1,1,1>
∀a, P(a|T, ls) = 1/(lt+1)^ls
P(S|a,T) =

Alignment



$$P(S|a,\bar{r}) = P(\text{राम}|\text{Ram}) \cdot P(\text{ने}|\text{Ram})\, P(\text{छड़ी}|\text{stick})\, P(\text{से}|\text{with})$$
$$P(\text{कुत्ते को}|\text{dog}) \times P(\text{को}|\text{the})\; P(\text{मारा}|\text{hit})$$

IBM Model-1

$$P(s \mid a, T) = P(a \mid T)\, P(s \mid a, T)$$

$$= \frac{1}{(l_{t+1})^{l_s}} \prod_{j=1}^{d_s} P(s_j \mid T_{a(j)})$$

TM

The noisy channel model

Generative Direction

In this model we take F and try to map to E
hence our model will be P(E|F)

Three modules of SMT

- Language Model(LM): Given F compute P(F)

  Good Foreign string → high P(F)

  Random word sequence → low P(F)

- Translation Model(TM): given(E,F) compute P(E|F)

  (E,F) look like translations →high P(E|F)

  (E,F) don't look like translations →low P(E|F)

- Decoding algorithm: given LM,TM,E, find F

  Find translation F that maximizes P(F) P(E|F)

IBM Model 1

- How do we model P(e|f)?
- English sentence e has l words $e_1,e_2..e_l$
- Foreign sentence f has m words $f_1,f_2..f_m$
- An alignment a identifies which English word each foreign word is originated from?
- Formally an alignment a is $\{a_1,,a_2 \ldots a_m\}$ where each $a_j \in \{0..\ l\}$
- There are $(l+1)^m$ possible alignments

IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation

- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$ → Equation 1

  - parameter ϵ is a normalization constant

so that $p(\mathbf{e}, a|\mathbf{f})$ is a proper probability distribution i.e $\sum_{\mathbf{e},a} p(\mathbf{e}, a|\mathbf{f}) = 1$

Example

das		Haus		ist		klein	
e	$t(e\|f)$	e	$t(e\|f)$	e	$t(e\|f)$	e	$t(e\|f)$
the	0.7	house	0.8	is	0.8	small	0.4
that	0.15	building	0.16	's	0.16	little	0.4
which	0.075	home	0.02	exists	0.02	short	0.1
who	0.05	household	0.015	has	0.015	minor	0.06
this	0.025	shell	0.005	are	0.005	petty	0.04

$$
\begin{aligned}
p(e,a|f) &= \frac{\epsilon}{5^4} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\
&= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\
&= 0.0029\epsilon
\end{aligned}
$$

Translation score
if we know the parameters t for this Translation model

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities t(e|f) from a parallel corpus
- ... but we do not have the alignments
- Chicken and egg problem
  - if we had the alignments,
    - we could estimate the parameters of our generative model
  - if we had the parameters,
    - we could estimate the alignments

EM Algorithm

- Incomplete data
  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*

- Expectation Maximization (EM) in a nutshell

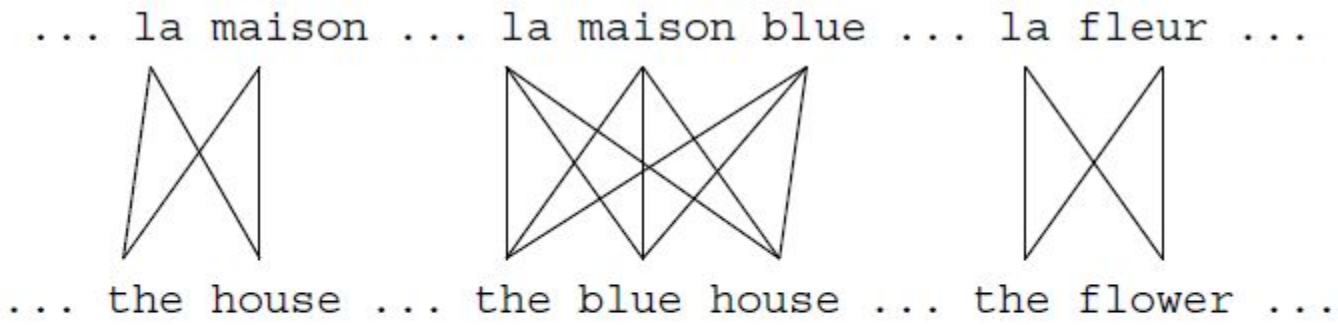
1. initialize model parameters (e.g. uniform)
2. assign probabilities to the missing data
3. estimate model parameters from completed data
4. iterate steps 2–3 until convergence

Given the parameters compute your expectation over what you think the missing data Should be (in this case it is alignments)

Given the alignments and missing data compute the model parameters given the complete data

EM Algorithm

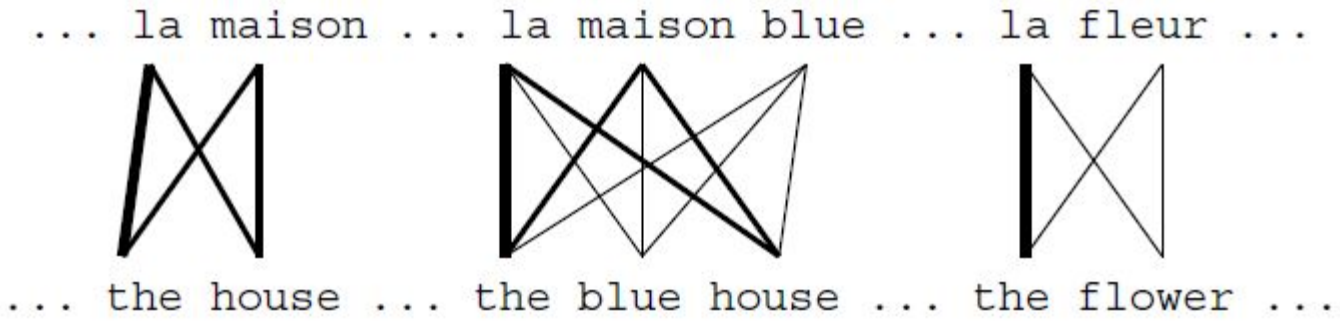
In this case we are figuring out alignments



- Initial step: all alignments equally likely
- Model learns that, e.g., la is often aligned with the

Learn translation parameters given those alignments
After one iteration we learn that the probability of getting the word the given la is higher
So that changes our idea as to what our alignment must be

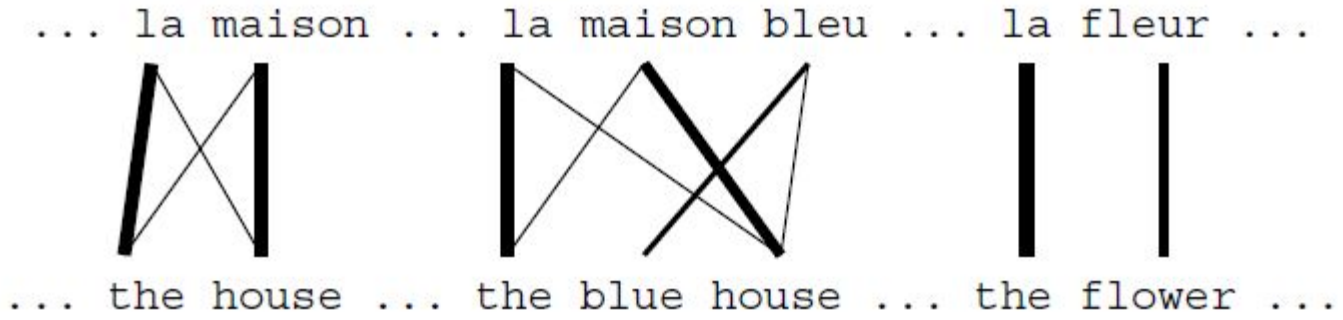
EM Algorithm



- After one iteration
- Alignments, e.g., between la and the are more likely

The expected number of alignments where the aligned to la is going to be higher
Given the alignment the - la the other alignment positions more compatible

EM Algorithm



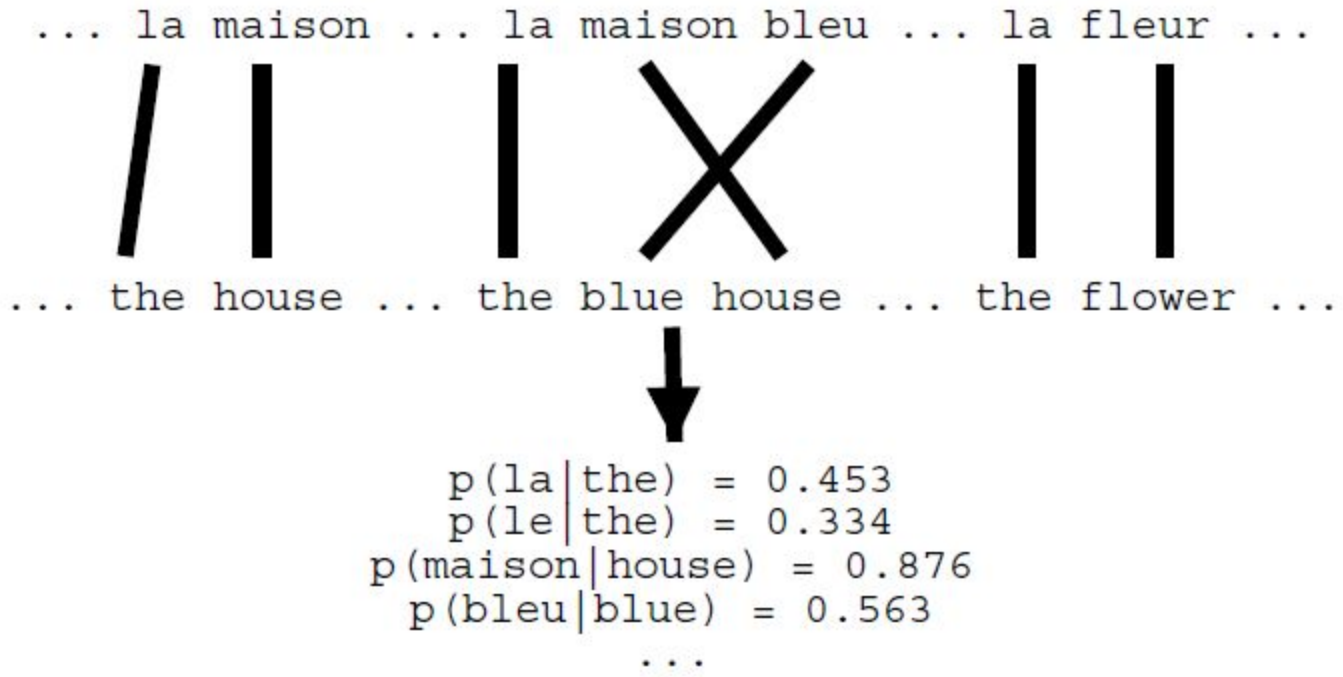
- After another iteration
- It becomes apparent that alignments, e.g., between fleur and flower are more likely (pigeon hole principle)

EM Algorithm

- Convergence
- Inherent hidden structure revealed by EM

Converged answer where we prefer very specifically some words aligned to others words

EM Algorithm



Not only alignments we also get good parameter estimation in this case the word Translation probabilities.

The Problem of Incomplete Data

- Estimating lexical translation probability distributions.
- We have sentence aligned parallel text not the word alignment.
- For each foreign input word *f* , we do not know which of the words in the English sentence is its translation.
- In other words, we lack the alignment function *a* for this data.
- We want to estimate our model from **incomplete data**.

The Expectation Maximization Algorithm

- The **expectation maximization algorithm**, or **EM algorithm**, addresses the situation of incomplete data.

The EM algorithm works as follows:

1. Initialize the model, typically with uniform distributions.
2. Apply the model to the data (expectation step).
3. Learn the model from the data (maximization step).
4. Iterate steps 2 and 3 until convergence.

IBM Model 1 and EM

- EM Algorithm consists of two steps
- Expectation-Step: Apply model to the data

Output: Alignments

  - parts of the model are hidden (here: alignments)
  - using the model(given alignments), assign probabilities to possible values

- Maximization-Step: Estimate model from data
  - take assign values as fact
  - collect counts (weighted by probabilities)
  - estimate model from counts
- Iterate these steps until convergence

IBM Model 1 and EM

- We need to be able to compute:
- Expectation-Step: probability of alignments
- Maximization-Step: count collection (Counting up how many times the words get alignment to each other)

IBM Model 1 and EM: Expectation Step

Compute the probability of an alignment given a complete English Foreign Sentence pair.

- We need to compute $p(a|\mathbf{e},\mathbf{f})$

- Applying the chain rule:

  In our case we have two events i.e **alignment** and English sentence generation.

  P(a|e) = P(e,a) / P(e) both the numerator and denominator are conditioned on f

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

- We already have the formula for $p(\mathbf{e},\mathbf{a}|\mathbf{f})$ (definition of Model 1)

P(e,a|f) probability of English sentences and alignment given a foreign sentence

So we need to compute the probability of English string given a foreign string p(e|f)

IBM Model 1 and EM: Expectation Step

- We need to compute $p(\mathbf{e}|\mathbf{f})$

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

$$= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} p(\mathbf{e}, a|\mathbf{f})$$

For every alignment

$$= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

For every word in e aligned to f

Definition of IBM model -1

IBM Model 1 and EM: Expectation Step

Probability of an alignment is constant

$$p(\mathbf{e}|\mathbf{f}) = \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

$$= \frac{\epsilon}{(l_f+1)^{l_e}} \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

Messy sums and then a product

Sum over products

$$= \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)$$

→ Equation 2

products over Sum

- Note the trick in the last line
  - removes the need for an exponential number of products

→ this makes IBM Model 1 estimation tractable

IBM Model 1 and EM: Expectation Step

- Combine what we have:

  Substituting the definition of P(e,a|f) from slide 47 Equation 1 and P(e|f) from slide 62 Equation 2

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

$$= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)}$$

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

IBM Model 1 and EM: Maximization Step

- Now we have to collect counts
- Evidence from a sentence pair e,f that word e is a translation of word f: count where single word in e is translated from the single foreign word f given the English sentence e and foreign sentence f.

$$c(e|f;\mathbf{e},\mathbf{f}) = \sum_a p(a|\mathbf{e},\mathbf{f}) \sum_{j=1}^{l_e} \delta(e,e_j)\delta(f,f_{a(j)})$$

Sum over all possible alignments

number of times e_j connects to $f_{a(j)}$ in alignment a

- With the same simplication as before:

$$c(e|f;\mathbf{e},\mathbf{f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e,e_j) \sum_{i=0}^{l_f} \delta(f,f_i)$$

IBM Model 1 and EM: Maximization Step

After collecting these counts over a corpus, we can estimate the model:

$$t(e|f;\mathbf{e},\mathbf{f}) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}{\sum_{f} \sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}$$

The probability of translating English word e from foreign word f just summing over all sentences in the corpus

Input: A training corpus $(f^{(k)}, e^{(k)}, a^{(k)})$ for $k = 1 \ldots n$, where $f^{(k)} = f_1^{(k)} \ldots f_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \ldots e_{l_k}^{(k)}$, $a^{(k)} = a_1^{(k)} \ldots a_{m_k}^{(k)}$.

Algorithm:

- Set all counts $c(\ldots) = 0$
- For $k = 1 \ldots n$
  - For $i = 1 \ldots m_k$, For $j = 0 \ldots l_k$,

$$
\begin{aligned}
c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k, i, j) \\
c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k, i, j) \\
c(j|i, l, m) &\leftarrow c(j|i, l, m) + \delta(k, i, j) \\
c(i, l, m) &\leftarrow c(i, l, m) + \delta(k, i, j)
\end{aligned}
$$

where $\delta(k, i, j) = 1$ if $a_i^{(k)} = j$, 0 otherwise.

Output: $t_{ML}(f|e) = \frac{c(e,f)}{c(e)}$, $q_{ML}(j|i, l, m) = \frac{c(j|i,l,m)}{c(i,l,m)}$

Parameter Estimation with the EM Algorithm

- Input: $(e^{(k)}, f^{(k)}), k = 1 \ldots n$

  Each $e^{(k)}$ is an English sentence, each $f^{(k)}$ is a French sentence

- The algorithm is related to algorithm with observed alignments, but with two key differences:
  - Iterative: start with initial (e.g., random) choice of q and t parameters, at each iteration: compute some "counts" base on data and parameters, and re-estimate parameters
  - The definition of of the delta function is different:

$$\delta(k, i, j) = \frac{q(j|i, l_k, m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i, l_k, m_k)t(f_i^{(k)}|e_j^{(k)})}$$

IBM Model 1 and EM: Pseudocode

Input: set of sentence pairs (e, f)
Output: translation prob. $t(e|f)$

initialize $t(e|f)$ uniformly
while not converged **do**
  // initialize
  count$(e|f) = 0$ **for all** e, f
  total$(f) = 0$ **for all** f
  for all sentence pairs (e,f) **do**
    // compute normalization
    for all words e in e **do**
      s-total$(e) = 0$
      for all words f in f **do**
        s-total(e) += $t(e|f)$
      end for
    end for
    // collect counts
    for all words e in e **do**
      for all words f in f **do**
        count$(e|f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$
        total(f) += $\frac{t(e|f)}{\text{s-total}(e)}$
      end for
    end for
  end for
  // estimate probabilities
  for all foreign words f **do**
    for all English words e **do**
      $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
    end for
  end for
end while

Take home message

- CLIR requires one extra step where either documents have to be translated into query space or viceversa.
- In general most common ways for translation are rule based or statistical methods.
- IBM Model1 is a statistical word based translation.

BITS Pilani

Dr.Aruna Malapati
Asst Professor
Department of CSIS

BITS Pilani
Hyderabad Campus

IBM Model-2

Today's Agenda

- Understand the model building in IBM Model 2 in machine translation

Reminder: IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation

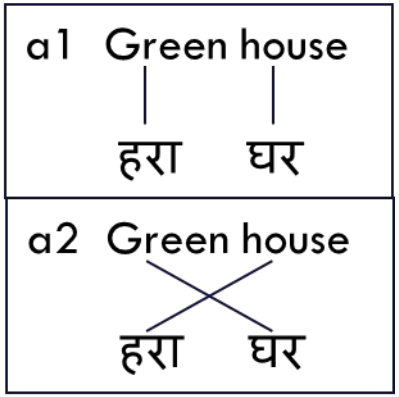
- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

  - parameter ϵ is a normalization constant

Drawbacks in IBM model -1

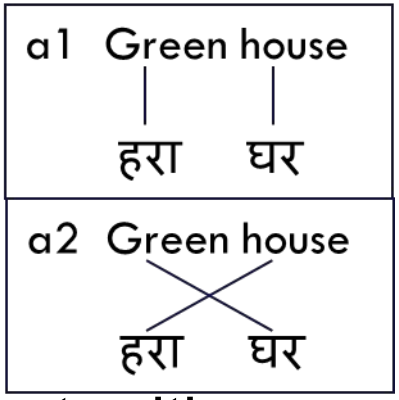
- According to IBM Model 1 the translation probabilities for the following two alternative translations are the same:



IBM Model - 2

SVO
SOV

- In **IBM Model 2**, we add an explicit model for alignment.
- In IBM Model 1 the following alignments a1 and a2 are treated equal. But some alignments are more likely than the others.



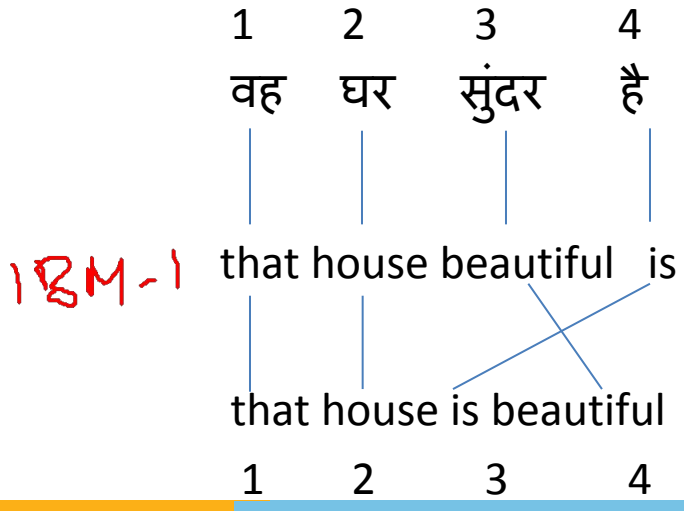
- IBM Model 2 addresses the issue of alignment with an explicit model for alignment based on the positions of the input and output words.

IBM Model - 2

- Translating source word at position i to target word at position j is modeled by an alignment probability distribution

$$a(i \mid j, l_e, l_f)$$

- We can view translation under IBM Model 2 as a two-step process

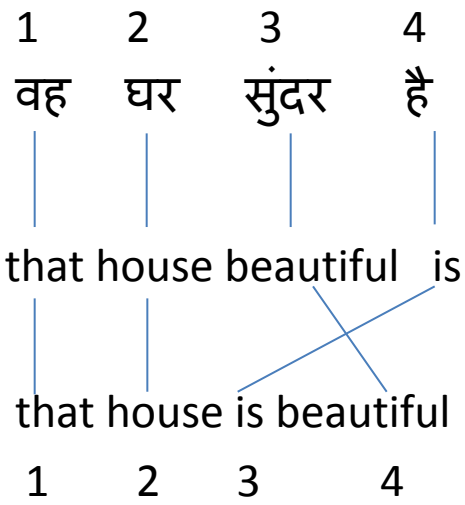


Lexical Translation

modeled as $t(e \mid f)$

alignment step

IBM Model - 2



Lexical Translation
modeled as t(e| f)

alignment step

t(वह | that) a(1|1, 4,4)
t (घर | house) a (2|2 ,4,4)
t (सुंदर | beautiful) a(4|3,4,4)
t(है | is) a(3|4, 4, 4)

For instance, translating है into *is* has a lexical translation probability of $t(is|है)$ and an alignment probability of $a(4|3, 4, 4)$ – the 3rd English word is aligned to the 4th foreign word.

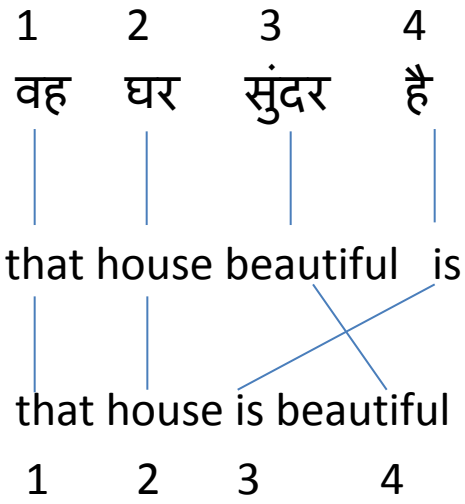
Note that the alignment function *a* maps each English output word *j* to a foreign input position *a*(*j*) and the alignment probability distribution is also set up in this reverse direction.

IBM Model - 2

- The two steps are combined mathematically to form IBM Model 2:
- The final form the model is going to be the joint probability of e and a conditioned on f

$$p(\mathbf{e}, a|\mathbf{f}) = \epsilon \prod_{j=1}^{l_e} t(e_j|f_{a(j)})\ a(a(j)|j, l_e, l_f)$$

Example



Lexical Translation

modeled as t(e| f)

alignment step

$l_e, l_f = 3$

The probability of this alignment is a=(1,2,4,3)

a(1|1,4,4) t(that| वह) X

a(2|2,4,4) X

a(4|3,4,4) X

a(3|4,4,4)

EM algorithm for parameter estimation

Expectation-Step

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

$$= \epsilon \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})\, a(a(j)|j, l_e, l_f)$$

$$= \epsilon \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_{a(j)})\, a(a(j)|j, l_e, l_f)$$

EM algorithm for parameter estimation

Maximization-step

$$c(e|f;\mathbf{e},\mathbf{f}) = \sum_{j=1}^{l_e}\sum_{i=0}^{l_f} \frac{t(e|f)\; a(a(j)|j,l_e,l_f)\; \delta(e,e_j)\; \delta(f,f_i)}{\sum_{i'=0}^{l_f} t(e|f_{i'})\; a(i'|j,l_e,l_f))}$$

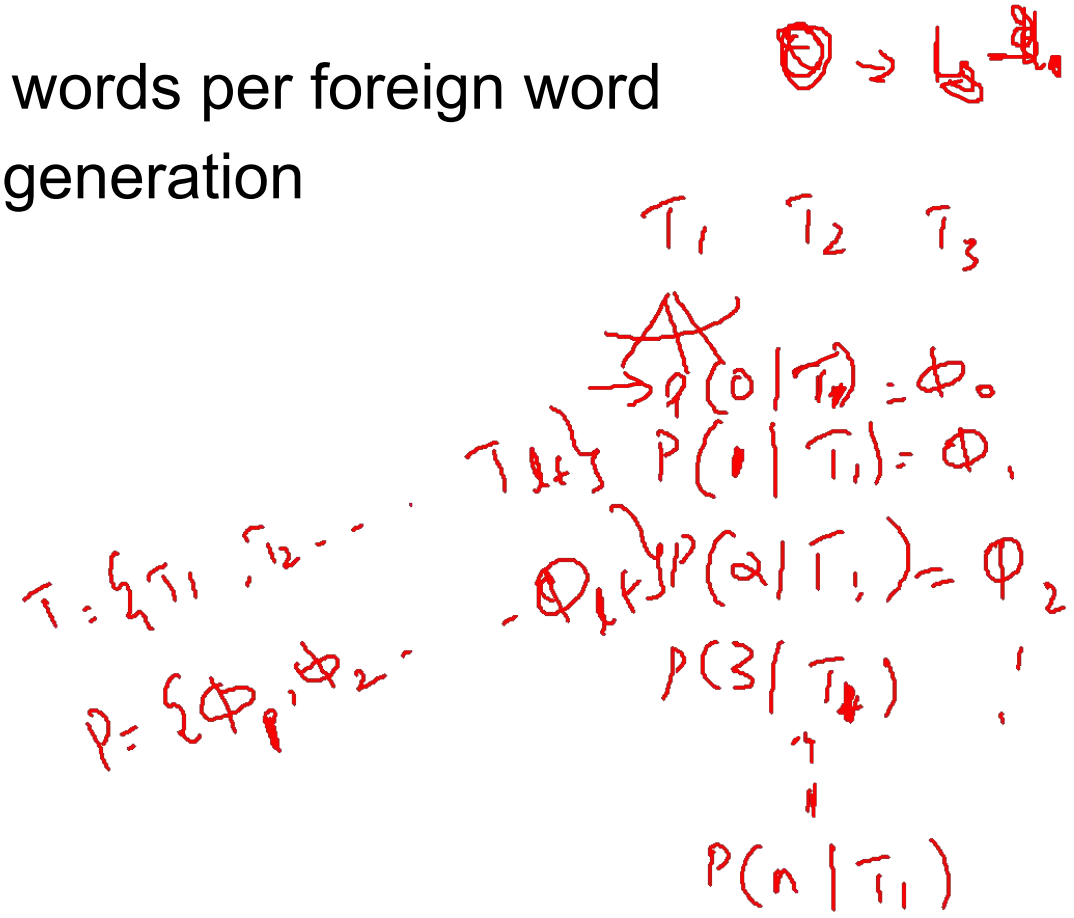
$$c(i|j,l_e,l_f;\mathbf{e},\mathbf{f}) = \frac{t(e_j|f_i)\; a(a(j)|j,l_e,l_f)}{\sum_{i'=0}^{l_f} t(e_j|f_{i'})\; a(i'|j,l_e,l_f))}$$

The algorithm for training Model 2 is very similar to that for IBM Model 1.

IBM Model 3

Features:

1. fertility: # of words per foreign word
2. NULL word generation
3. distortion



IBM Model 3

- We have not explicitly modeled how many words are generated from each input word.
- In most cases, a foreign word translates to one single English word.
- However, some foreign words like काम करने के लिए typically translate to two English words, i.e., to work.
- Others, may get dropped.

IBM Model 3

- We now want to model the **fertility** of input words directly with a probability distribution $n(\varphi | f)$
- For each foreign word f, this probability distribution indicates how many $\varphi = 0, 1, 2, \ldots$ output words it usually translates to.

n(1|घर) ~ 1
n(2| खुला होना) ~ 1
n(0|f1) ~ 1 // foreign word f1 that gets dropped

IBM Model 3

- **fertility** of input words:
  - $n(\varphi|f)$ = a probability
  - φ = 0, 1, 2, ... output words

Each input word fi generates φi output words according to the distribution n(φi| fi).

- **Note**:
  - Fertility deals explicitly with dropping input words by allowing $\varphi = 0$.
- φ_i = number of words generated by f_i
- assume f_0 (NULL)

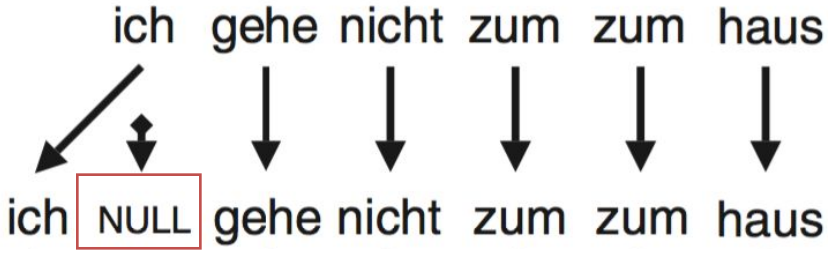
$$\sum_{i=1}^{l_f} \phi_i = l_e - \phi_0$$

IBM Model 3

- The number of inserted NULL tokens $\varphi 0$ depends on the number of output words generated by the input words.
- Each generated word may insert a NULL token.

IBM Model 3

Null Generation (Insertion) step (fs □ fs):



NULL insertion step

- # of inserted words *clearly* depends on the sentence length.
- After the fertility step, introduce one NULL token with probability p_1 **after each generated word**, or none with probability p_0=1−p_1.
- Let φ_0 be the # of NULLs generated.
- By the Binomial Distribution B(# generated words;p_0):

$$\sum_{i=1}^{l_f} \phi_i$$

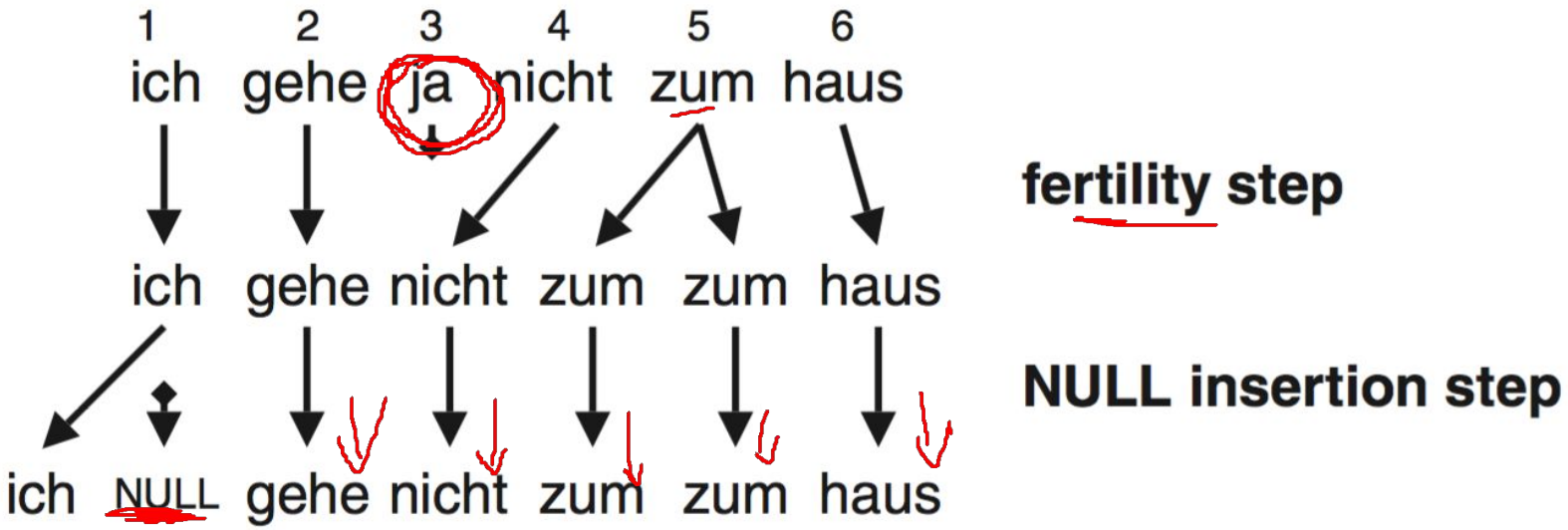
$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$p(\phi_0) = \binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} p_0^{l_e - 2\phi_0}$$

IBM Model 3

Fertility + NULL insertion:

$$\binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} p_0^{l_e - 2\phi_0} \prod_{i=1}^{l_f} \phi_i! \; n(\phi_i | e_i)$$



IBM Model 3

$$\binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} p_0^{l_e - 2\phi_0} \prod_{i=1}^{l_f} \phi_i!\ n(\phi_i | e_i)$$

Multiple ways (ϕ_i!):

- 2! $f \rightarrow {}^{f}_{f} {}^{\rightarrow}_{\rightarrow} {}^{e2}_{e1}$ or $f \rightarrow {}^{f}_{f} \times {}^{e2}_{e1}$
- 3! $e_1\ e_2\ e_3$ from $f_i\ f_i\ f_i$ (f_i fertility 3) in 6 ways

IBM Model 3

To finish up, let us combine the three elements of Model 3 – fertility, lexical translation, and distortion – in one formula:

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

$$= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} \, p_0^{l_e - 2\phi_0} \prod_{i=1}^{l_f} \phi_i! \; n(\phi_i|e_i)$$

$$\times \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \; d(j|a(j), l_e, l_f)$$

Null insertion

fertility

lexical translation

distortion

IBM Models 1-5 summary

- IBM Model 1: lexical translation
- IBM Model 2: alignment model
- IBM Model 3: fertility
- IBM Model 4: relative alignment model
- IBM Model 5: deficiency.

BITS Pilani

Dr.Aruna Malapati
Asst Professor
Department of CSIS

BITS Pilani
Hyderabad Campus

Extensions to Phrase based SMT

Today's Agenda

- Explore different features to extend SMT

Log Linear Models

- Phrase based SMT has three components in the model
  - Phrase Translation table $\phi(\bar{s}_i|\bar{t}_i)$
  - Reordering Model d
  - Language Model P_{LM}

$$s_{best} = \text{argmax}_s \prod_{i=1}^{L} \phi(\bar{s}_i|\bar{t}_i)\, d(\text{start}_i - \text{end}_{i-1} - 1) \prod_{i=1}^{|t|} P_{LM}(t_i|t_1, t_2 \ldots t_{i-1})$$

- Some sub-models may be more important than others

Log Linear Models (contd..)

- To improve our model we may want to add weights to our model to ensure fluent output.

$$S_{best} = \text{argmax}_S \prod_{i=1}^{L} \phi(\bar{s}_i|\bar{t}_i)^{\lambda_\phi} d(\text{start}_i - \text{end}_{i-1} - 1)^{\lambda_d} \prod_{i=1}^{|t|} P_{LM}(t_i|t_1 t_2 \ldots t_{i-1})^{\lambda_{LM}}$$

λ_ϕ
λ_d
λ_{LM} } weights

Log Linear Models (contd..)

- A generic log linear model is of the form

$$P(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x)$$

- Num of feature functions n=3
- Random Variable x=(s,t,start,end)
- Feature function h1=log ϕ
- Feature function h2=log d
- Feature function h3=log P_{LM}

Log Linear Models (contd..)

$$P(S, a|T) = \exp\left[\lambda_\phi \sum_{i=1}^{l} \log \phi(\bar{s}_i|\bar{t}_i) + \lambda_d \sum_{i=1}^{l} \log d(start_i - end_{i-1}) + \lambda_{LM} \sum_{i=1}^{|t|} \log P_{LM}(t_i|t_1, t_2 \ldots t_{i-1})\right]$$

Bidirectional Translation Probabilities

- Using Bayes rule we have inverted the translation probability

$$P(T|S)=P(T)\,P(S|T)\,P(S)^{-1}$$

- In the training data if an unusual phrase s is mistakenly mapped to common target phrase t then $\phi(s|t)$ will be high.
- If the same phrase is encountered in test data this phase will score high.
- Hence it is better to condition on the actual translation or use combinations schemes or weighted scheme for combinations.

Lexical Weighting

- Infrequent phrases pairs collected from noisy data are always troublesome.
- They will have $\phi(s|t) = \phi(t|s) = 1$ which overestimates how reliable rare phrases are.
- To judge the reliability of rare phrases decompose it into word translations(lexical weighting).

$$lex(s|t,a) = \prod_{i=1}^{length(\bar{s})} \frac{1}{|\{j|(i,j)\in a\}|} \sum_{\forall(i,j)\in a} w(\bar{s}_i|\bar{t}_j)$$

- If source word is aligned to multiple target words the average of the corresponding word translation probabilities is taken.

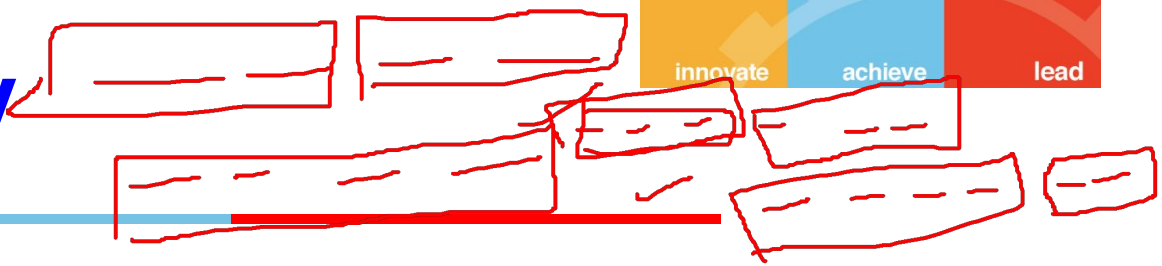
Word Penalty

- Language model has a bias towards short translations.
- We have not explicitly modeled the output length in terms of no of words. We introduce the word penalty factor ω for each produced word.

  word count: $wc(t) = \log |t|^{\omega}$

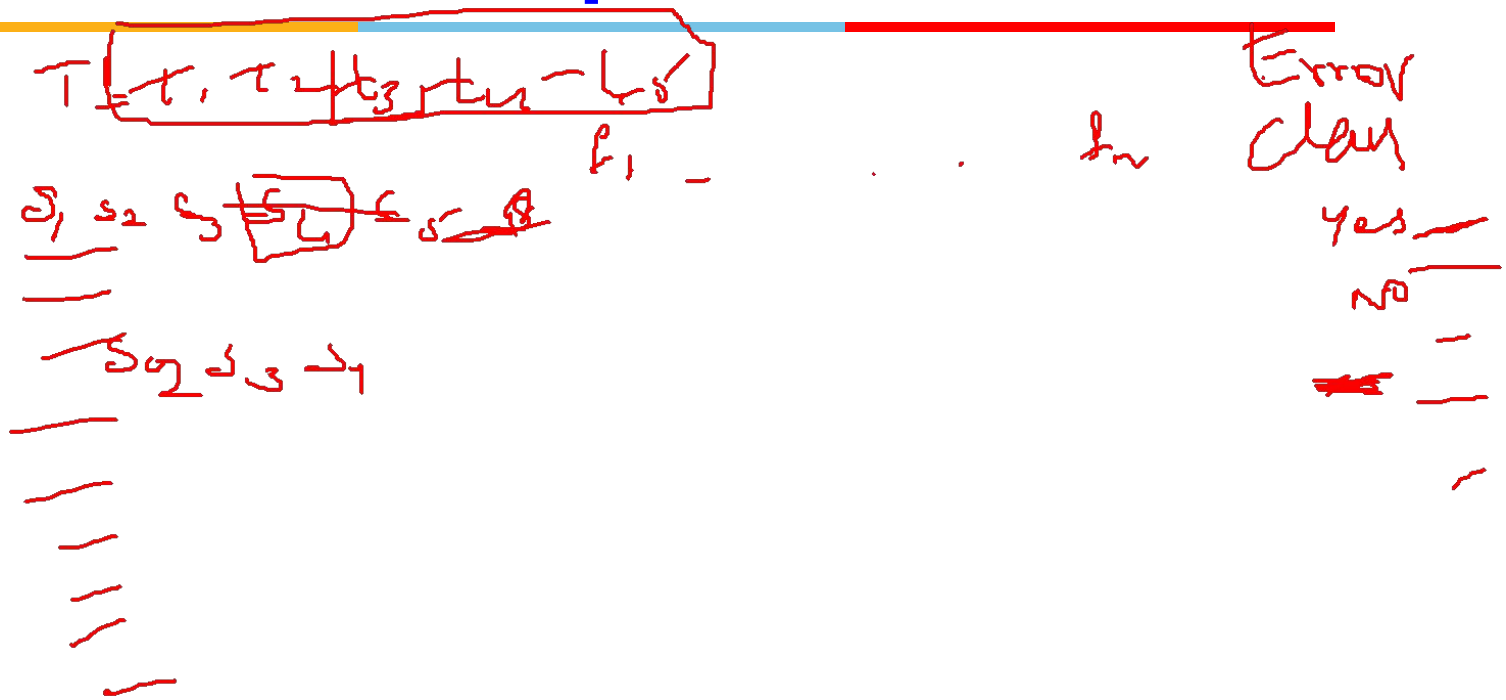
- If $\omega < 1$ we prefer shorter translations and if $\omega > 1$ we prefer longer translations.

Phrase Penalty

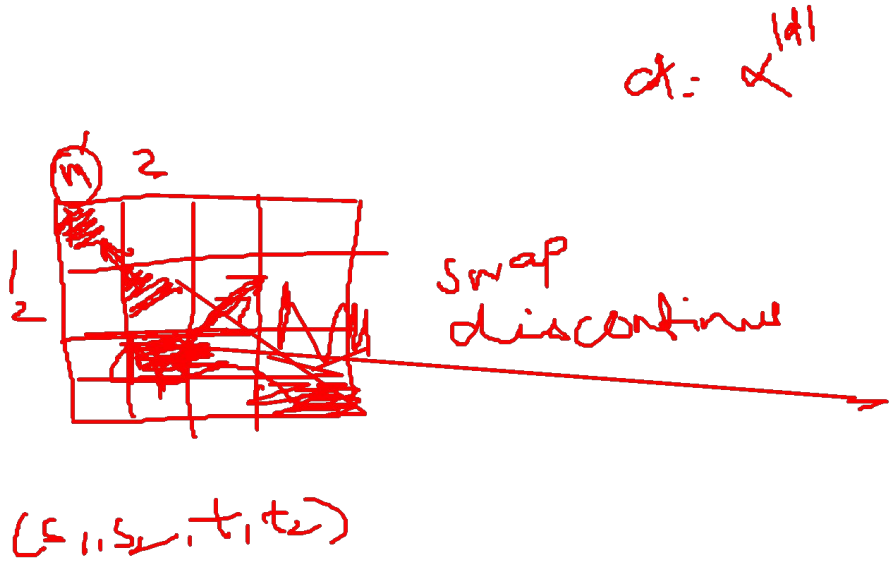


- Phrase segmentation is not explicitly modeled hence the assumption was that all phrases segmentations are equi probable.
- To control whether shorter or longer are preferred we add another parameter ρ(phrase penalty) for each phrase translation.
- If $\rho < 1$ we prefer fewer (longer) phrases and if $\rho > 1$ we prefer more (shorter) translations.

Phrase translation as a classification problem



Reordering



Decoding

Evaluation of SMT

Bilingual evaluation understudy (BLEU) score

- Given a machine generated output translation it allows to compute a score that measures how good was the translation.

Example - unigram

बिल्ली चटाई पर है

Reference1 : the cat is on the mat

Reference2 : there cat is on the mat

SMT output: the the the the the the the

Precision = 7/7 = 1

Modified Precision = 2/7

Modified Precision favors short translation.

Example-Bigram

Reference1 : the cat is on the mat

Reference2 : there cat is on the mat

SMT output: the cat the cat on the mat

Unique Bigrams	Count	$Count_{clip}$
the cat	2	1
cat the	1	0
cat on	1	1
on the	1	1
the mat	1	1

Modified Precision = 4/6

Brevity Penalty

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \leq r \end{cases}$$

R is the length of the reference sentence and c is the length of the MT sentence.