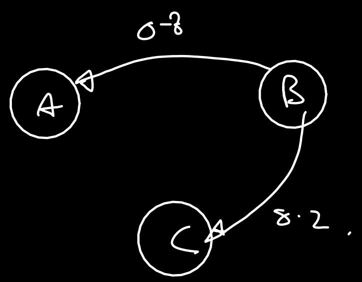
POS Tagging

— Most frequent class baseline.

Markov Assumption : $P(q_i = a \mid q_1 \dots q_{i-1}) = P(q_i = a \mid q_{i-1})$

$p(A|B) = 0.8$

$p(C|B) = 0.2$ $\Rightarrow$



A	a_1	a_2	...	a_n
a_1	a_{11}	a_{12}		a_{1n}
a_2	a_{21}	a_{22}		
$\vdots$				
a_n	a_{n1}			a_{nn}

$Q: q_1 q_2 \dots q_n$ $\quad A = a_{11} a_{12} \dots a_{nn}$

$\pi = \pi_1 \pi_2 \dots \pi_n$

HMM Model

Tags: "hidden". Words observed.

t: h $\quad$ w: o

$Q: q_1 q_2 \dots q_N$ → word tags

A	a_1	a_2	...	a_n
a_1	a_{11}	a_{12}		a_{1n}
a_2	a_{21}	a_{22}		
$\vdots$				
a_n	a_{n1}			a_{nn}

$A: a_{11} a_{12} \dots a_{NN}$

$O: o_1 o_2 \dots o_T$ → sequence of observations (words)

$B: b_i(o_t)$

$\pi = \pi_1 \pi_2 \dots \pi_N$

$b_i(o_t): P(o = o_t \mid q = q_i)$

each o_t: N 'b's

T 'o's

$P(q_i \mid q_1 \dots q_{i-1}) = P(q_i \mid q_{i-1})$

$P(o_i \mid q_1 \dots q_T, o_1 \dots o_t) = P(o_i \mid q_i)$

Basic HMM problems

1. Likelihood of a sequence

given observation sequence $O = \{o_1, o_2 \dots o_n\}$

efficiently estimate $P(O \mid \lambda)$

$O = \{o_1, o_2, \dots\dots o_T\}$

$Q = \{q_1, q_2, \dots\dots q_T\}$

Complexity:

$O(2TN^T)$

Now,

$$P(O \mid Q, \lambda) = \prod_{i=1}^{T} P(o_i \mid q_i, \lambda)$$

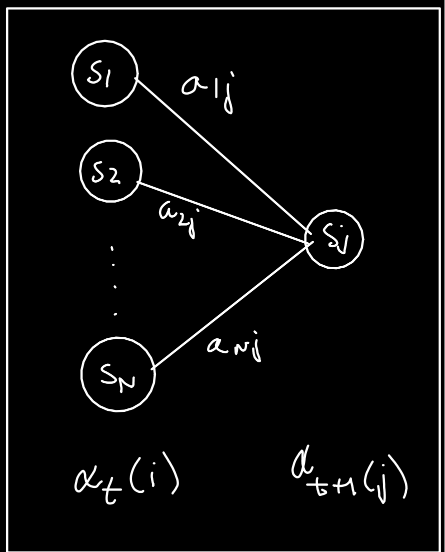
$$= b_{q_1}(o_1) \dots b_{q_T}(o_T)$$

$$P(Q \mid \lambda) = \pi_q \, a_{q_1 q_2} \dots a_{q_{T-1}} a_{q_T}$$

$$P(O \mid \lambda) = \sum \pi_q \, a_{q_1 q_2} \dots a_{q_{T-1}} a_{q_T} \, b_{q_1}(o_1) \dots b_{q_T}(o_T)$$

$[P(A,B) = P(A \mid B) \cdot P(B)]$

Forward Procedure :



DEF:

$$\alpha_t(i) = P(o_1 \dots o_t, q_t = S_i \mid \lambda)$$

given partial obs. seq$_s$

Prob. that State at position t is S_i

STEP:

1. Init

$$\alpha_1(i) = \pi_i b_i(o_1)$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} \right] b_j(o_{t+1})$$

3. Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Backward Procedure

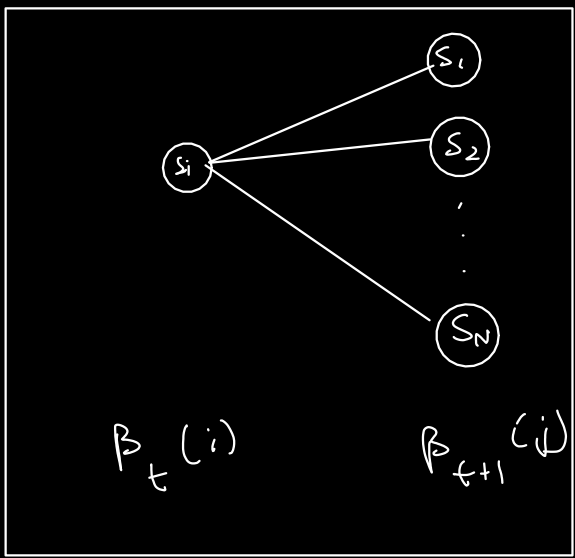
$$\beta_t(i) = P(O_{t+1} \ldots O_T \mid q_t = S_i, \lambda)$$

1. Init

$$\beta_T(i) = 1$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$



$$\boxed{O(N^2 T)}$$

2. Given observation sequence $O = \{o_1, \ldots o_T\}$

Get best $Q = \{q_1 \ldots q_T\}$

Solution:

- Best state individually likely at a position i
- Best state given all the previously observed states and observations
- Viterbi Algorithm

Viterbi Algorithm

DEF

$$\delta_t(i) = \max_{q_1 \ldots q_{t-1}} P(q_1 q_2 \ldots q_t = i, O_1 O_2 \ldots O_t \mid \lambda)$$

$$\Rightarrow \delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij} \right] b_j(O_{t+1})$$

Init

$$\delta_1(i) = \pi_i \, b_i(o_1)$$

$$\psi(i) = 0$$

Recursion

$$\delta_t(j) = \max \left[\delta_{t-1}(i) \, a_{ij} \right] b_j(O_t)$$

$$\psi_1(j) = \underset{1 \leq i \leq N}{\operatorname{argmax}} \left[\delta_{t-1}(i) \, a_{ij} \right]$$

Termination

$$p^* = \max_{1 \leq i \leq N} \left(\delta_T(i) \right)$$

$$q_T^* = \underset{1 \leq i \leq N}{\operatorname{argmax}} \left(\delta_T(i) \right)$$

Backtracking

$$q_t^* = \psi_{t+1}\left(q^*_{t+1} \right)$$

Tokenisation, LM, Smoothing

Tokenization

Challenges :
- hyphenation
- number
- dates
- abbreviation
- punctuation
- URLs
- sentencification

N-grams

$$P(w \mid s(w)) = \frac{c(w, s(w) \ldots)}{c(s(w) \ldots)}$$

Language Model

A module that computes $P(W)$ or $P(w_n \mid w_1, w_2, \ldots w_{n-1})$

Perplexity and Entropy

Entropy over RV : X

$$H(X) = -\sum_{x=1}^{n} p(x) \log p(x)$$

Perplexity:

$$PP(w) = 2^{H(W)}$$

$$\text{Perplexity} = 2^{\text{entropy}} = 2^{\text{avg. \# of bits}}$$

(of our model q)

tot # of words

X can be any R.V. In NLP, it's each word.

$$= 2^{-\sum_{i=1}^{N} \underbrace{p(x_i)}_{\text{real dist.}} \cdot \underbrace{\log_2 q(x_i)}_{\text{our prediction!}}}$$

$$= e^{-\sum_{i=1}^{N} p(x_i) \cdot \ln q(x_i)}$$

We assume all words have the same frequency.

$$= e^{-\sum_{i=1}^{N} \frac{1}{N} \cdot \ln q(x_i)}$$

$$= q(x_1)^{-\frac{1}{N}} \cdot q(x_2)^{-\frac{1}{N}} \cdots q(x_N)^{-\frac{1}{N}}$$

$$= \prod_{i=1}^{N} q(x_i)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{q(x_1)\, q(x_2) \cdots q(x_n)}}$$

Smoothing

Laplace: Add 1 to all counts.

Linear Interpolation:

$$\hat{P}(w_n \mid w_{n-1} w_{n-2}) = \lambda_1 P(w_n \mid w_{n-1} w_{n-2}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Weights from held-out corpus.

Katz

$$P_{katz}\left(w_n \mid w_{n-N+1}^{n-1}\right)=\begin{cases} P^*\left(w_n \mid w_{n-N+1}^{n-1}\right) & \text{if } C\left(w_{n-N+1}^{n}\right)>0 \\ \alpha\left(w_{n-N+1}^{n-1}\right)P_{katz}\left(w_n \mid w_{n-N+2}^{n-1}\right) & \text{otherwise} \end{cases}$$

The w_i, w_{i-1}, w_{i-2} for clarity are referred as a sequence x, y, z.
Katz method incorporates discounting as integral part of the algorithm.

$$P_{katz}(z \mid x,y)=\begin{cases} P^*(z \mid x,y) & \text{if } C(x,y,z)>0 \\ \alpha(x,y)P_{katz}(z \mid y) & \text{else if } C(x,y)>0 \\ P^*(z) & \text{otherwise} \end{cases}$$

$$P_{katz}(z \mid y)=\begin{cases} P^*(z \mid y) & \text{if } C(y,z)>0 \\ \alpha(y)P_{katz}(z) & \text{otherwise} \end{cases}$$

$$P^*\left(w_n \mid w_{n-N+1}^{n-1}\right)=\frac{c^*\left(w_{n-N+1}^{n}\right)}{c\left(w_{n-N+1}^{n-1}\right)}$$

Kneser-Ney

- Re-estimated counts c^* for greater than 1 counts could be estimated pretty well by just subtracting 0.75 from the MLE count c.
- Absolute discounting method formalizes this intuition by subtracting a fixed (absolute) discount d from each count.
  - The rational is that we have good estimates already for the high counts, and a small discount d won't affect them much.
  - The affected are only the smaller counts for which we do not necessarily trust the estimate anyhow.
- The equation for absolute discounting applied to bigrams (assuming a proper coefficient α on the backoff to make everything sum to one) is:

$$P_{absolute}(w_i \mid w_{i-1})=\begin{cases} \frac{c(w_{i-1}w_i)-D}{c(w_{i-1})}, & \text{if } c(w_{i-1}w_i)>0 \\ \alpha(w_i)P_{absolute}(w_i) & \text{otherwize} \end{cases}$$

- In practice distinct discount values d for the 0 and 1 counts are computed.
- Kneser-Ney discounting augments absolute discounting with a more sophisticated way to handle the backoff distribution. Consider the job of predicting the next word in the sentence, assuming we are backing off to a unigram model:
  - I can't see without my reading XXXXXX.
- The word "*glasses*" seem much more likely to follow than the word "*Francisco*".
  - But "*Francisco*" is in fact more common, and thus a unigram model will prefer it to "*glasses*".

1. Thus we would like to capture that although "*Francisco*" is frequent, it is only frequent after the word "*San*".
2. The word "*glasses*" has a much wider distribution.

Thus the idea is instead of backing off to the unigram MLE count (the number of times the word w has been seen), we want to use a completely different backoff distribution!

- We want a heuristic that more accurately estimates the number of times we might expect to see word w in a new unseen context.
- The Kneser-Ney intuition is to base our estimate on the number of ***different contexts word w has appeared in***.
- Words that have appeared in more contexts are more likely to appear in some new context as well.
- New backoff probability can be expressed as the "***continuation probability***" presented in following expression:

- Continuation Probability:

$$P_{continuation}(w_i) = \frac{\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}{\sum_{w_i} \left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}$$

- Kneser-Ney backoff is formalized as follows assuming proper coefficient α on the backoff to make everything sum to one:

$$P_{KN}(w_i \mid w_{i-1}) = \begin{cases} \frac{c(w_{i-1}w_i) - D}{c(w_{i-1})}, & \text{if } c(w_{i-1}w_i) > 0 \\ \alpha(w_i) \frac{\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}{\sum_{w_i} \left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|} & \text{otherwize} \end{cases}$$

- Kneser-Ney **backoff** algorithm was shown to be less superior to its **interpolated** version. **Interpolated Kneser-Ney** discounting can be computed with an equation like the following (omitting the computation of β):

$$P_{KN}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) - D}{c(w_{i-1})} + \beta(w_i) \frac{\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}{\sum \left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}$$

- Practical note – it turns out that any interpolation model cab be represented as a backoff model, and hence stored in ARPA backoff format. The interpolation is done when the model is built, thus the 'bigram' probability stored in the backoff format is really 'bigram already interpolated with unigram'.

WITTEN BELL

HYDERABAD

Witten-Bell Smoothing

- Let's consider bi-grams
  - $T(w_{i-1})$ is the number of different words (types) that occur to the right of w_{i-1}
  - $N(w_{i-1})$ is the number of all word occurrences (tokens) to the right of w_{i-1}
  - $Z(w_{i-1})$ is the number of bigrams in the current data set starting with w_{i-1} that do not occur in the training data

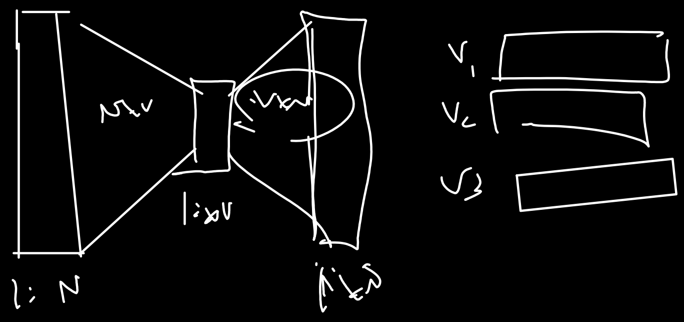
- If $c(w_{i-1},w_i) = 0$

$$P^{WB}(w_i \mid w_{i-1}) = \frac{T(w_{i-1})}{Z(w_{i-1})(N + T(w_{i-1}))}$$

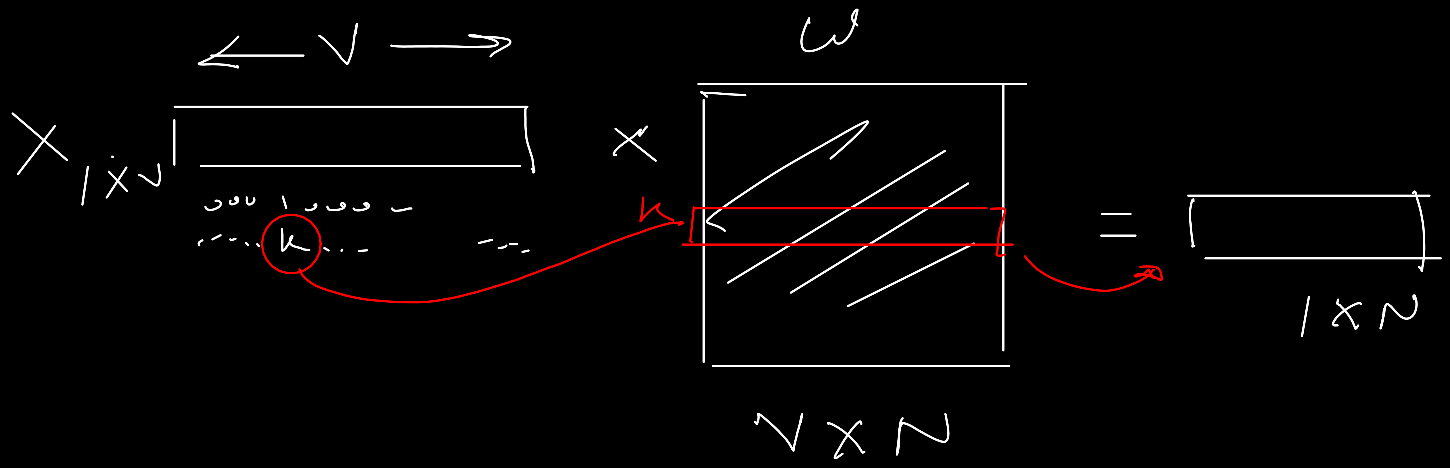
- If $c(w_{i-1},w_i) > 0$

$$P^{WB}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{N(w_{i-1}) + T(w_{i-1})}$$

Word2Vec

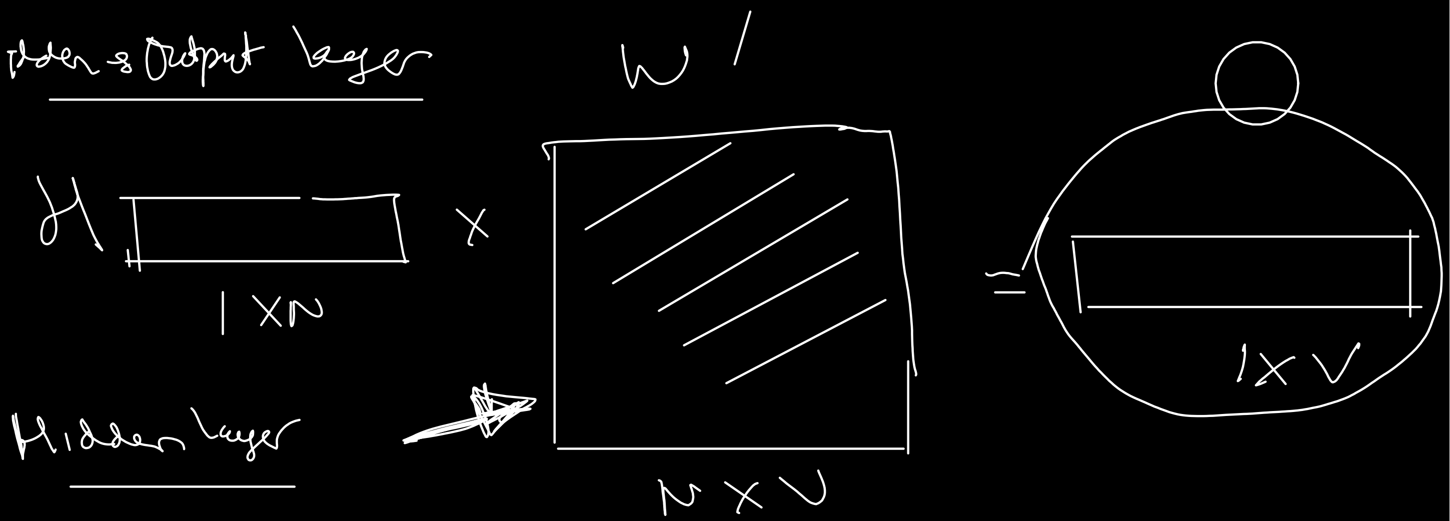


Input→hidden layer

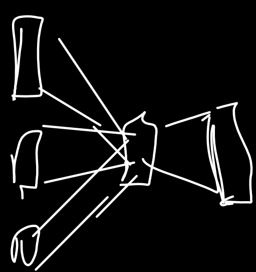


$$X_{1\times V} \times W_{V\times N} = H_{1\times N}$$

Hidden→Output layer



$h = xW$



$h = \frac{1}{C}(x_1 \dots x_n)W$

https://arxiv.org/pdf/1411.2738.pdf

https://ronxin.github.io/wevi/

GloVe

- uses global count statistics.
- uses co-occurence matrix

We have P ratios,

As you can see, words that are related to the nature of "ice" and "steam" ("solid" and "gas" respectively) occur far more often with their corresponding words that the non-corresponding word. In contrast, words like "water" and "fashion" which are not particularly related to either have a probability ratio near 1. Note that the

$$\frac{P(k|ice)}{P(k|steam)}$$

k = solid : 8.9

k = gas : 8.5×10^{-2}

k = fashion, water $\sim = 1$

Building co-occurrence can be done during vocab building, so does not affect complexity.

General Word Embeddings

<u>Matrix methods</u> : LSA (SVD), HAL, COALS, Hellinger PCA

<u>Neural approaches</u> : NNLM HLBL RNN, Skip-gram / CBOW / vLBL

<u>Matrix</u> v <u>Neural</u>

Matrix
- stat
- fast train
- scales poor
- similarity

→ Disprop to small counts

Neural
- ineff stat
- slow train
- scales with size.
- more than sim.

Non compositionality

Similarity Datasets

MEN	3000 w pairs
RW	2034 " "

Evaluation

- NN
- Analogy
- IR
- Sem tagging.

Karakas

k1: karta

k2: karma

k3: karan (instrument)

k4: recipient

k6: source

k7: →

t → time

p → place

Thematic Roles

- Agent
  sentient (ate → lions)
- Instrument
  (opened → key)
- Cause
  (killed → epidemic)
- Experiencer
- Recipient
- Path
- Location
- Measure
- Theme

The boy plucked the flower
det n v det n.
1 2 3 4 5

Shift_Reduce Parsing

Rules :

R1	S → NP VP
R2	NP → det n
R3	PP → prep NP
R4a	VP → v
R4b	VP → v NP
R4c	VP → v NP NP
R4d	VP → VP PP

Sl. No	stack	Inp rem.	Action	Rules rem.
1	—	1		nil.
1'	det	2	shift	R1
1''	det n	3	Shift	R1
2	NP	3	reduce	R3
2'	NP v	4	Shift	R1
3	NP VP	4	reduced	R4b.
4	S	4	reduce	R2
4'	S det	5	Shift	R1
4''	S det n	null	shift	R1

Sl. No

5 S NP null reduce R3

Error : Backtrack

3' NP VP det 5 shift R1

3'' NP VP det n null shift R1

4 NP VP NP null reduce R3

Error : Backtrack

2''' NP v det 5 shift R1

2'''' NP v det n null shift R1

3 NP v NP null reduce R3

4 NP VP null reduce R4c

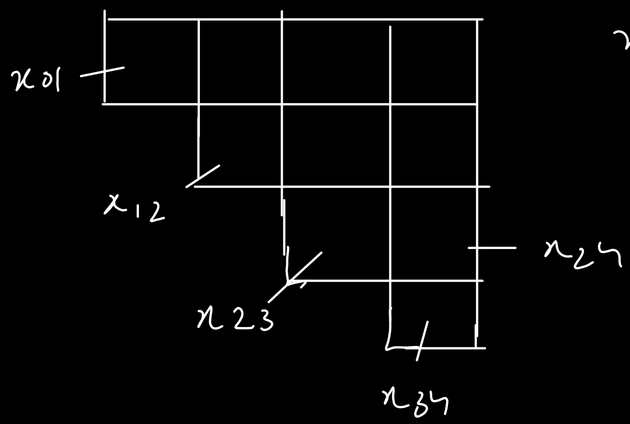
5 S null reduce R2

Accept

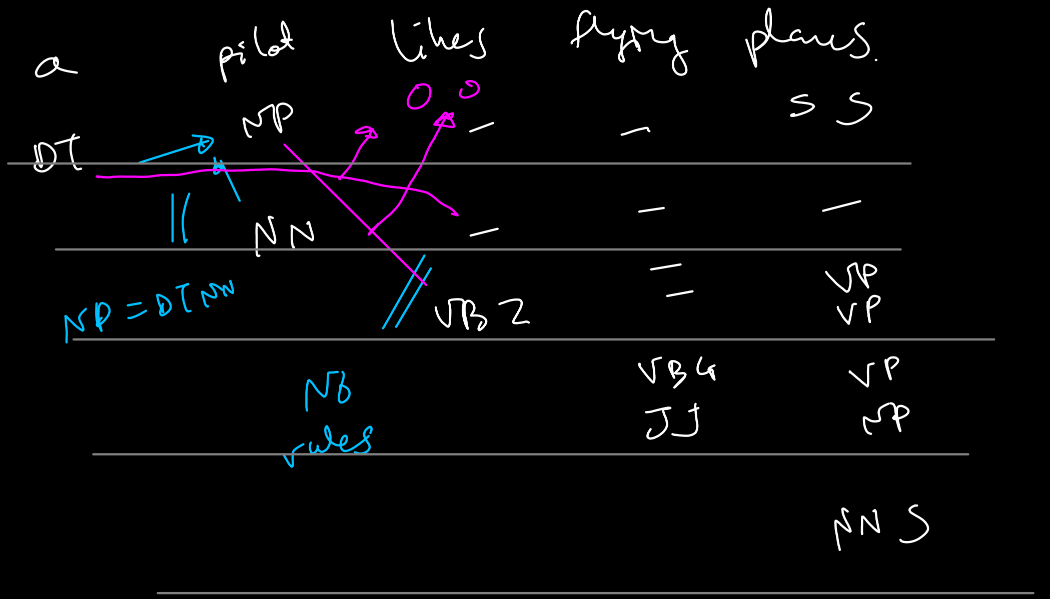
CYK Parsing

| w_1 | w_2 | w_3 | . . . | w_n |

0 1 2 3 n-1 n



$x_{ij} : w_{i+1} \dots w_j$



05 = 01 15

02 25

03 35

04 45

- . The mother goes . . .

null ⊥ null

gender ⊥ gender.

spec: det
root : mother
attr : n : sing
gender : fem

goes: root : go
attr : T : present
A : simple.
N : sing .