

# Statistical Methods in Natural Language Processing

## 6. Mutual Information and Word Classes

Pavel Pecina, Jindřich Helcl

18 November, 2025

# Course Segments

1. Introduction, probability, essential information theory
2. Statistical language modelling (n-gram)
3. Statistical properties of words
4. Word embeddings
5. Hidden Markov models, Tagging

## Recap from Last Week

# Collocations

- J. R. Firth (1957):
  - “*You shall know a word by the company it keeps.*”
  - “Collocations of a given word are statements of the habitual or customary places of that word.”
- M&S (Chapter 5):
  - “A collocation is an expression consisting of two or more words that correspond to some conventional way of saying things.”
- Examples:
  - *strong tea, weapons of mass destruction*
  - *to make up, the rich and powerful*
- Valid or invalid?
  - *a stiff breeze, but not a stiff wind*
  - *broad daylight, but not bright daylight*

# Properties of Collocations

- Typical properties/criterions of collocations:
  - non-compositionality
  - non-substitutability
  - non-modifiability
- Collocations usually cannot be translated word-by-word
  - e.g. *take a shower* → *osprchovat se*, but not *vzít sprchu*
- A phrase can be a collocation even if it is not consecutive
  - e.g. *knock ... door*

# How to Find Collocations?

- Frequency (simplest method)
  - plain
  - filtered
- Mean and variance of the distance between focal word and collocating word
- Hypothesis testing
  - $t$  test
  - $\chi^2$  test
- Pointwise Mutual Information

# Hypothesis Testing

- Two words can co-occur by chance
  - High frequency and low variance can be accidental
- Hypothesis Testing measures the confidence that this co-occurrence was really due to association, and not just due to chance.
- Formulate a **null hypothesis**  $H_0$  that there is no association between the words beyond chance occurrences:
  - $H_0$  states what should be true if two words do not form a collocation.
  - If  $H_0$  can be rejected, the words do not co-occur by chance, and they form a collocation
- Compute the probability  $p$  that the event would occur if  $H_0$  were true:
  - reject  $H_0$  if  $p$  is too low (typically beneath a significance level of  $p < 0.05, 0.01, \dots$ )
  - retain  $H_0$  as possible otherwise.

## *t*-test (Student's *t*-test)

- The test looks at the difference between the **observed** and **expected** means, scaled by the variance of the data, and tells us how likely one is to get a sample of that mean and variance, assuming that the sample is drawn from a normal distribution with mean  $\mu$ .

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}}$$

- Where  $\bar{x}$  is the real data mean (**observed in data**)
- $s^2$  is the variance
- $N$  is the sample size
- $\mu$  is the mean of the distribution (**expected under  $H_0$** )

# $\chi^2$ test: An example

- Observed occurrences:

	$w_1 = \text{new}$	$w_1 \neq \text{new}$
$w_2 = \text{companies}$	8 (new companies)	4667 (e.g., old companies)
$w_2 \neq \text{companies}$	15820 (e.g., new machines)	14287181 (e.g., old machines)

- The  $\chi^2$  statistic sums differences between observed and expected values in all cells of the table, scaled by the magnitude of the expected values:

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

- where  $i$  ranges over rows of the table,  $j$  ranges over columns,  $O_{ij}$  is the observed value for cell  $(i, j)$  and  $E_{ij}$  is the expected value.



## $\chi^2$ test: A Simpler Form

- $\chi^2$  test can be applied to tables of any size, but it has a simpler form for 2-by-2 tables (i.e. bigram collocations):

$$\chi^2 = \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{(O_{11} + O_{12})(O_{11} + O_{21})(O_{12} + O_{22})(O_{21} + O_{22})}$$

# Pointwise Mutual Information

- An information-theoretically motivated measure for discovering interesting collocations is pointwise mutual information
- It is a measure of how much one word tells us about the other:

$$\text{PMI}(x, y) = \log \frac{P(x, y)}{P(x) P(y)}$$

← “observed”  
← “expected (under  $H_0$ )”

- $P(x, y)$  is the joint probability of  $x$  and  $y$  occurring together,
- $P(x)$  and  $P(y)$  are the individual probabilities of  $x$  and  $y$
- PMI is NOT the MI as defined in Information Theory (MI is the average of PMI)

# Mutual Information and Word Classes

# The Problem

Not enough data:

- Language Modeling: we do not see “correct” n-grams
  - solution so far: smoothing
- suppose we see:
  - *short homework, short assignment, simple homework*
- but not:
  - *simple assignment*
- What happens to our (bigram) LM?
  - $p(\text{homework} \mid \text{simple}) = \text{high probability}$
  - $p(\text{assignment} \mid \text{simple}) = \text{low probability (smoothed with } p(\text{assignment}))$
- They should be much closer!

# Word Classes

Observation:

- similar words behave in a similar way
- i.e. appear in similar context (the Distributional Hypothesis)
- Assuming trigram language model:
  - *a ... homework* (any attribute of homework: *short, simple, late, difficult*),
  - *... the woods* (any verb that has the woods as an object: *walk, cut, save*)
  - *a (short,long,difficult,...) (homework,assignment,task,job,...)*

# Solution

Use the Word Classes as the “reliability” measure:

- Example: we see
  - *short homework, short assignment, simple homework*
- but not:
  - *simple assigment*
- Cluster into classes:
  - *(short, simple) (homework, assignment)*
  - covers “*simple assignment*”, too
- Gaining:
  - realistic estimates for unseen n-grams
- Losing:
  - accuracy (level of detail) within classes

# The New Model

Rewrite the n-gram LM using classes:

- Original definition: [ $k = 1 \dots n$ ]

$$p_k(w_i|h_i) = c(h_i, w_i) / c(h_i) \quad [\text{history: } (k-1) \text{ words}]$$

- Introduce classes:

$$p_k(w_i|h_i) = p(w_i|c_i) p_k(c_i|h_i)$$

- history: classes, too: [for trigram:  $h_i = c_{i-2}, c_{i-1}$ , bigram:  $h_i = c_{i-1}$ ]
- Smoothing as usual
  - over  $p_k(w_i|h_i)$ , where each is defined as above (except uniform which is  $1/|V|$ )

# Training Data

- Suppose we already have a mapping:
  - $r: V \rightarrow C$  assigning each word its class ( $c_i = r(w_i)$ )
- Expand the training data:
  - $T = (w_1, w_2, \dots, w_{|T|})$  into
  - $T_C = (\langle w_1, r(w_1) \rangle, \langle w_2, r(w_2) \rangle, \dots, \langle w_{|T|}, r(w_{|T|}) \rangle)$
- Effectively, we have two streams of data:
  - word stream:  $w_1, w_2, \dots, w_{|T|}$
  - class stream:  $c_1, c_2, \dots, c_{|T|}$  (def. as  $c_i = r(w_i)$ )
- Expand Heldout, Test data too

# Training the New Model

- Using ML estimates (as expected):
  - $p(w_i|c_i) = p(w_i|r(w_i)) = c(w_i) / c(r(w_i)) = c(w_i) / c(c_i)$ 
    - !!!  $c(w_i, c_i) = c(w_i)$  [since  $c_i$  determined by  $w_i$ ]
  - $p_k(c_i|h_i)$ :
    - $p_3(c_i|h_i) = p_3(c_i|c_{i-2}, c_{i-1}) = c(c_{i-2}, c_{i-1}, c_i) / c(c_{i-2}, c_{i-1})$
    - $p_2(c_i|h_i) = p_2(c_i|c_{i-1}) = c(c_{i-1}, c_i) / c(c_{i-1})$
    - $p_1(c_i|h_i) = p_1(c_i) = c(c_i) / |T|$
- Then smooth as usual
  - not the  $p(w_i|c_i)$  nor  $p_k(c_i|h_i)$  individually, but the  $p_k(w_i|h_i)$

# Classes: How To Get Them

- We supposed the classes are given
- Maybe there are in (human) dictionaries, but...
  - dictionaries are incomplete
  - dictionaries are unreliable
  - do not define classes as equivalence relation (overlap)
  - do not define classes suitable for LM
    - small, short... maybe; small and difficult?

→ we have to construct them from data (again...)

# Creating the Word-to-Class Map (Brown's Classes)

- Consider bigram model for now.
- Bigram estimate:

$$p_2(c_i|h_i) = p_2(c_i|c_{i-1}) = c(c_{i-1}, c_i) / c(c_{i-1}) = c(r(w_{i-1}), r(w_i)) / c(r(w_{i-1}))$$

- Form of the model:
  - just raw bigram for now:
- Maximize over  $r$  (given  $r \rightarrow$  fixed  $p, p_2$ ):
  - define objective

$$L(r) = 1/|T| \sum_{i=1..|T|} \log(p(w_i|r(w_i)) p_2(r(w_i))|r(w_{i-1})))$$

$$r_{\text{best}} = \operatorname{argmax}_r L(r) \quad (L(r) = \text{norm. logprob of training data ... as usual})$$

# Simplifying the Objective Function

- Start from  $L(r) = 1/|T| \sum_{i=1..|T|} \log(p(w_i|r(w_i)) p_2(r(w_i)|r(w_{i-1})))$ :  
 $1/|T| \sum_{i=1..|T|} \log(p(w_i|r(w_i)) \cancel{p(r(w_i))} p_2(r(w_i)|r(w_{i-1})) / \cancel{p(r(w_i))}) =$   
 $1/|T| \sum_{i=1..|T|} \log(\cancel{p(w_i,r(w_i))} p_2(r(w_i)|r(w_{i-1})) / p(r(w_i))) =$   
 $1/|T| \sum_{i=1..|T|} \log(\cancel{p(w_i)}) + 1/|T| \sum_{i=1..|T|} \log(p_2(r(w_i)|r(w_{i-1})) / p(r(w_i))) =$   
 $-H(W) + 1/|T| \sum_{i=1..|T|} \log(p_2(r(w_i)|r(w_{i-1})) \cancel{p(r(w_{i-1}))} / (\cancel{p(r(w_{i-1}))} p(r(w_i))))$   
=
- $-H(W) + 1/|T| \sum_{i=1..|T|} \log(\cancel{p(r(w_i),r(w_{i-1}))} / (p(r(w_{i-1})) p(r(w_i)))) =$   
 $-H(W) + \sum_{d,e \in C} p(d,e) \log( p(d,e) / (p(d) p(e)) ) =$   
 $-H(W) + I(D,E)$       (event E picks class adjacent (to the right) to the one picked by D)
- Since W does not depend on r, we ended up with  $I(D,E)$ .

# Maximizing Mutual Information (dependent on mapping r)

- Result from previous slide:
  - Maximizing the probability of data amounts to maximizing  $I(D, E)$ , the Mutual Information of the adjacent classes.
- Good:
  - We know what a MI is, and we know how to maximize.
- Bad:
  - There is no way how to maximize over so many possible partitionings:  $|V|^{|V|}$  - no way to test them all.

# Training or Heldout?

- Training:
  - best  $I(D, E)$ : all words in a class of its own  
→ will not give us anything new.
- Heldout: ok, but:
  - must smooth to test any possible partitioning (unfeasible):  
→ using raw model: 0 probability of heldout (almost) guaranteed  
→ will not be able to compare anything
- Solution:
  - use training anyway, but only keep  $I(D, E)$  as large as possible

# The Greedy Algorithm

- Define merging operation on the mapping  $r: V \rightarrow C$ :
  - merge:  $R \times C \times C \rightarrow R' \times C-1: (r,k,l) \rightarrow r',C'$  such that
  - $C^{-1} = \{C - \{k,l\} \cup \{m\}\}$  (throw out k and l, add new m  $\notin C$ )
  - $r'(w) = m \quad \text{for } w \in r_{INV}(\{k,l\}),$   
 $r(w) \quad \text{otherwise.}$
- 1. Start with each word in its own class ( $C = V$ ),  $r = \text{id}$ .
- 2. Merge two classes  $k,l$  into one,  $m$ , such that  
$$(k,l) = \operatorname{argmax}_{k,l} I_{\text{merge}(r,k,l)}(D,E).$$
- 3. Set new  $(r,C) = \text{merge}(r,k,l)$ .
- 4. Repeat 2 and 3 until  $|C|$  reaches predetermined size.

# Brown's Classes: Programming Tips & Tricks

# Complexity Issues

Still too complex:

- $|V|$  iterations of the steps 2 and 3.
- $|V|^2$  steps to maximize  $\text{argmax}_{k,l}$  (selecting  $k,l$  freely from  $|C|$ , which is in the order of  $|V|^2$ )
- $|V|^2$  steps to compute  $I(D,E)$  (sum within sum, all classes, also: includes log)  
⇒ total:  $|V|^5$
- i.e., for  $|V| = 100$ , about  $10^{10}$  steps (several hours!)
- but  $|V| \sim 50,000$  or more

# Formula breakdown

- Mutual Information at  $k^{\text{th}}$  iteration (=  $k$  classes):
  - $I_k = \sum_{l,r \in C} p_k(l,r) \log(p_k(l,r) / (p_{kl}(l) p_{kr}(r)))$
- For each pair of classes at iteration  $k$ , we define:
  - $q_k(l,r) = p_k(l,r) \log(p_k(l,r) / (p_{kl}(l) p_{kr}(r)))$
- So:
  - $I_k = \sum_{l,r \in C} q_k(l,r)$
- $q_k(l,r)$  using bigram counts  $c_k(l,r)$ :  
$$q_k(l,r) = c_k(l,r)/N \log(N c_k(l,r) / (c_{kl}(l) c_{kr}(r)))$$

$l \setminus r$	$c_1$	$c_2$	$c_3$	$c_4$
$c_1$	10	2	0	1
$c_2$	0	0	5	2
$c_3$	0	2	0	3
$c_4$	2	3	0	0

unigram/marginal counts

# Trick #1: Recomputing MI the Smart Way

- For test-merging  $c_2$  and  $c_4$  we recompute only rows/columns 2 & 4:

k:  $l \setminus r$  |  $c_1$  |  $c_2$  |  $c_3$  |  $c_4$

	$c_1$	$c_2$	$c_3$	$c_4$
$c_1$	10	2	0	1
$c_2$	0	0	5	2
$c_3$	0	2	0	3
$c_4$	2	3	0	0

The diagram shows a 5x5 matrix representing a confusion table. The columns are labeled  $l \setminus r$ ,  $c_1$ ,  $c_2$ ,  $c_3$ , and  $c_4$ . The rows are labeled  $c_1$ ,  $c_2$ ,  $c_3$ , and  $c_4$ . The matrix contains the following values:  
Row  $c_1$ : 10, 2, 0, 1  
Row  $c_2$ : 0, 0, 5, 2  
Row  $c_3$ : 0, 2, 0, 3  
Row  $c_4$ : 2, 3, 0, 0  
Arrows from the bottom of each row  $c_2$ ,  $c_3$ , and  $c_4$  point to the right edge of the matrix, indicating that only these columns need to be recomputed during the merge process.

# Trick #1: Recomputing MI the Smart Way

- For test-merging  $c_2$  and  $c_4$  we recompute only rows/columns 2 & 4:
  - Subtract column/row (2 & 4) from the MI sum:

1 \ r	$c_1$	$c_2$	$c_3$	$c_4$
$c_1$	10	2	0	1
$c_2$	0	0	5	2
$c_3$	0	2	0	3
$c_4$	2	3	0	0

The diagram illustrates the step of subtracting the total value of row  $c_2$  (which is 5) and column  $c_4$  (which is 1) from the corresponding entries in the matrix. Arrows point from the row total 5 and column total 1 to their respective counterparts in the matrix. Specifically, the value 2 in the  $c_2, c_4$  cell is being adjusted.

# Trick #1: Recomputing MI the Smart Way

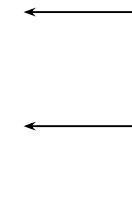
- For test-merging  $c_2$  and  $c_4$  we recompute only rows/columns 2 & 4:

- Subtract column/row (2 & 4) from the MI sum:

(be careful at the intersections)

k:

1 \ r	$c_1$	$c_2$	$c_3$	$c_4$
$c_1$	10	2	0	1
$c_2$	0	0	5	2
$c_3$	0	2	0	3
$c_4$	2	3	0	0



# Trick #1: Recomputing MI the Smart Way

- For test-merging  $c_2$  and  $c_4$  we recompute only rows/columns 2 & 4:

- Subtract column/row (2 & 4) from the MI sum:

(be careful at the intersections)

1 \ r	$c_1$	$c_2$	$c_3$	$c_4$
$c_1$	10	2	0	1
$c_2$	0	0	5	2
$c_3$	0	2	0	3
$c_4$	2	3	0	0

↑                              ↑

- Add sums of the merged counts (row & column for ( $c_2'$  is the merged class):

(watch the intersection again)

1 \ r	$c_1$	$c_2'$	$c_3$
$c_1$	10	3	0
$c_2'$	2	5	5
$c_3$	0	5	0

↑

## Trick #2: Precompute the Counts-to-be-Subtracted

- Summing loop goes through  $i, j$ 
  - ... but the single row/column sums do not depend on the (resulting sums after the) merge  $\Rightarrow$  can be precomputed
  - only  $2k$  logs to compute at each algorithm iteration, instead of  $k^2$
- Then for each “merge-to-be” compute only add-on sums, plus “intersection adjustment”

# Formulas for Tricks #1 and #2

- Recap:

$$q_k(l,r) = p_k(l,r) \log(p_k(l,r) / (p_{kl}(l) p_{kr}(r)))$$

the same, but using counts:

$$q_k(l,r) = c_k(l,r)/N \log(N c_k(l,r)/(c_{kl}(l) c_{kr}(r)))$$

- Define further (row+column a sum):

$$s_k(a) = \sum_{l=1..k} q_k(l,a) + \sum_{r=1..k} q_k(a,r) - q_k(a,a)$$

- Then, the subtraction part of Trick #1 amounts to

$$\text{sub}_k(a,b) = s_k(a) + s_k(b) - q_k(a,b) - q_k(b,a)$$

precomputed

Intersection adjustment

	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>
c <sub>2</sub>	0	5	2
c <sub>3</sub>	2	0	3
c <sub>4</sub>	3	0	0

remaining intersection adjustment

## Formulas - cont.

- After-merge add-on:

$$\text{add}_k(a,b) = \sum_{l=1..k, l \neq a,b} q_k(l,a+b) + \sum_{r=1..k, r \neq a,b} q_k(a+b,r) + q_k(a+b,a+b)$$

- What is it a+b? Answer: the new (merged) class.
- Hint: use the definition of  $q_k$  as a “macro”, and then

$$p_k(a+b,r) = p_k(a,r) + p_k(b,r) \quad (\text{same for other sums, equivalent})$$

- The above sums cannot be precomputed
- Mutual Information after merge of class a,b:
  - $I_{k-1}(a,b) = I_k - \text{sub}_k(a,b) + \text{add}_k(a,b)$
  - $I_k$  is the “old” MI, kept from previous iteration of the algorithm

## Trick #3: Ignore Zero Counts

- Many bigrams are 0
  - e.g. in the Canadian Hansards corpus, < 0.1 % of bigrams are non-zero)
- Create linked lists of non-zero counts in columns and rows
  - similar effect: use hashes (store non-zero-count bigrams)
- Update links after merge (after step 3)

## Trick #4: Use Updated Loss of MI

- We are now down to  $|V|^4$ :  $|V|$  merges, each merge takes  $|V|^2$  “test-merges”, each test-merge involves order-of- $|V|$  operations ( $\text{add}_k(i,j)$  term, slide 34)
- Observation:
  - many numbers ( $s_k, q_k$ ) needed to compute the mutual information loss due to a merge of  $i+j$  **do not change**: namely, those which are not in the vicinity of either  $i$  nor  $j$ .
- Idea:
  - keep the MI loss matrix for all pairs of classes, and (after a merge) update only those cells which have been influenced by the merge.

## Formulas for Trick #4 ( $s_{k-1}, L_{k-1}$ )

- Keep a matrix of "losses"  $L_k(d,e)$  [symmetry:  $L_k(d,e) = L_k(e,d)$ ]
- Init:  $L_k(d,e) = \text{sub}_k(d,e) - \text{add}_k(d,e)$  [then  $I_{k-1}(d,e) = I_k - L_k(d,e)$ ]
- Suppose a,b are now the two classes merged into a
- Update ( $k-1$ : index used for the next iteration;  $i,j \neq a,b$ ):

$$s_{k-1}(i) = s_k(i) - q_k(i,a) - q_k(a,i) - q_k(i,b) - q_k(b,i) + q_{k-1}(a,i) + q_{k-1}(i,a)$$

$$\begin{aligned} L_{k-1}(i,j) = & L_k(i,j) - s_k(i) + s_{k-1}(i) - s_k(j) + s_{k-1}(j) + \\ & + q_k(i+j,a) + q_k(a,i+j) + q_k(i+j,b) + q_k(b,i+j) - \\ & - q_{k-1}(i+j,a) - q_{k-1}(a,i+j) \end{aligned}$$

## Completing Trick #4

- $s_{k-1}(a)$  must be computed using the “Init” sum (see the prev. slide).
- $L_{k-1}(a,i) = L_{k-1}(i,a)$  must be computed in a similar way, for all  $i \neq a,b$ .
- $s_{k-1}(b), L_{k-1}(b,i), L_{k-1}(i,b)$  are not needed anymore (keep track of such data, i.e. mark every class already merged into some other class and do not use it anymore).
- Keep track of the minimal loss during the  $L_k(i,j)$  update process (so that the next merge to be taken is obvious immediately after finishing the update step).

# Efficient Implementation

Data Structures: ( $N$  - # of bigrams in data [fixed])

- $\text{Hist}(k)$  - history of merges  
 $\text{Hist}(k) = (a,b)$  merged when the remaining number of classes was  $k$
- $c_k(i,j)$  - bigram class counts [updated after merge]
- $c_{kl}(i), c_{kr}(i)$  - unigram (marginal) counts [updated]
- $L_k(a,b)$  - table of losses; upper-right triangle [updated]
- $s_k(a)$  - “subtraction” subterms [optionally updated]
- $q_k(i,j)$  - subterms involving a log [optionally updated]

The optionally updated data structures will give linear improvement only in the subsequent steps, but at least  $s_k(i)$  is necessary in the initialization phase (1<sup>st</sup> iteration)

# Implementation: the Initialization Phase

1. Read data in, set  $k=|V|$ , init counts  $c_k(l,r)$ ; then  $\forall l,r,a,b; a < b$ :
2. Init unigram counts  $c_{kl}(l), c_{kr}(r)$ :

$$c_{kl}(l) = \sum_{r=1..k} c_k(l,r), \quad c_{kr}(r) = \sum_{l=1..k} c_k(l,r)$$

[must take care of start & end of data!]

3. Init  $q_k(l,r)$ : use the 2<sup>nd</sup> formula (count-based) on slide 27,

$$q_k(l,r) = c_k(l,r)/N \log(N c_k(l,r)/(c_{kl}(l) c_{kr}(r)))$$

4. Init  $s_k(a) = \sum_{l=1..k} q_k(l,a) + \sum_{r=1..k} q_k(a,r) - q_k(a,a)$
5. Init  $L_k(a,b) = s_k(a) + s_k(b) - q_k(a,b) - q_k(b,a) - q_k(a+b,a+b) +$   
 $- \sum_{l=1..k, l \neq a,b} q_k(l,a+b) - \sum_{r=1..k, r \neq a,b} q_k(a+b,r)$

# Implementation: Select & Update

6. Select the best pair  $(a,b)$  to merge into  $a$   
watch the candidates when computing  $L_k(a,b)$ ; save to  $\text{Hist}(k)$
7. Optionally, update  $q_k(i,j)$  for all  $i,j \neq b$ , get  $q_{k-1}(i,j)$   
remember those  $q_k(i,j)$  values needed for the updates below
8. Optionally, update  $s_k(i)$  for all  $i \neq b$ , to get  $s_{k-1}(i)$   
again, remember the  $s_k(i)$  values for the “loss table” update
9. Update the loss table,  $L_k(i,j)$ , to  $L_{k-1}(i,j)$ , using the tabulated  $q_k$ ,  
 $q_{k-1}$ ,  $s_k$  and  $s_{k-1}$  values, or compute the needed  $q_k(i,j)$  and  $q_{k-1}(i,j)$   
values dynamically from the counts:

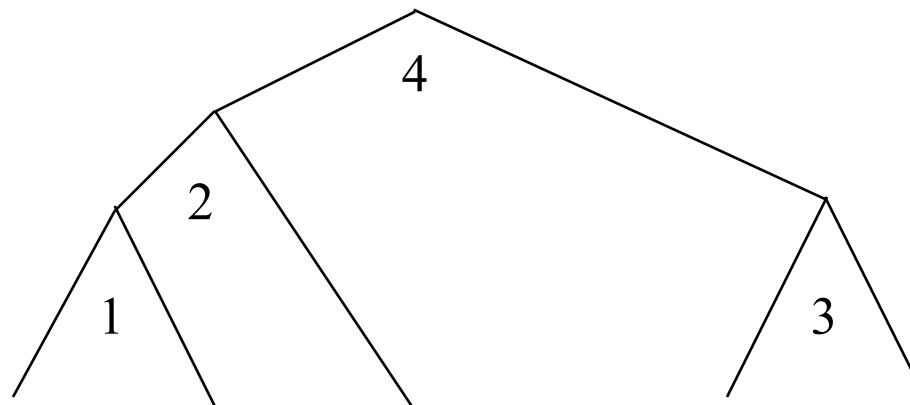
$$c_k(i+j,b) = c_k(i,b) + c_k(j,b); \quad c_{k-1}(a,i) = c_k(a+b,i)$$

## Towards the Next Iteration

10. During the  $L_k(i,j)$  update, keep track of the minimal loss of MI, and the two classes which caused it.
11. Remember such best merge in  $\text{Hist}(k)$ .
12. Get rid of all  $s_k$ ,  $q_k$ ,  $L_k$  values.
13. Set  $k = k - 1$ ; stop if  $k = 1$ .
14. Start the next iteration
  - either by the optional updates (steps 7 and 8), or
  - directly updating  $L_k(i,j)$  again (step 9).

# Using the Hierarchy

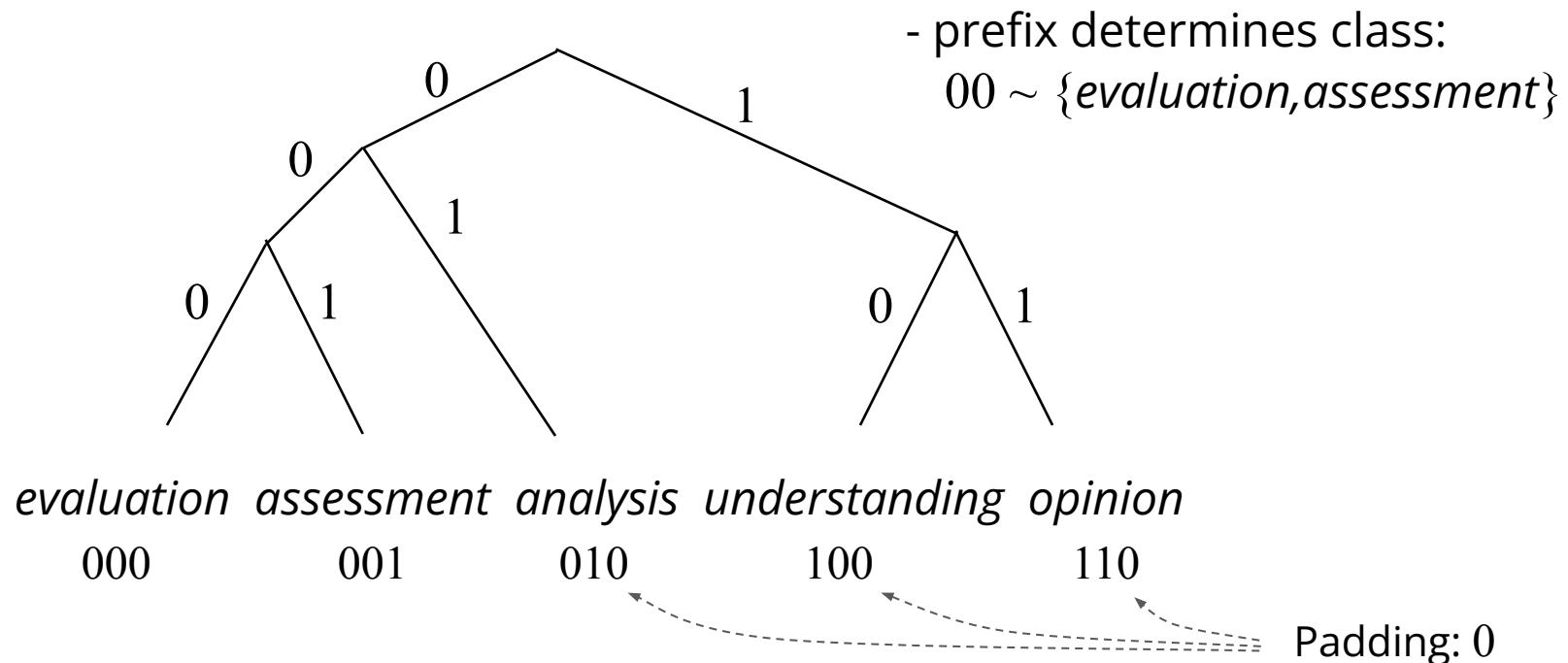
- Natural Form of Classes
- follows from the sequence of merges:



*evaluation assessment analysis understanding opinion*

# Numbering the Classes (within the Hierarchy)

- Binary branching
- Assign 0/1 to the left/right branch at every node:



# Word Classes in Applications

- Even in the era of neural embeddings, Brown classes have modern, practical applications (such as POS tagging)
- Especially useful in low-resource and domain-specific scenarios (tens of millions words)
- Provide compact, interpretable word classes that can replace sparse one-hot or n-gram features.

# Moodle Quiz

# Moodle Quiz



<https://dl1.cuni.cz/course/view.php?id=18547>