Hidden Markov Models

Rafael E. Banchs (rbanchs@gps.tsc.upc.edu)

Session 4: The EM algorithm & HMMs

Centre de Tecnologies i Aplicacions del Llenguatge i la Parla
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Exercise 6: A data-driven inference experience

Consider the twelve pairs of sentences provided, which correspond To translations between two hypothetical languages: A and B.

Try to infer correspondences among tokens from both languages, in order to:

- Build a bilingual dictionary of tokens
- Compute translations for the provided test data

Review on HMMs

Few days ago, we defined HMMs, and recalled that they can be formally specified by a set of five elements:

$$\{ S,\ O, \pi, \mathbf{P}, \mathbf{Q} \}$$

- where, S is the set of states: $S = \{s_1, s_2, s_3 \ldots s_m\}$;
- O is the observation alphabet: $O = \{o_1, o_2, o_3 \ldots o_h\}$;
- and, π **P** and **Q** are the HMM associated probabilities.

Review on HMMs

- State sequence: $\boldsymbol{X} = X_1 , X_2 , X_3 \ldots X_N$
- State alphabet (state set): $\boldsymbol{S} = \{s_1 , s_2 , s_3 \ldots s_m\}$
- Observation sequence: $\boldsymbol{Y} = Y_1 , Y_2 , Y_3 \ldots Y_N$
- Observation alphabet (symbols): $\boldsymbol{O} = \{o_1 , o_2 , o_3 \ldots o_h\}$
- Initial state probabilities: $\boldsymbol{\pi} = \{\pi_j\}$ with $j \in \boldsymbol{S}$
- State transition probabilities: $\mathbf{P} = \{p_{ij}\}$ with $i, j \in \boldsymbol{S}$
- Symbol emission probabilities: $\mathbf{Q} = \{q_{ijk}\}$ with $i, j \in \boldsymbol{S}$, $k \in \boldsymbol{O}$

Review on HMMs

Last day we studied the problem of estimating the most probable state sequence $\boldsymbol{X} = X_1, X_2, X_3 \ldots X_N$ that explains an observation sequence $\boldsymbol{Y} = Y_1, Y_2, Y_3 \ldots Y_N$ for a given HMM $\lambda = \{\pi, \mathbf{P}, \mathbf{Q}\}$

$$\underset{X}{Argmax}\, P(X \mid Y, \lambda) \quad = \quad \underset{X}{Argmax}\, P(Y \mid X, \lambda)\, P(X \mid \lambda)$$

This problem constitutes actually an optimization problem and it can be efficiently solved by using the ***Viterbi algorithm.***

Estimating HMM parameters: Model Training

The problem we are going to deal with today is the following:

- Consider a HMM defined as $\lambda = \{\pi, \mathbf{P}, \mathbf{Q}\}$, what would be the best model parameters: π, **P** and **Q** that explain an observation data set?
- Such data set is frequently called the training data set and it may contain information on observations only, or also on states.
- There are actually two different formulations for this problem depending on the available training data.

Mathematical formulation of the problem

The two different formulations are the following:

- The "easy" problem: $\hat{\lambda} = Argmax\ P(X,Y \mid \lambda)$

  training data

  model space

  when training data includes information about both state sequences and their corresponding observation sequences.

- The "difficult" problem: $\hat{\lambda} = Argmax\ P(Y \mid \lambda)$

  training data

  model space

  when training data only includes information about observation sequences.

Estimating parameters from states and observations

In the "easy" problem formulation, parameters are estimated just by counting from training data... So, for a given state sequence $X_0, X_1 ... X_N$ and its corresponding observation sequence $Y_1, Y_2 ... Y_n$

- $\hat{\pi} = \delta(X_0)$, i.e. only the element at index position X_0 is equal to *1*
- $\hat{p}_{ij} = \dfrac{total\ number\ of\ transitions\ from\ X_i\ to\ X_j}{total\ number\ of\ transitions\ from\ X_i}$
- $\hat{q}_{ijO_k} = \dfrac{total\ number\ of\ transitions\ from\ X_i\ to\ X_j\ emitting\ O_k}{total\ number\ of\ transitions\ from\ X_i\ to\ X_j}$

Exercise 2 revisited

Consider the following sequences of states and observations for the HMM of exercise 2:

States

123331233312231112312312312231122233323231223123332231223122312223223123123331223123122312222231222222322222331123123112 31
22222222312322223332222223312312312222333123223231222223312312223331222233312323311232222233333223122312323312311223222222231
12222231222312312331222311223322312312231233112223112223312312231231123322332323233123112232222322331223231223123122223312
23233122222223122312222231222311232331222231222222311231122322312331232233112323122312223123123312333332322222231123123222
2231231232223332311222311231231222312311123122323311222231222222233122222231122222322231222222231222222233333 1232

Observations

BBAABBAAABAABAAABBBBBABAAAABBABAABAABAABBAAAABAAAAABABABABAAABAAABAAABBBBAAABAAAABAABAA
ABAAAABABAAAAAABAAAAAAAAAAABAABBABBABAAAAAAAAABBAAAABAABAAAAABABBBBAABAAAAAAAAABABABBABAAAA
AAABAABBAAAAABBAAAAAAAABABAAAABBBAAAABAAAABAABBAABBABAABBAAAAAAABAAAAABAABAAAABBBAABBBABA
BAABAAABABAAABAAAABBBAABBAABAAAABBABAABAABBBBABBAABAABAAABABABBAAABBABAAAABAAABAAAAABAA
AABBABAAAAAAAAAAABAABBBAAAAAAAAAABBAAAABAAAABABAAAAAABAABAAAAAAABABAAAAAABBABBAAAAABABBABAA
AAAAAAAAABBBBAAABBAAAAABAABABAAAAAAAABBAAAAABAAABBABBAAAABABAAABAAABAABAAABAAABAAABAABBA
ABABABAABBAAAAABAAAAAAAAAABAAAAAABAABAAAAAAAAABAAAABBAAAAAAAAAAAABBAAAAAAAABAAAABAAA

Exercise 2 revisited

- Since the initial state X_0 is state S_1, then: $\hat{\pi} = [1\ 0\ 0]$

- Considering the state sequence, the total counts for state transitions are given by:

$$\mathbf{P}_{counts} = \begin{pmatrix} 24 & 96 & 0 \\ 0 & 167 & 130 \\ 95 & 35 & 53 \end{pmatrix}$$

- Normalizing $\mathbf{P}_{counts}$ so that its row sums are equal to *1*:

$$\hat{\mathbf{P}} = \begin{pmatrix} 0.2000 & 0.8000 & 0 \\ 0 & 0.5623 & 0.4377 \\ 0.5191 & 0.1913 & 0.2896 \end{pmatrix}$$

Exercise 2 revisited

- Considering both the state and observation sequences, the total state-transition counts for each observation symbol are:

$$\mathbf{Q}_{\mathbf{A}\,counts} = \begin{pmatrix} 24 & 41 & 0 \\ 0 & 167 & 68 \\ 48 & 13 & 53 \end{pmatrix} \qquad \mathbf{Q}_{\mathbf{B}\,counts} = \begin{pmatrix} 0 & 55 & 0 \\ 0 & 0 & 62 \\ 47 & 22 & 0 \end{pmatrix}$$

- Normalizing each of these counts with respect to the total number of transitions for each pair of states (i.e. $\mathbf{P}_{counts}$)

$$\hat{\mathbf{Q}}_{\mathbf{A}} = \begin{pmatrix} 1.0000 & 0.4271 & NaN \\ NaN & 1.0000 & 0.5231 \\ 0.5053 & 0.3714 & 1.0000 \end{pmatrix} \qquad \hat{\mathbf{Q}}_{\mathbf{B}} = \begin{pmatrix} 0 & 0.5729 & NaN \\ NaN & 0 & 0.4769 \\ 0.4947 & 0.6286 & 0 \end{pmatrix}$$

Exercise 2 revisited

So, the resulting estimates for $\lambda = \{\pi, \mathbf{P}, \mathbf{Q}\}$, are:

$$\hat{\pi} = [1\ 0\ 0] \qquad \hat{\mathbf{P}} = \begin{pmatrix} 0.2000 & 0.8000 & 0 \\ 0 & 0.5623 & 0.4377 \\ 0.5191 & 0.1913 & 0.2896 \end{pmatrix}$$

$$\hat{\mathbf{Q}}_A = \begin{pmatrix} 1.0000 & 0.4271 & 0.5000 \\ 0.5000 & 1.0000 & 0.5231 \\ 0.5053 & 0.3714 & 1.0000 \end{pmatrix} \qquad \hat{\mathbf{Q}}_B = \begin{pmatrix} 0 & 0.5729 & 0.5000 \\ 0.5000 & 0 & 0.4769 \\ 0.4947 & 0.6286 & 0 \end{pmatrix}$$

Which happen to match very well the actual HMM parameters...

Estimating parameters from observations only

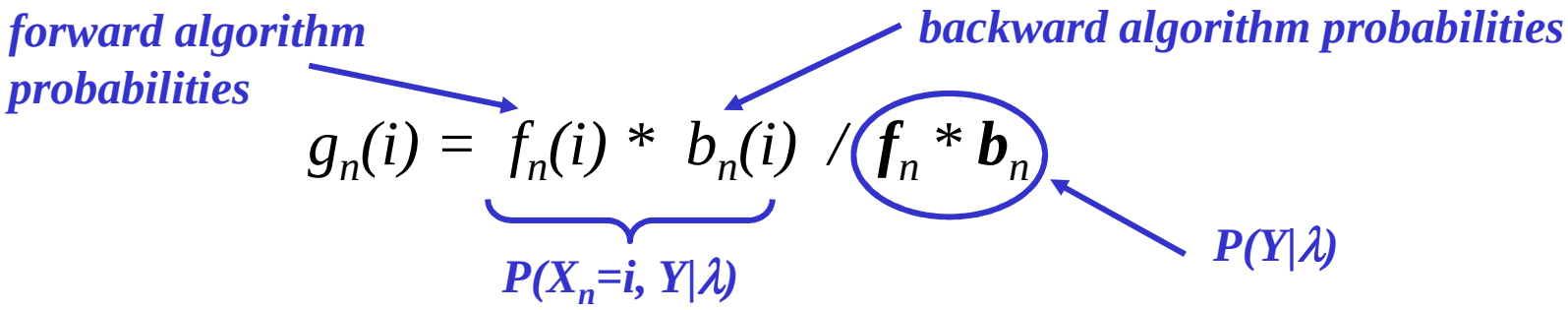
In the "difficult" problem formulation, the state sequences that generate the observations are unknown...

- For this problem, we are interested in maximizing $P(Y \mid \lambda)$ over the model space λ, without caring about specific state sequences.
- This problem cannot be solved analytically, but suboptimal solutions can be computed in an iterative fashion.
- Local optimizations can be performed by using the ***Baum-Welch*** algorithm, that is a special case of ***Expectation Maximization.***

The Baum-Welch algorithm

- Let us define the vector $\boldsymbol{g}_n$ which contains the probabilities of being at each state at step n for a given observation sequence $\boldsymbol{Y}$ (It is a column vector of M elements)

$$g_n(i) = P(X_n = i \mid \boldsymbol{Y}, \lambda) \quad for \;\; i=1, 2 \ldots M$$



The Baum-Welch algorithm

- Let us define the matrix $\mathbf{W}_n$ which contains the probabilities of moving from actual state *i* to next state *j* at time step *n* for a given observation sequence $\boldsymbol{Y}$. (It is an $M \times M$ matrix)

$$w_n(i,j) = P(X_n=i, X_{n+1}=j \mid \boldsymbol{Y}, \lambda) \quad for \;\; i, j = 1, 2 \ldots M$$

$$w_n(i) = \underbrace{f_n(i) * p_{ij} * q_{ij_{Y_n}} * b_{n+1}(j)}_{P(X_n=i,\, X_{n+1}=j,\, Y|\lambda)} \; / \; \boldsymbol{f}_n * \boldsymbol{b}_n$$

$\boldsymbol{f}_n * \boldsymbol{b}_n$: $P(Y|\lambda)$

The Baum-Welch algorithm

- So, starting from an initial model $\lambda = \{\pi, \mathbf{P}, \mathbf{Q}\}$, an updated model can be computed as follows:

$$\hat{\pi} = \boldsymbol{g}_0$$

transition "counts" **normalization**

$$\hat{p}_{ij} = \frac{\textit{expected transitions from } X_i \textit{ to } X_j}{\textit{expected transitions from } X_i} = \frac{\sum_n w_n(i,j)}{\sum_n g_n(i)}$$

transition "counts" for a given observation

$$\hat{q}_{ij\,O_k} = \frac{\textit{expected transitions from } X_i \textit{ to } X_j \textit{ emitting } O_k}{\textit{expected transitions from } X_i \textit{ to } X_j} = \frac{\sum_{n O_k} w_n(i,j)}{\sum_n w_n(i,j)}$$

The Baum-Welch algorithm

- By using the updated model $\hat{\lambda}$, new values for $\boldsymbol{f}_n$, $\boldsymbol{b}_n$, $\boldsymbol{g}_n$ and $\mathbf{W}_n$ can be computed, from which a new model update can be performed, and so on...

- This re-estimation process is guaranteed to improve $P(\mathbf{Y}|\lambda)$ so that at each new re-estimation $P(\mathbf{Y}|\lambda_{new}) \geq P(\mathbf{Y}|\lambda_{previous})$

- However, the procedure gets easily trapped in suboptimal solutions which strongly depend on the initial model parameters.

Exercise 2 revisited again

Now let us consider the previous observation sequence only:

Observations

BBAABBAAABAABAAABBBBBABAAAABBABAABAABAABBAAAABAAAAABABABABAAABAAABAAABBBBAAABAAAABAABAA
ABAAAABABAAAAAABAAAAAAAAAABAABBABBABAAAAAAAABBAAAABAABAAAABABBBBAABAAAAAAAAABABABBABAAAA
AAABAABBAAAAABBAAAAAAABABAAAABBBAAAABAAAABAABBAABBABAABBAAAAAABAAAAABAABAAAABBBAABBBABA
BAABAAABABAAABAAAABBBAABBAABAAAABBABAABAABBBBABBAABAABAAABABABBAAABBABAAAABAAABAAAAABAA
AABBABAAAAAAAAAAABAABBBAAAAAAAAABBAAAABAAAABABAAAAABAABAAAAAABABAAAAABBABBAAAAABABBABAA
AAAAAAAAABBBBAAABBAAAAABAABABAAAAAAABBAAAAABAAABBABBAAAABABAAABAAABAABAAABAAABAAABAABBA
ABABABAABBAAAAABAAAAAAAAABAAAAABAABAAAAAAAABAAAABBAAAAAAAAAAABBAAAAAAABAAAABAAA

And we also need a first guess for the model λ:

- Let us consider the following ones:

$$\pi = [1 \;\; 0 \;\; 0], \quad \text{and} \quad p_{ij} = 1/3, \; q_{ijA} = q_{ijB} = 1/2 \quad \text{for} \quad i, j = 1,2,3$$

Exercise 2 revisited again

After a hundred iterations of the *Baum-Welch* algorithm we reach the following model values:

$$\hat{\pi} = [1.0000 \;\; 0 \;\; 0]$$

$$\hat{\mathbf{P}} = \begin{pmatrix} 0.3990 & 0.3005 & 0.3005 \\ 0.3088 & 0.3456 & 0.3456 \\ 0.3088 & 0.3456 & 0.3456 \end{pmatrix}$$

$$\hat{\mathbf{Q}}_A = \begin{pmatrix} 0.9385 & 0.8095 & 0.8095 \\ 0.7516 & 0.6042 & 0.6042 \\ 0.7516 & 0.6042 & 0.6042 \end{pmatrix}$$

$$\hat{\mathbf{Q}}_B = \begin{pmatrix} 0.0615 & 0.1905 & 0.1905 \\ 0.2484 & 0.3958 & 0.3958 \\ 0.2484 & 0.3958 & 0.3958 \end{pmatrix}$$

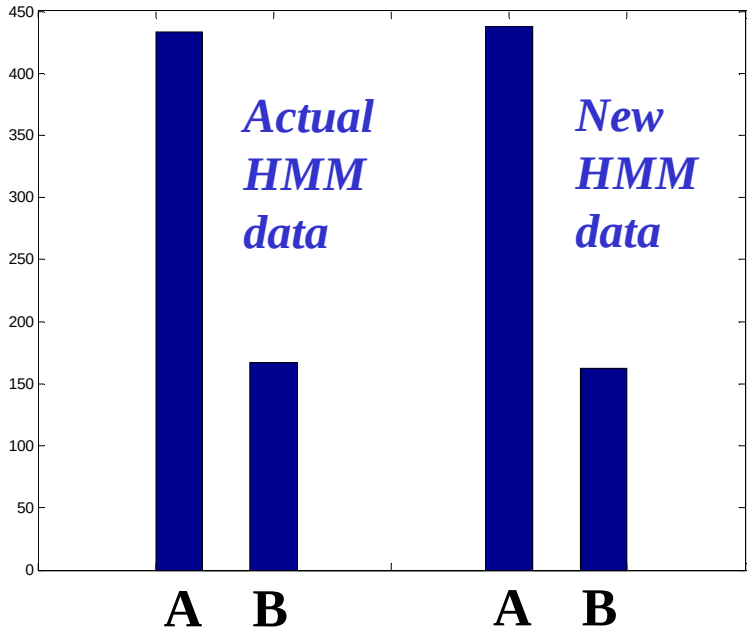
Note that these values have nothing to do with the actual model parameters... ***However they do represent an equivalent HMM !!!***

Exercise 2 revisited again

Consider now the state sequence that was used with the original HMM for generating the training observation sequence.

If we use the same state sequence to generate observations with the new estimated HMM parameters, we should reproduce (statistically) the original observation sequence.

These are the resulting histograms..



Comments on exercise 6

The data in exercise 6 are actually Chinese and English sentences.

A common procedure used in machine translation and other NLP applications is the so called word-to-word alignment, which consists of inferring word correspondences between two languages.

This procedure can be implemented by means of HMMs as it is described in Vogel *et.al.* (1999).

However, in practice HMMs are combined with other models in order to achieve better results.

HMM-based word-to-word alignment model

The model can be represented as:

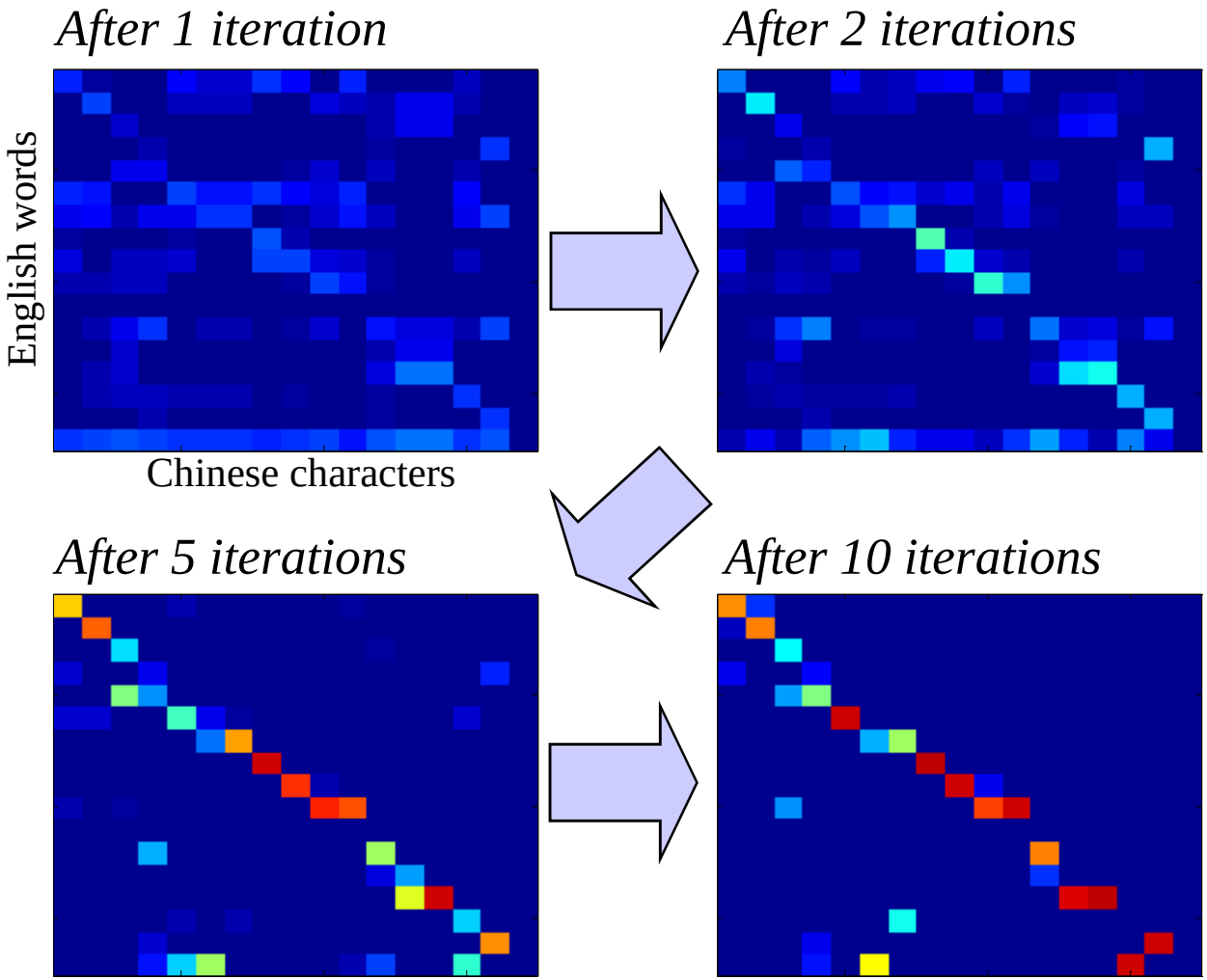
Probability of the Chinese sentence ZH given the English sentence EN

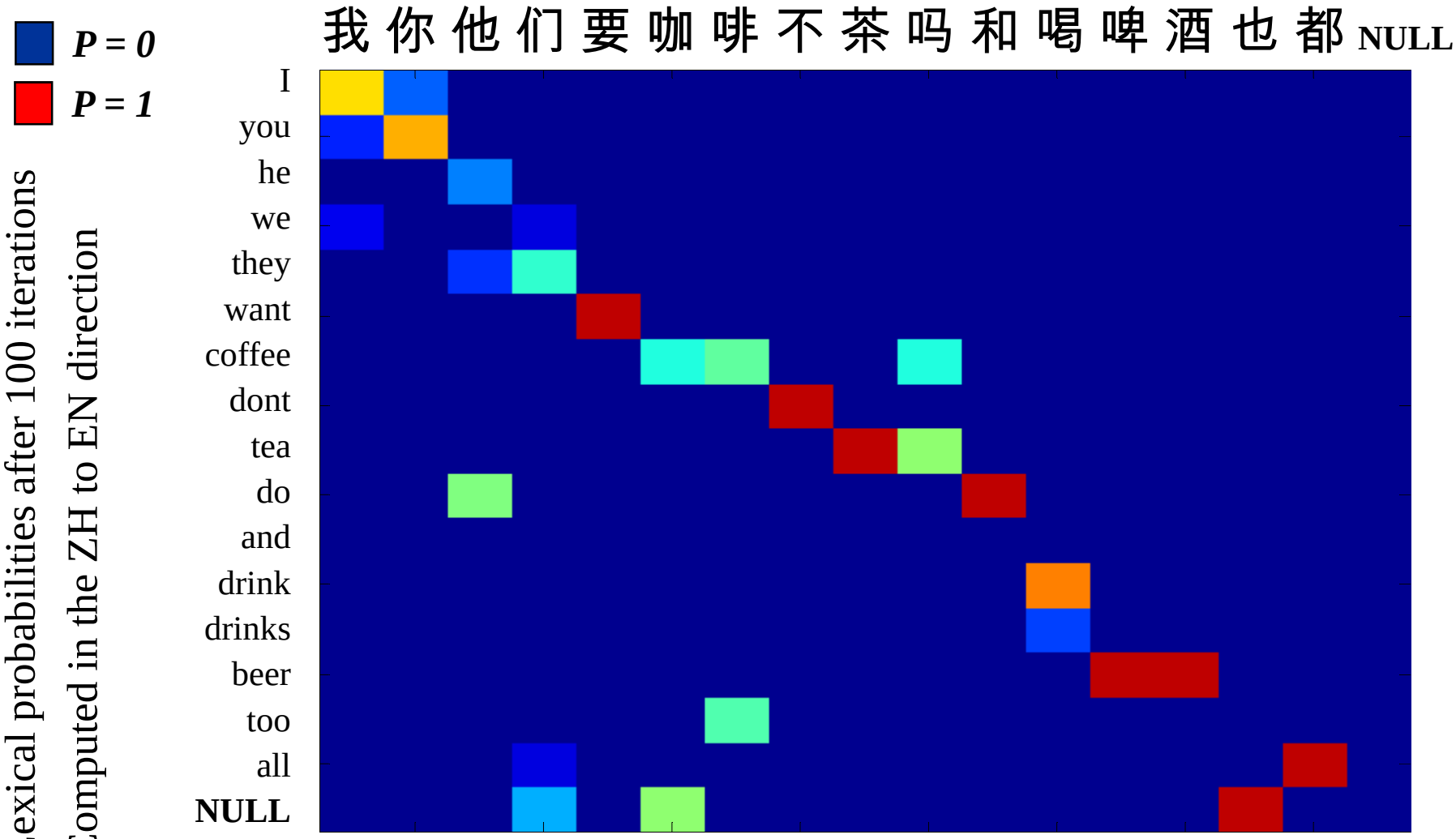
$$P(ZH \mid EN) = \sum_{a_j} \prod_{j} p(a_j \mid a_{j-1})\, p(zh_j \mid en_{aj})$$

Alignment probability:
Probability of the j^{th} Chinese word to be linked to the a_j^{th} English word, given that the $(j-1)^{th}$ Chinese word is linked to the a_{j-1}^{th} English word

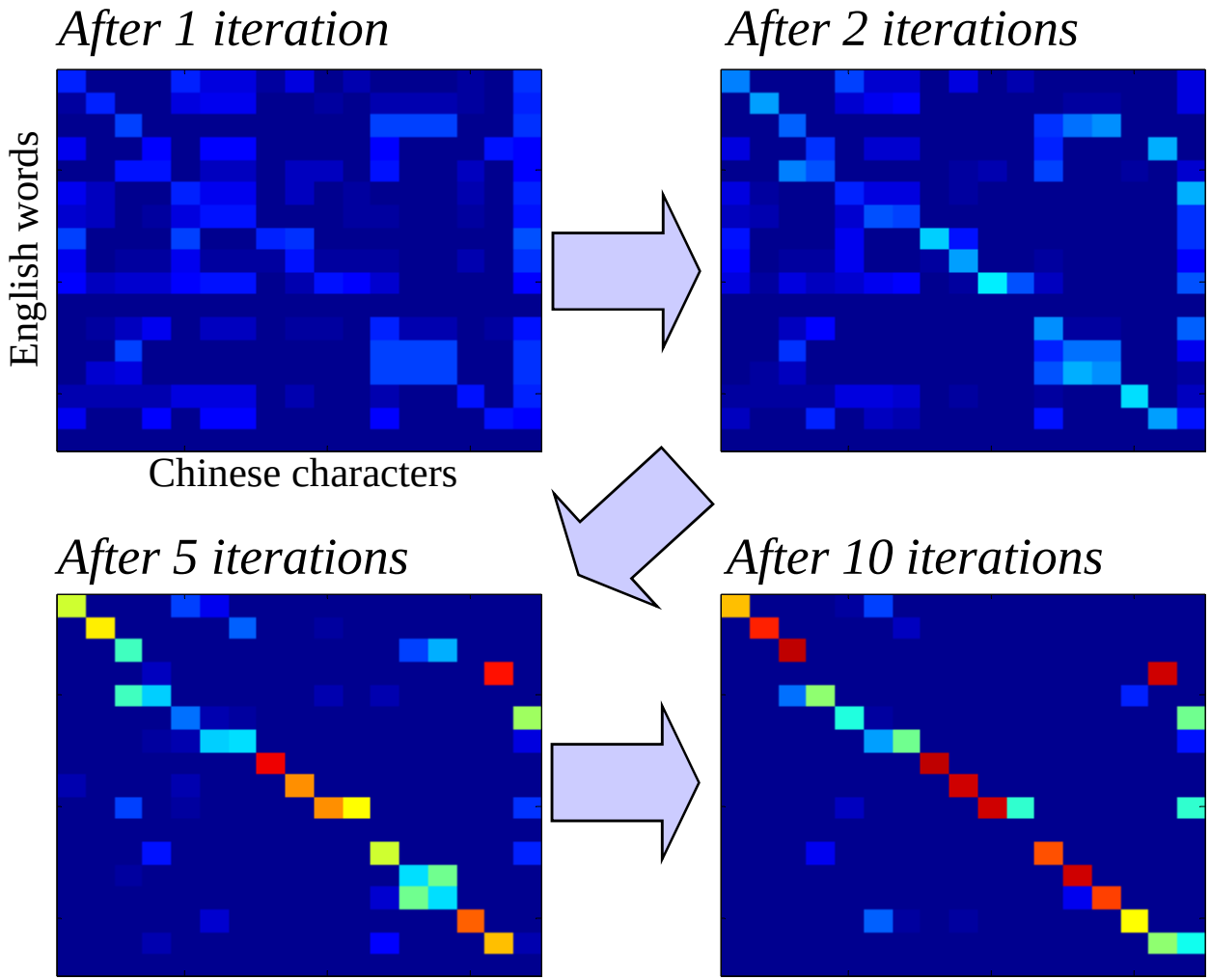
Lexical Probability:
Probability of the j^{th} Chinese word given the a_j^{th} English word

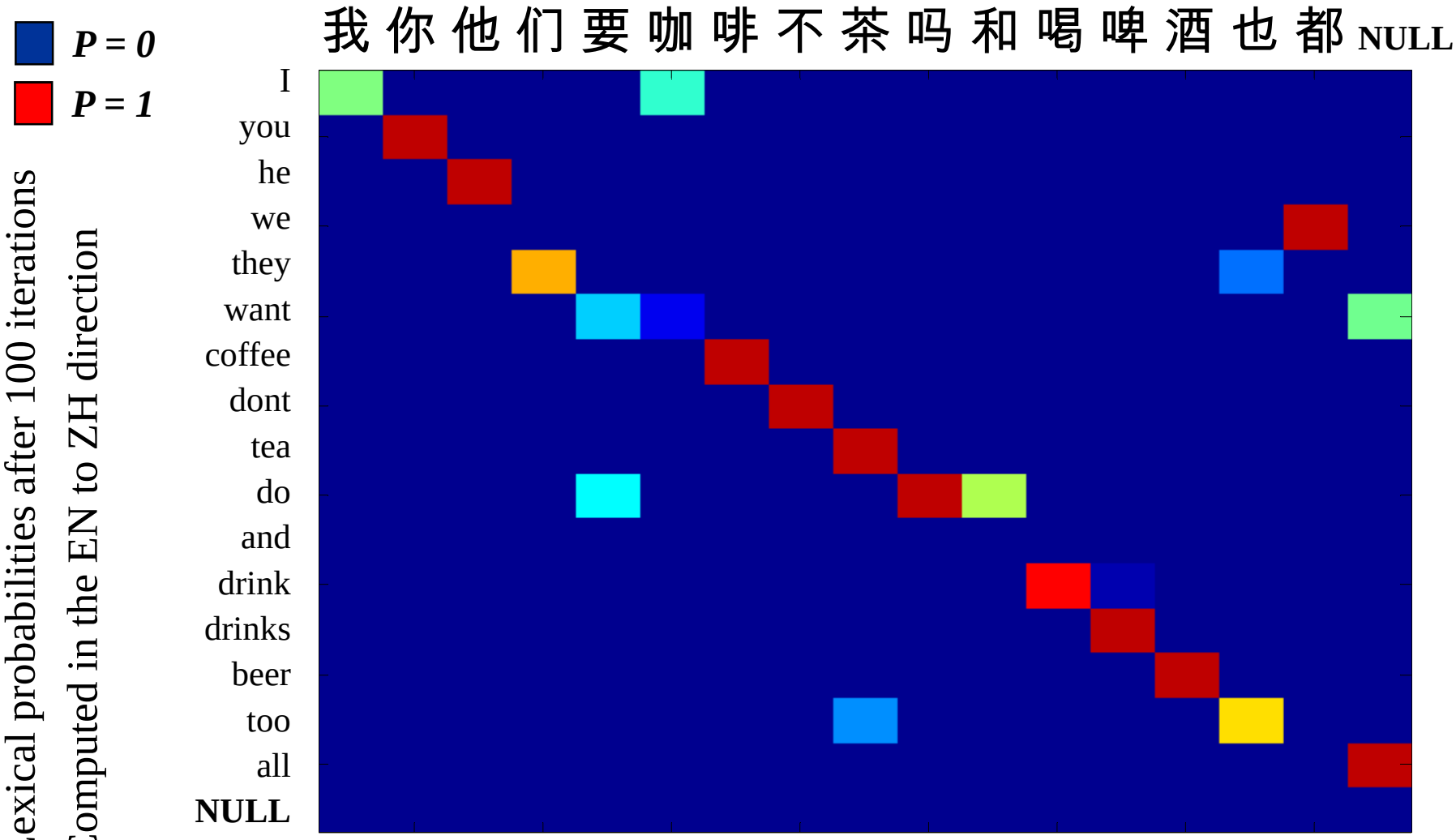
Lexical probabilities estimated in the ZH to EN direction





Lexical probabilities estimated in the EN to ZH direction



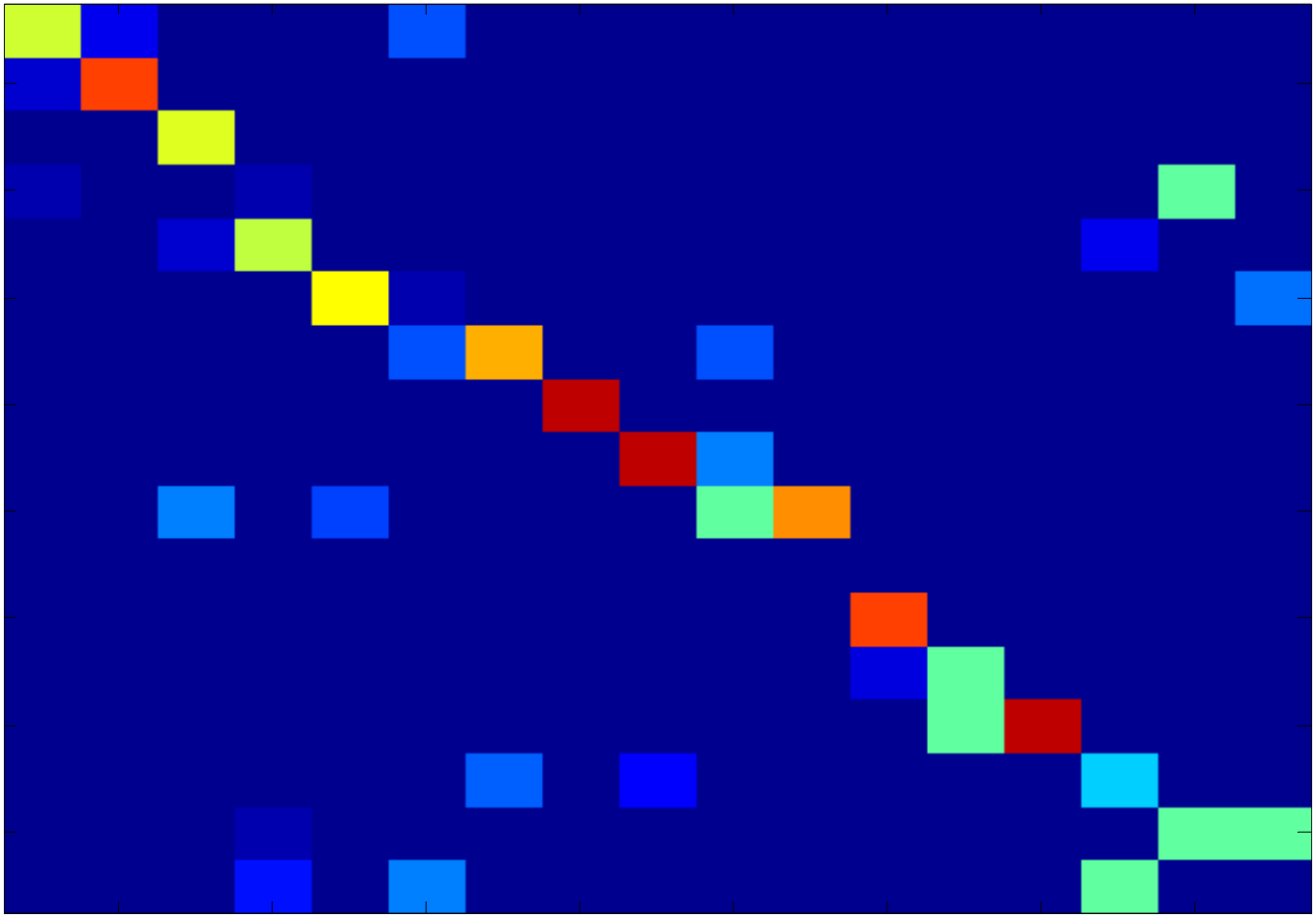


P = 0

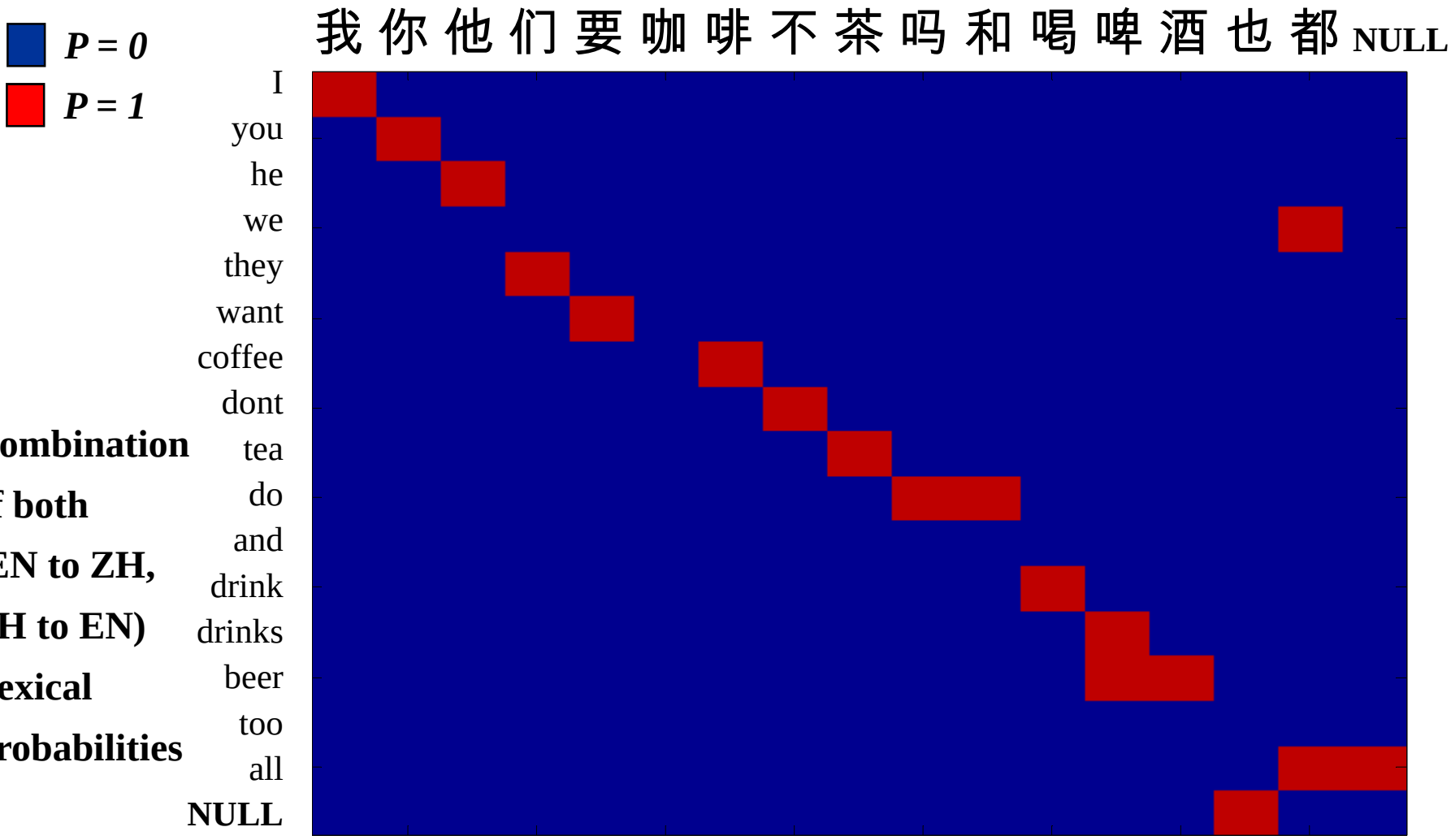
P = 1

我 你 他 们 要 咖 啡 不 茶 吗 和 喝 啤 酒 也 都 NULL

I
you
he
we
they
want
coffee
dont
tea
do
and
drink
drinks
beer
too
all
NULL



Combination of both (EN to ZH, ZH to EN) Lexical Probabilities



Combination of both (EN to ZH, ZH to EN) Lexical Probabilities

Exercise 7

Consider again the HMM of exercise 2...

1.- Use MATLAB or any other language to implement the *Baum-Welch* algorithm.

2.- Generate a random sequence of *500* states and observations by using the HMM parameters of exercise 2

3.- Estimate some different models from the observation sequence generated in 2.- by considering different initial models λ_0.

Important information about Exercise 7...

- If you already did the programming related to exercises 3: the forward and backward procedures...
- and you also did the programming related to exercise 5: Viterbi algorithm...

Programming exercise 7 should be very easy, and then...

YOU HAVE ALMOST READY THE FINAL PROJECT !!!

You only need an application. If interested we can talk now...

Important information about Exercise 7...

Some possible applications are:

- communication channel equalization,
- convolutional code decoding,
- part of speech tagging,
- automatic speech recognition,
- automatic word-to-word alignment,
- and any other idea you may have…

Hidden Markov Models

Rafael E. Banchs (rbanchs@gps.tsc.upc.edu)

Session 4: The EM algorithm & HMMs

Centre de Tecnologies i Aplicacions del Llenguatge i la Parla
UNIVERSITAT POLITÈCNICA DE CATALUNYA