Naive Bayes Classifier

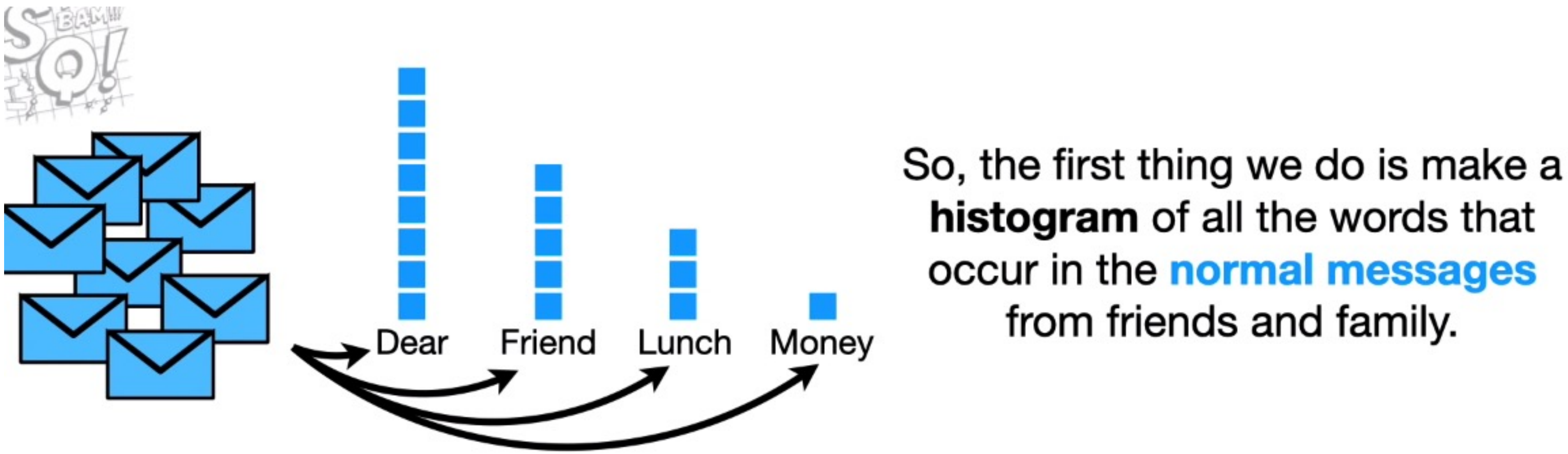
Set Up

- Imagine we get messages from friends and family
- But also unsolicited messages (advertising, scams…)

- And we don’t want to sort out the messages anymore, or at least try to reduce the amount by quite a bit and create an algorithm which will automatically send most spams to a spam box.

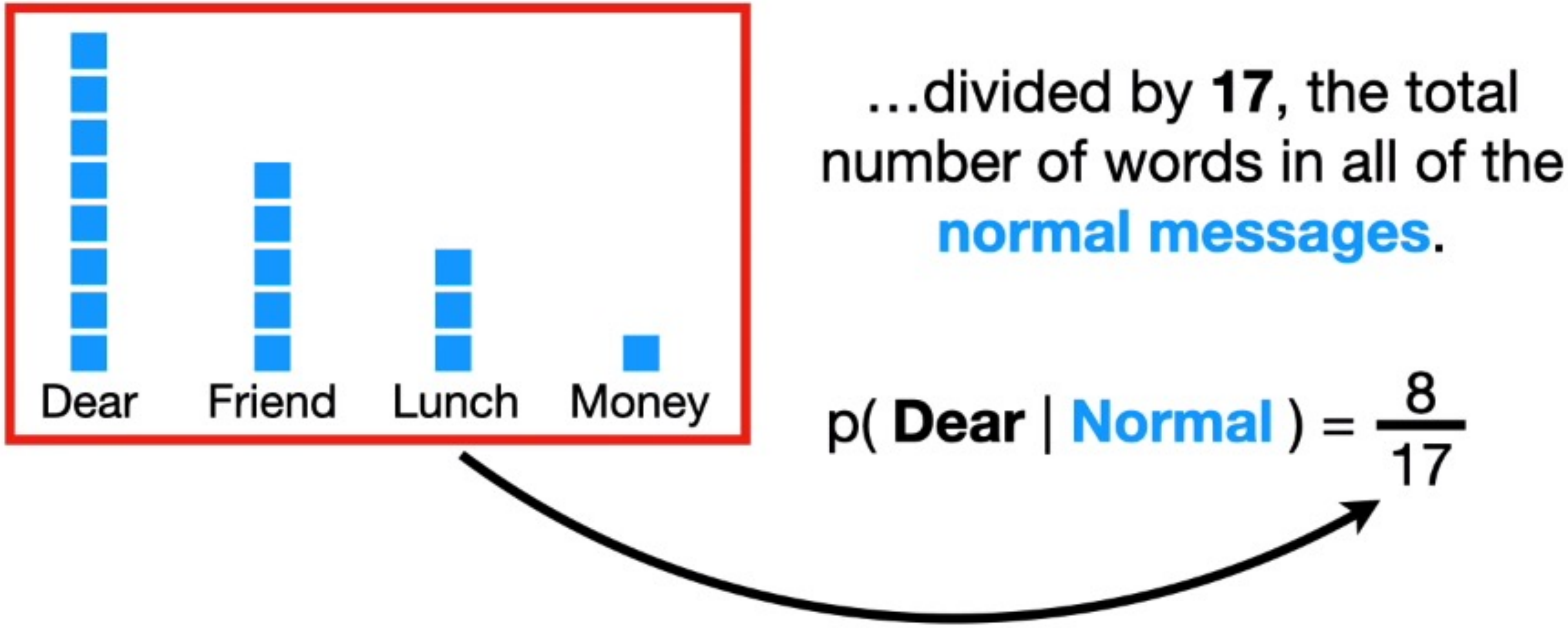
Intuition

- The following figures are from: https://www.youtube.com/watch?v=O2L2Uv9pdDA&t=695s



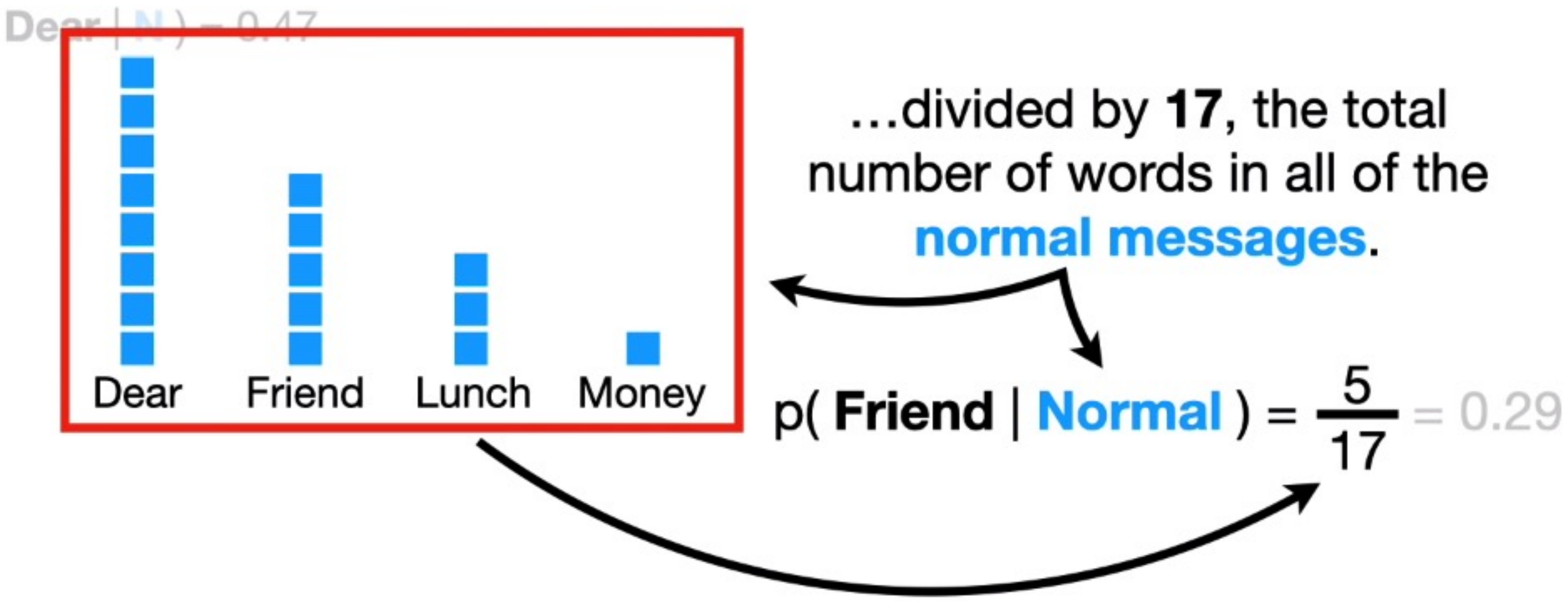
Intuition

- Can use the histogram to calculate the **probabilty** of seeing each word given it was in a **normal** message



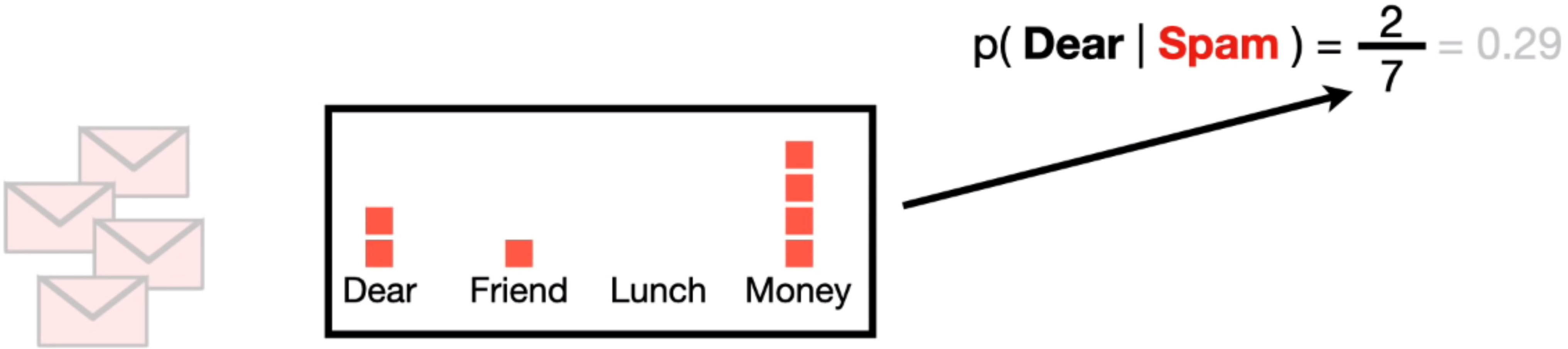
Intuition

- Can repeat the process for each word we see in our normal messages (hams).



Intuition

- Can do the exact same thing for our spams



Intuition

- We end up with a set of conditional **probabilities** or **likelihoods.**

$p(\textbf{Dear} \mid \textbf{N}) = 0.47$
$p(\textbf{Friend} \mid \textbf{N}) = 0.29$
$p(\textbf{Lunch} \mid \textbf{N}) = 0.18$
$p(\textbf{Money} \mid \textbf{N}) = 0.06$

$p(\textbf{Dear} \mid \textbf{S}) = 0.29$
$p(\textbf{Friend} \mid \textbf{S}) = 0.14$
$p(\textbf{Lunch} \mid \textbf{S}) = 0.00$
$p(\textbf{Money} \mid \textbf{S}) = 0.57$

Terminology Alert!!!

Because we have calculated the probabilities of discrete, individual words, and not the probability of something continuous, like weight or height, these **Probabilities** are also called **Likelihoods**.

Intuition 1st Example

- Now imagine we get a message that says:

Dear friend

- **We want to decide where it should go:**
  - **Our normal inbox**
  - **Or our spam box**

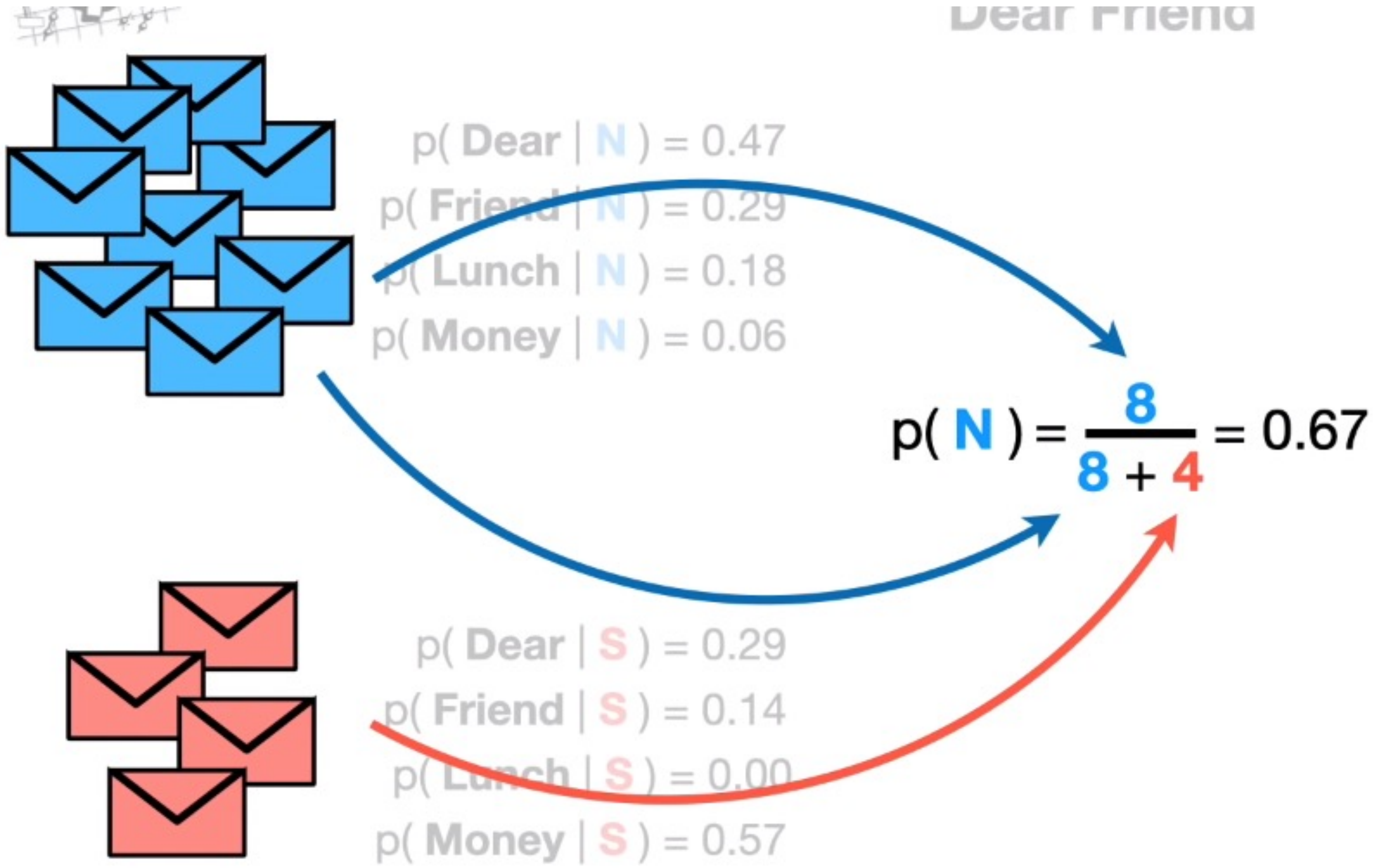
Intuition 1st Example

Dear Friend

We start with an initial guess about the probability that any message, regardless of what it says, is a **normal message**.

$p(\mathbf{N})$

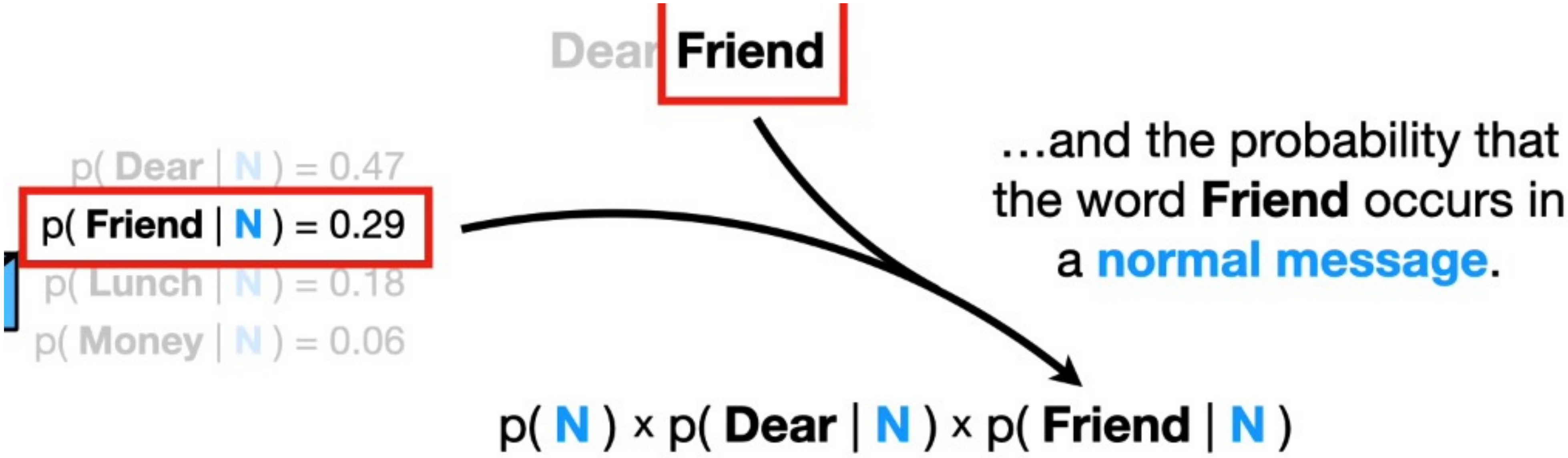
Intuition 1st Example



For example, since **8** of the **12** messages are **normal messages**, our initial guess will be **0.67**.

Intuition 1st Example

- Multiply the probabilities with each other



Intuition 1st Example

However, technically, it is *proportional* to the probability that the message is **normal**, given that it says **Dear Friend**.

$$0.67 \times 0.47 \times 0.29 = 0.09 \propto p(\,\mathbf{N} \mid \mathbf{Dear\ Friend}\,)$$

Intuition 1st Example

- Same process for spam:

Like before, we can think of **0.01** as the score that **Dear Friend** gets if it is **Spam**.

$$0.33 \times 0.29 \times 0.14 \boxed{= 0.01}$$

Intuition 1st Example

- Compare both results :

$$p(\mathbf{N}) \times p(\mathbf{Dear} \mid \mathbf{N}) \times p(\mathbf{Friend} \mid \mathbf{N}) \boxed{= 0.09}$$

$$p(\mathbf{S}) \times p(\mathbf{Dear} \mid \mathbf{S}) \times p(\mathbf{Friend} \mid \mathbf{S}) = 0.01$$

Intuition 2nd Example

- Now let’s try and classify :

Lunch Money Money Money Money

Intuition 2^{nd} Example

$p(\text{Dear} \mid N) = 0.47$

$p(\text{Friend} \mid N) = 0.29$

$p(\text{Lunch} \mid N) = 0.18$

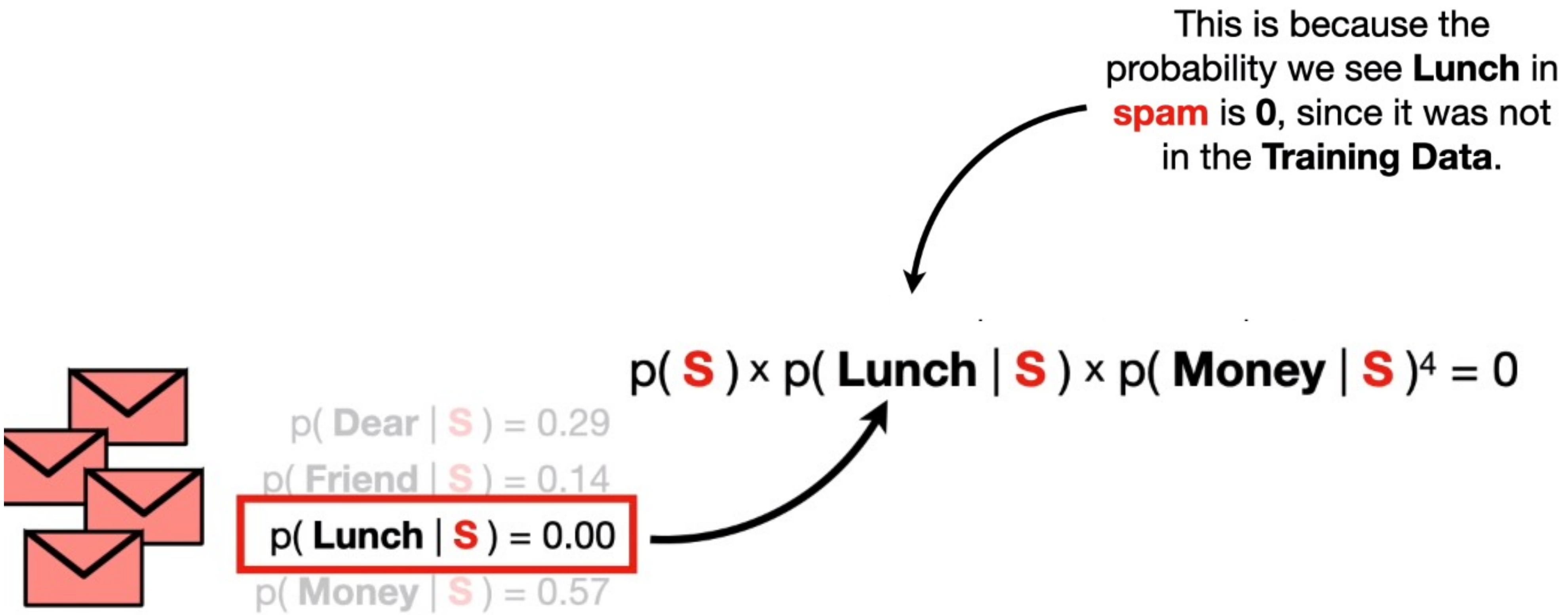
$\boxed{p(\text{Money} \mid N) = 0.06}$

…and the probability we see **Money** four times, given that it is in a **normal message**.

$$p(N) \times p(\text{Lunch} \mid N) \times p(\text{Money} \mid N)^4$$

$$p(N) \times p(\text{Lunch} \mid N) \times p(\text{Money} \mid N)^4 = 0.000002$$

Intuition 2nd Example

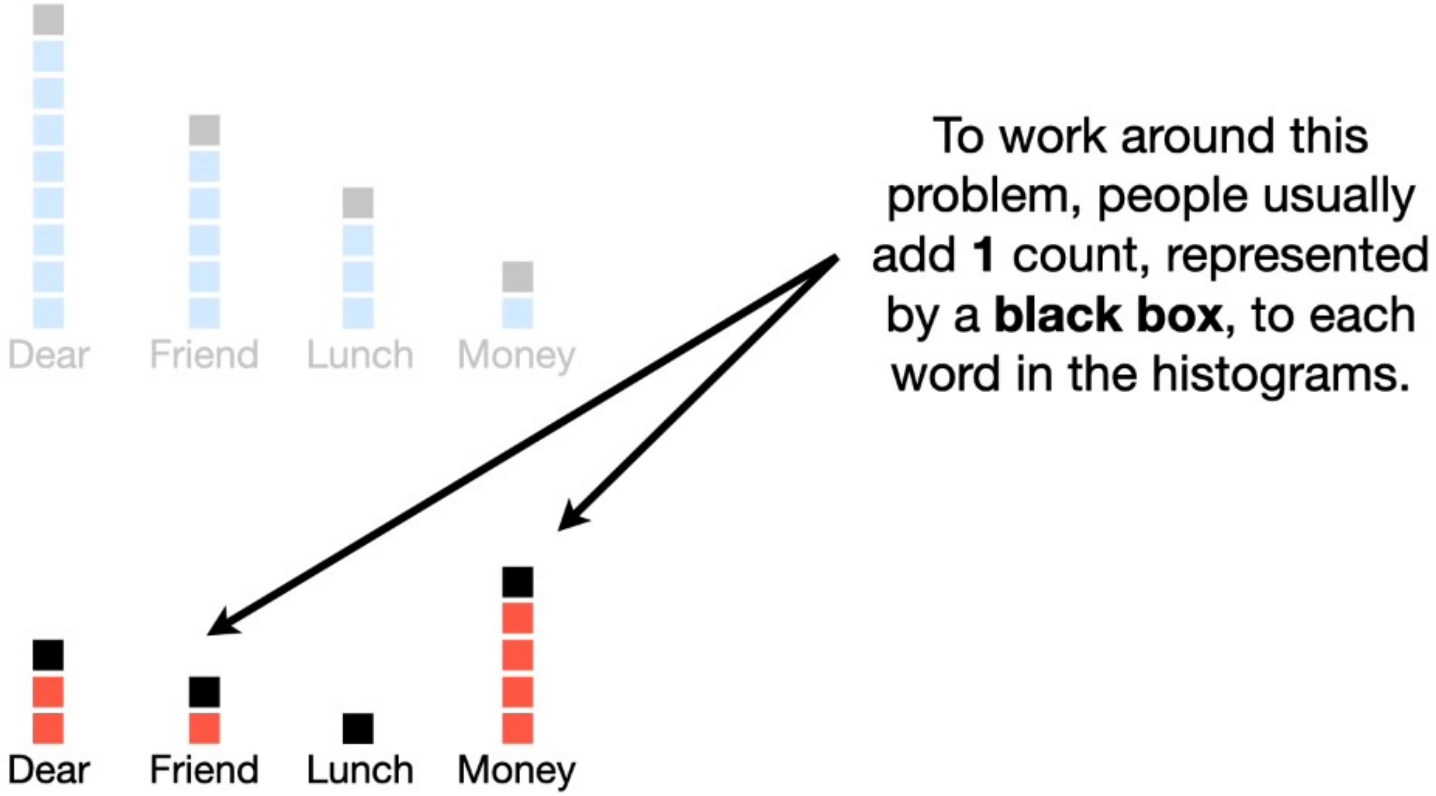


Intuition 2^{nd} Example

- In other words, if a message contains lunch, it will not be classified as spam…
- This is not satisfactory !
- No matter how many times we see the word **money,** or any other word which has a high probability of being in spam, the end result will be 0 …

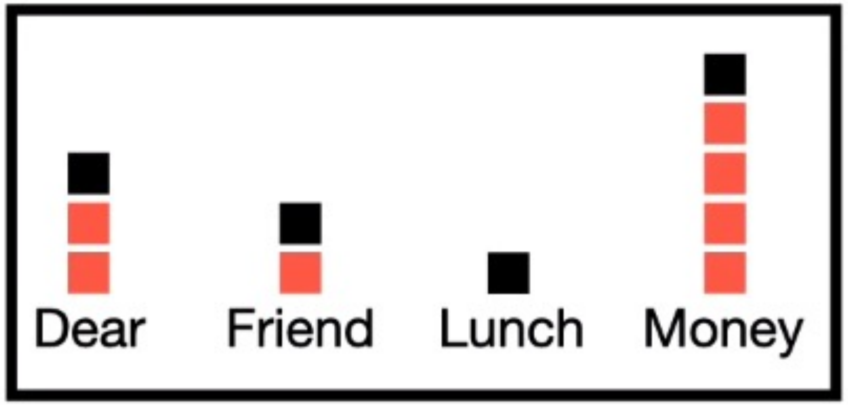
Pseudocounts

- The pseudocount is referred to with α (=1 in this case)



- Now, when calculating the probas of observing each word, we never get 0.
- Careful : we now need to add 4 (total counts added = our vocab length) to the denominator

$$p(\textbf{Lunch} \mid \textbf{Spam}) = \frac{1}{7 + 4}$$



- Our values for P(Normal) and P(Spam) do not change however.
- We are '**hallucinating**' data so our model will ***generalize*** better; meaning we're taking into account seeing certain words that our data doesn't account for so that if we *do* ever see this in the future, we have a probability estimate that we can use.

$$p(\mathbf{N}) \times p(\mathbf{Lunch} \mid \mathbf{N}) \times p(\mathbf{Money} \mid \mathbf{N})^4 = 0.00001$$

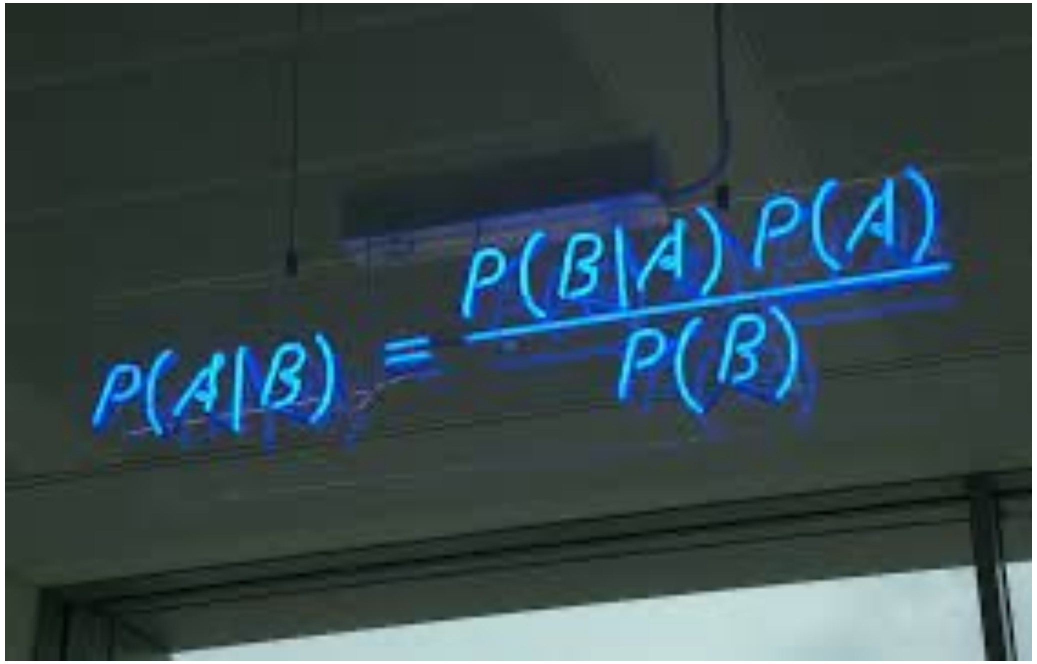
$$p(\mathbf{S}) \times p(\mathbf{Lunch} \mid \mathbf{S}) \times p(\mathbf{Money} \mid \mathbf{S})^4 = 0.00122$$

Why is this classifier called *Naive* Bayes ?

- This classifier treats all word orders the same, there is no sequence information taken into account, only single word counts.

- That said it tends to perform suprisingly well !

Putting things into perspective with the formula



In the case of Spam Filtering

- P(A|B) is *proportional* to our estimated probabilities

$$P(Spam|w_1, w_2, \ldots, w_n) \propto P(Spam) \cdot \prod_{i=1}^{n} P(w_i|Spam)$$

- P(spam) — probability of encountering a spam message
- P(w_i|spam) — the probability of a word being in the spam messages.

- Same logic for not spam (ham) :
  - P(ham) — probability of encountering a ham message
  - P(w_i|ham) — the probability of a word being in the ham messages

Why are we using proportinality vs equality ?

- Where did the denominator from the formula P(B) go ?
- P(B) = P(w1, w2, w3, …) will be the same when calculating the probabilities for both spam and ham.

- It can therefore be regarded as a constant :
  - We're not interested in the exact result, only which class has higher probability.
  - So we can save a little computation by doing so.

How de we measure the probability of a word given a class ? (same for ham)

$$P(w_i|Spam) = \frac{N_{w_i|Spam} + \alpha}{N_{Spam} + \alpha \cdot N_{Vocabulary}}$$

- N_wi|spam
  - the number of times a word wi is repeated in all spam messages.
- N_spam
  - the total number of words in the spam messages.
- N_vocabulary
  - the number of unique words in the whole dataset.
- Alpha
  - the coefficient we automatically add for the cases when a word is absent in spam but present in ham.

Implementation

- Now we have all we need to know to implement the algorithm ☺