Computing Relative Frequencies

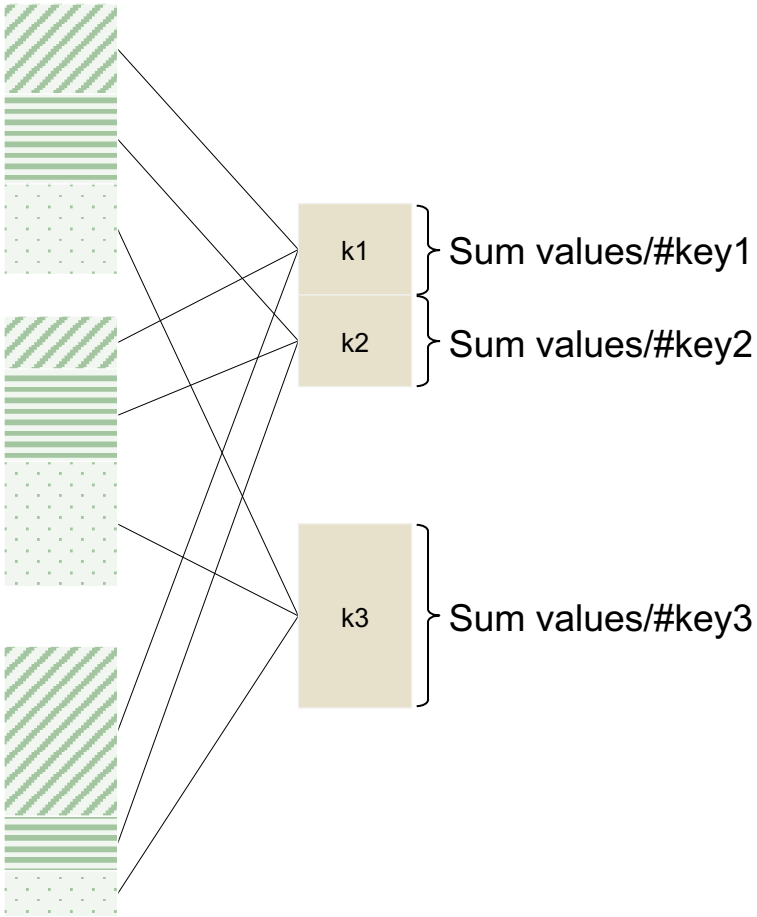
Applications

From absolute counts to ratios

- Natural language processing
- Summary statistics
- Marginal probabilities

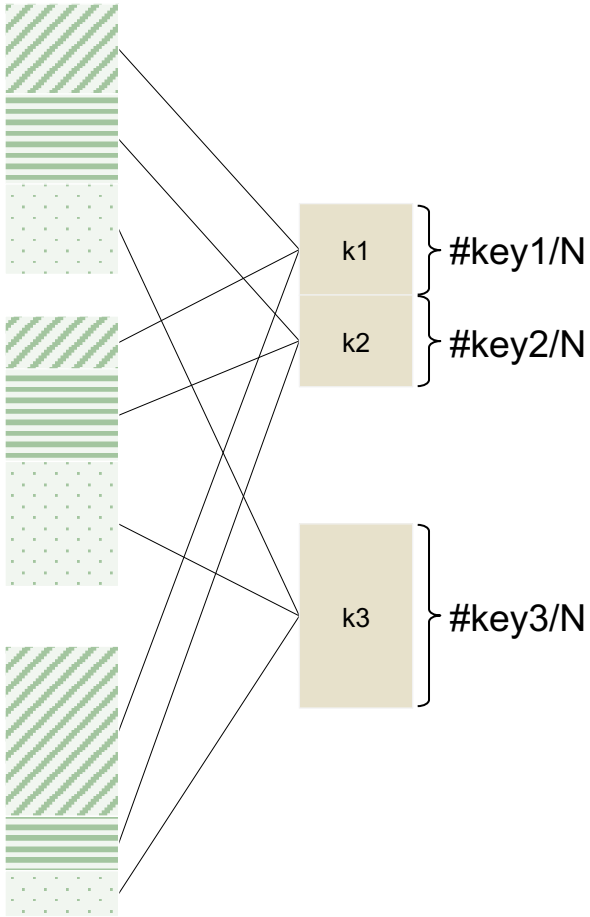
Design Patterns

Mean for each key



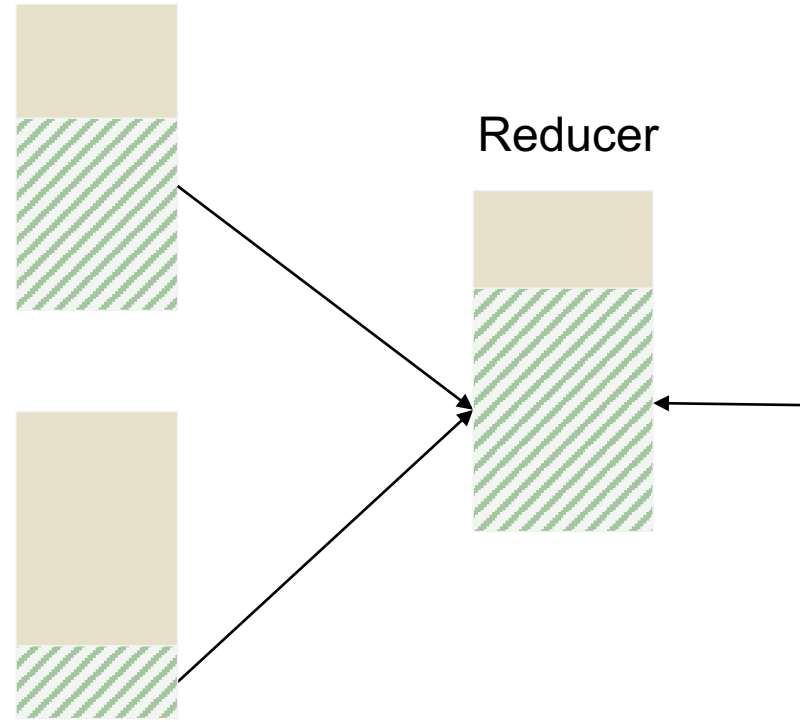
⇒ count, cardinality

Relative frequency (ratio)



N = total number of observations

Relative Frequencies Version 1: Reduce Side "in Memory" Hash Table



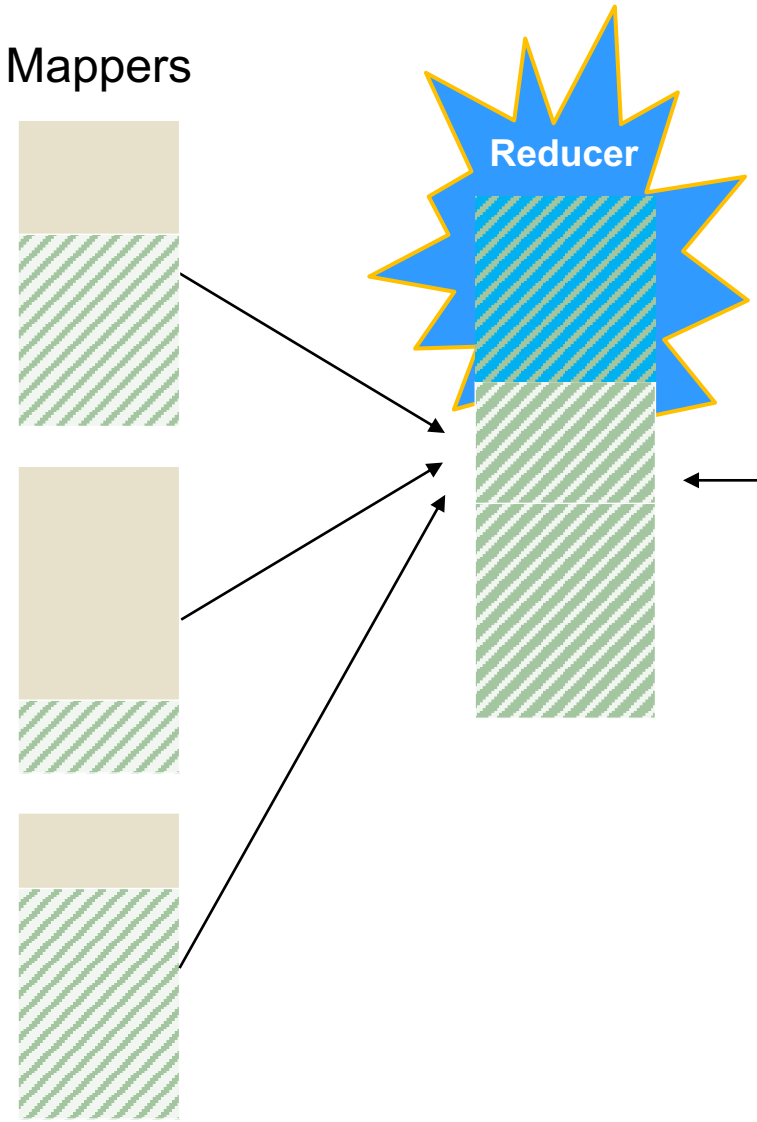
Dict = {key1: count,
key2: count
….}

Store everything in memory

Then, compute relative frequencies once we know N, where N=len(Dict)

Emit key, count/len(Dict)

Relative Frequencies Version 1: Reduce Side "in Memory" Hash Table



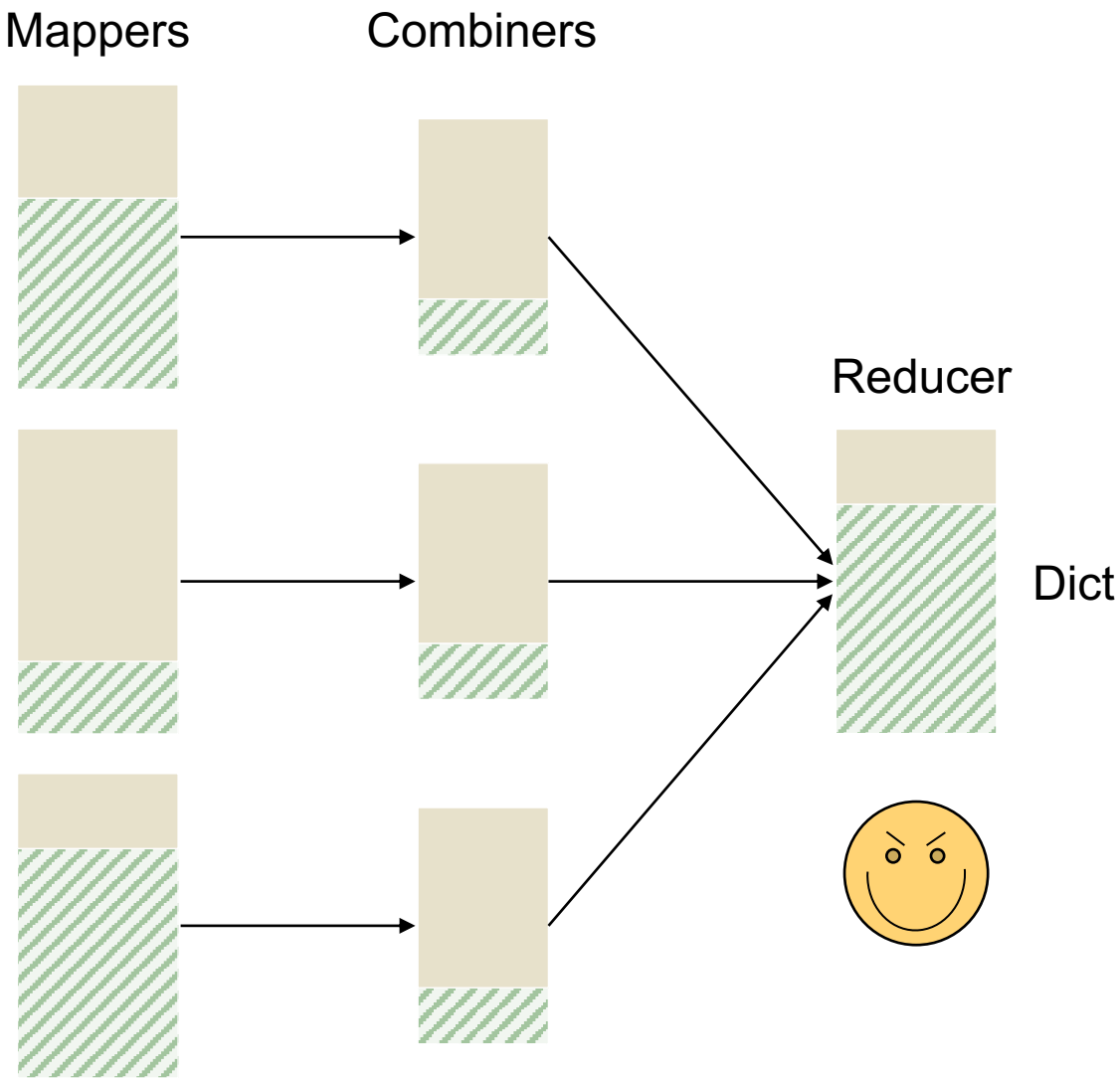
Dict = {key1: count,
key2: count
….}

Store
everything
in memory

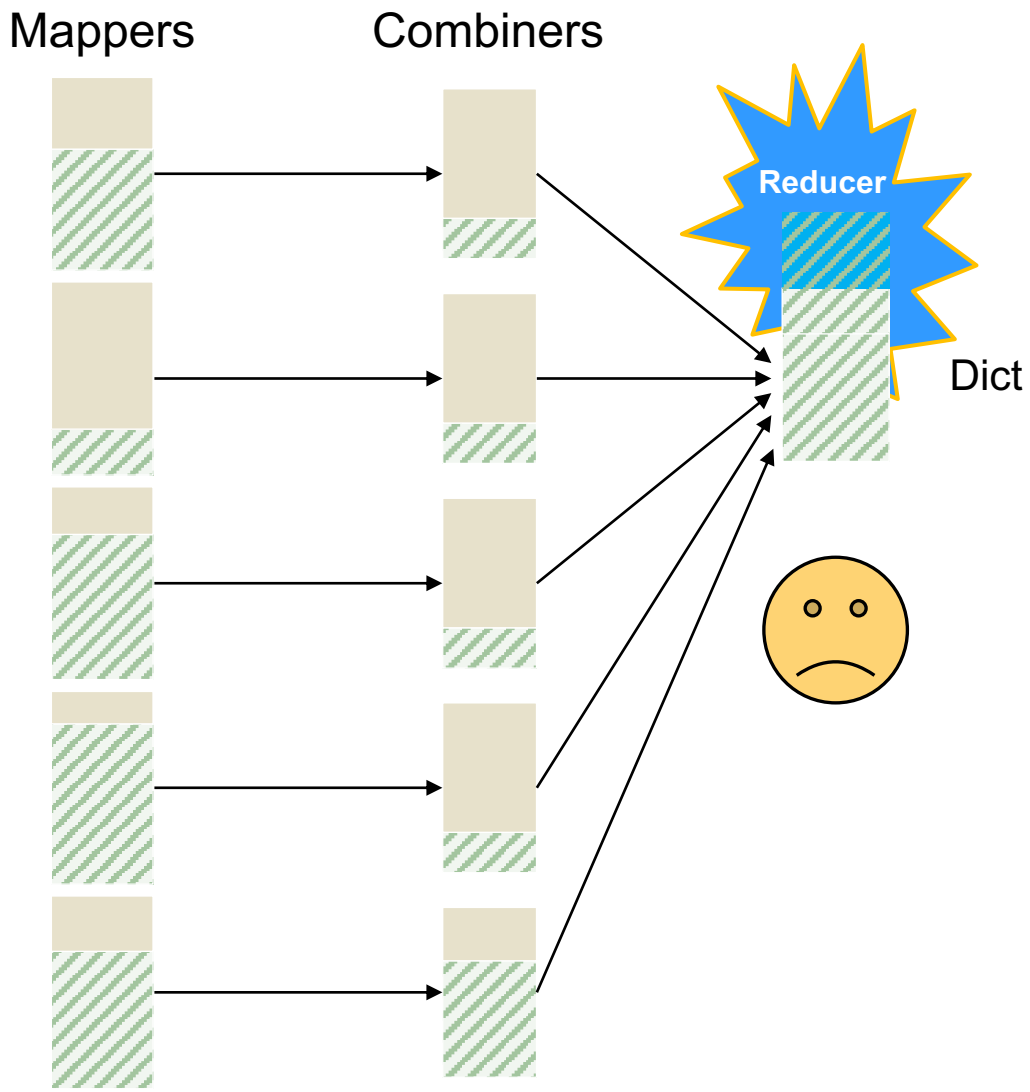
Then, compute
relative frequencies
once we know N,
where N=len(Dict)

Emit key, count/len(Dict)

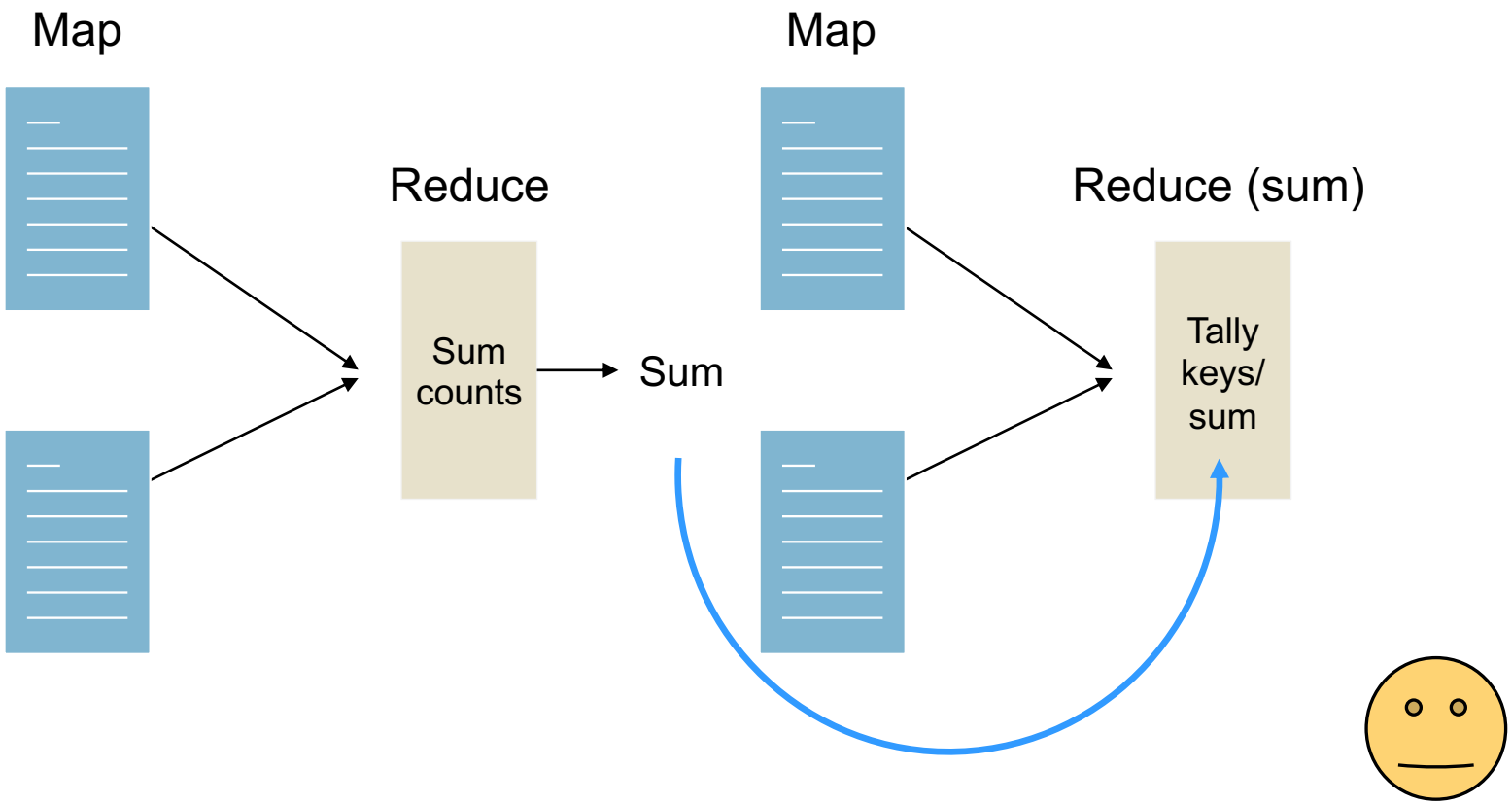
Relative Frequencies
Version 2: With Combiners



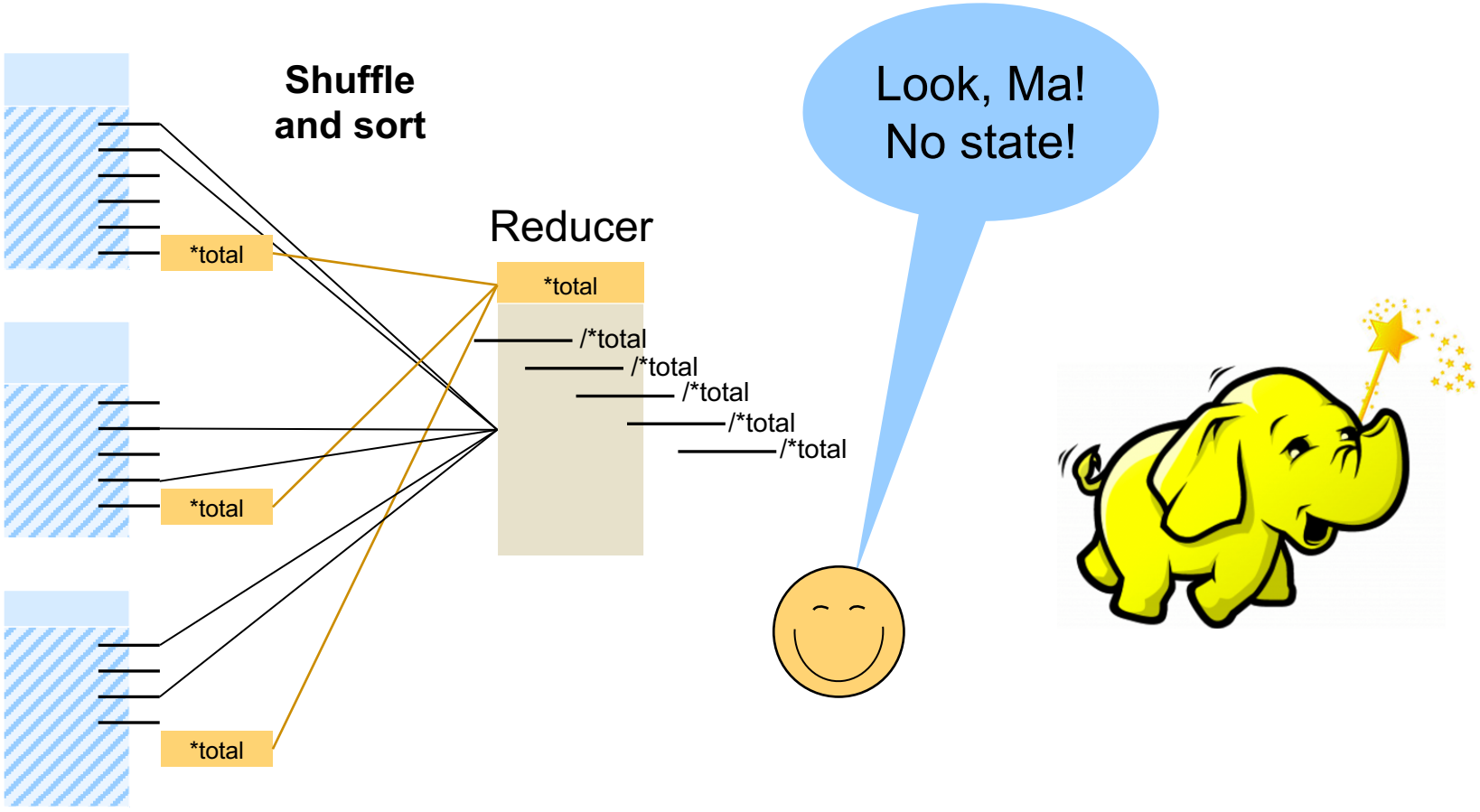
Relative Frequencies
Version 2: With Combiners



Stateless Relative Frequencies Version 1: Two MapReduce Jobs



Stateless Relative Frequencies Version 2: Order Inversion Pattern



Pseudo-code

```
def map(stdin):
   totalWords = 0
   for line in stdin:
      for word in line.split():
      print(word, 1)
      totalWords += 1
   print("**totalWords", totalWords)
```

```
def reduce(stdin):
   currentWord = None
   currentWordCount = 0
   N = 0
   for key, count in stdin:
      if key == "**totalWords":
         N += count
      elif key == currentWord:
         currentWordCount += count
      else:
         if currentWord:
         print(key,currentWordCount/N)
         currentWord = key
         currentWordCount = count
```

From Relative Frequencies to Naive Bayes

Next, we'll see how we can leverage the **order inversion pattern** to implement a naive Bayes text classification model in parallel.

See you in the next section!

The End

Naive Bayes Theory

From Word Count to Naive Bayes

- **Bag of Words:** order doesn't matter
- **Bernoulli Naive Bayes:** number of times a given word occurs doesn't matter
- **Multinomial Naive Bayes:** number of times a given word occurs matters

Probability Basics

Probability as **relative frequency:** the number of occurrences of an event out of a total number of events

The cat poked **the** dog in **the** eye.

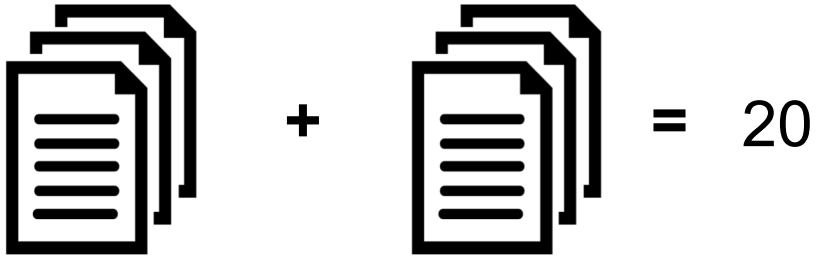
Number of occurrences of **the** = 3
Total number of words = 8

Relative frequency of **the** = ⅜ = 0.375

Probability Basics: Marginal Probability ("Prior")

Example: The number of documents in a given class out of the total number of documents in the training set.

Ham = 5 Spam = 15



Probability of ham = 5/20 = 0.25

Probability Basics: Joint Probability

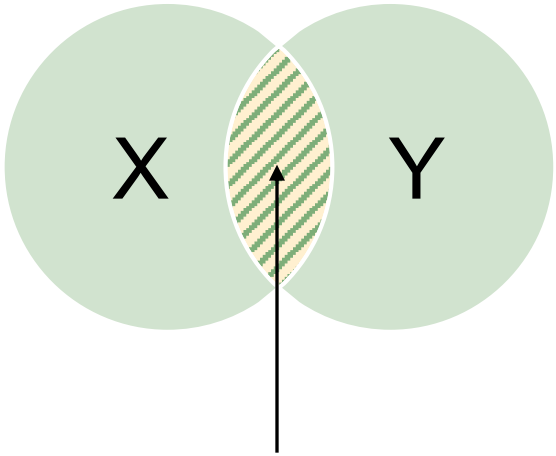
Example: The joint probability of picking up a card that is both red and 6.

$P(6 \cap \text{red})$
$= P(6) \times P(\text{red})$
$= 4/52 \times 26/52$
$= 1/26$

P (X ∩ Y)

P (X ,Y)

P (X and Y)



The **joint** probability that X *and* Y occur at the same time

Probability Basics: Conditional Probability

Example: the probability that you get a 6, given that you drew a red card is:

$P(6 \mid \text{red})$

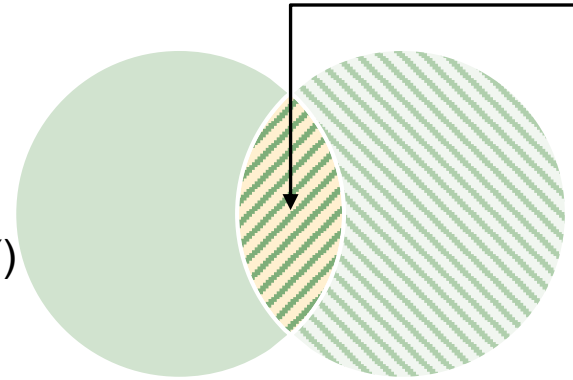
$= 2/26$

$= 1/13$

since there are two 6s out of 26 red cards

P (X|Y)

P (X given Y)



P (X ,Y)

= P (X ∩ Y)

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}$$

Probability Basics: Chain Rule

For two random variables X, Y, to find the joint probability, we can apply the definition of conditional probability to obtain:

$$P(X,Y) = P(X|Y)P(Y)$$

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}$$

$$P(X|Y)P(Y) = P(X,Y)$$

$$P(X,Y) = P(X|Y)P(Y)$$

Probability Basics: Chain Rule

With four variables, the chain rule produces this product of conditional probabilities:

$$P(X_4, X_3, X_2, X_1) = P(X_4|X_3, X_2, X_1) * P(X_3, X_2, X_1)$$
$$= P(X_4|X_3, X_2, X_1) * P(X_3|X_2, X_1) * P(X_2, X_1)$$
$$= P(X_4|X_3, X_2, X_1) * P(X_3|X_2, X_1) * P(X_2|X_1) * P(X_1)$$

Bayes Rule

Example: Suppose that we know the probability of a document given that the class is SPAM, P(Doc|SPAM), but we want to find the probability that the class is SPAM given the document, P(SPAM|Doc).

Applying the definition of conditional probability, and some algebra, we have …

Bayes Rule!

$$
\begin{aligned}
P(X|Y)P(Y) &= P(X,Y) \\
&= P(X)P(Y) \\
&= P(Y)P(X) \\
&= P(X,Y) \\
&= P(Y|X)P(X)
\end{aligned}
$$

$$
\frac{P(X|Y)PY}{P(X)} = \frac{P(Y|X)\,P(X)}{P(X)}
$$

$$
P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}
$$

Prior

Posterior

Likelihood

Evidence

Learning a (Full) Bayesian Model: P(Y|X)

One way to learn P(Y|X)

- Estimate the joint probability distribution P(X|Y) and P(Y) from the training data.
- Use these estimates, together with Bayes rule above, to determine $P(Y|X = x_k)$ for any new instance x_k.

1. **Estimate of P(Y) takes just O(n) time**
   a. #ExamplesWithLabel/#TotalNumberOfExamples.
   b. Example: If 45 examples are in class 1 and 55 are in class 2, then P(Y = class1) = 45/100 = 0.45.
2. **However, estimating P(X|Y) requires $2^n * 2$ possible states of the world** (parameter estimates)
   a. Assume n input attributes X_i take two discrete values, and Y has two possible class values; $2^n * 2$ possible states!
   b. $P(X = x_i|Y = y_j)$; 2^n possible states of the input world two possible output states

Learning a (Naive) Model

Learning $2^n * 2$ possible states of the world requires
is intractable!

Solution: the "naivete" assumption

- Instead of assuming that all the different permutations (states of the world) have different probabilities
- Make the simplifying assumption that combined elements (e.g., words in a sentence) are ...***statistically independent***
- Two events, X and Y, are statistically independent if the occurrence of one does not affect the probability the other

Probability Basics: Independence

Two events X and Y are independent if

$P(X,Y) = P(X)P(Y)$

Probability Basics: Independence

Furthermore, using the formula for conditional probability and independence, we can see that if X and Y are independent, then:

$P(X|Y) = P(X)$

Conditional Probability	Independence
$P(X\|Y) = \frac{P(X,Y)}{P(Y)}$	$P(X,Y) = P(X)P(Y)$

$$P(X|Y)P(Y) = P(X,Y)$$

$$P(X|Y)P(Y) = P(X)P(Y)$$

$$P(X|Y) = P(X)$$

Independence Summary

"Naive": The probability of event X does not depend on Y.

- **Pairwise independence**
  - P(X|Z) = P(X)
  - P(X,Y|Z) = P(X|Z)*P(Y|Z)
    - Since P(X,Y|Z) = P(X|Y,Z)*P(Y|Z) *[by the chain rule]*
    - and P(X|Y, Z) = P(X|Z)
- **Conditional independence**
  - P(X|Y) = P(X)
  - P(X|Y, Z) = P(X|Z)
  - P(X, Y) = P(X)*P(Y)

Derive Naive Bayes Algorithm

$$P(Y = y_k | X_1, X_2 \ldots X_n) = \frac{P(Y = y_k) P(X_1 \ldots X_n | Y = y_k)}{\Sigma_j P(Y = y_j) P(X_1 \ldots X_n | Y = y_j)}$$

Example: Consider the case where X = <X_1, X_2>.

$$P(X|Y) = P(X_1, X_2 | Y)$$

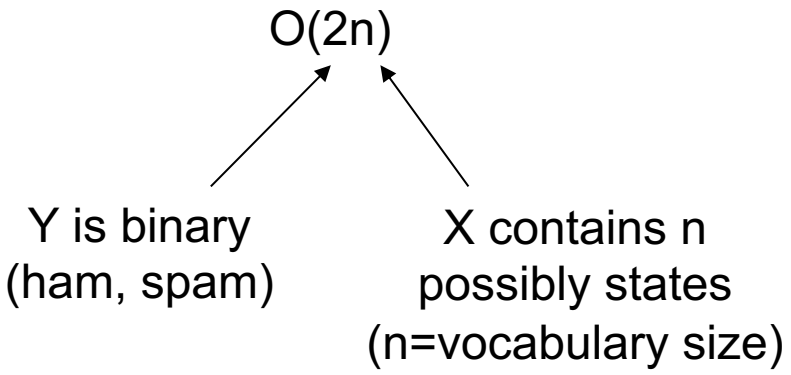
(Chain Rule) $$= P(X_1 | X_2, Y) P(X_2 | Y)$$

(Independence Assumption) $$= P(X_1 | Y) P(X_2 | Y)$$

Model Complexity

- O(2n)
  - When X has n states and Y has two states
- High bias
- Assumes no interaction between the variables

$$P(X_1 \ldots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$



Class Inference:
Classify a New Data Point

$$Y \leftarrow \operatorname{argmax}_{y_k} P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

Flavors of Naive Bayes for Text Classification

Multinomial $P(d|c) = P(\langle t_1, \ldots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k | c)$

Bernoulli $P(d|c) = P(\langle e_1, \ldots, e_M \rangle | c) = \prod_{1 \leq i \leq M} P(U_i = e_i | c).$

Training a Multinomial NB Model

► **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Priors

P(c) = ¾

P(~c) = ¼

Likelihoods

P(Chinese|c) = ⅝

P(Chinese|~c) = ⅓

…

Classification Using Multinomial NB Model

Model

Priors

P(c) = ¾

P(~c) = ¼

Likelihoods

P(chinese | c) = 5/8
P(chinese | ~c) = 1/3
P(beijing | c) = 1/8
P(beijing | ~c) = 0/3
P(shanghai | c) = 1/8
P(shanghai | ~c) = 0/3
P(macao | c) = 1/8
P(macao | ~c) = 0/3
P(japan | c) = 0/8
P(japan | ~c) = 1/3
P(tokyo | c) = 0/8
P(tokyo | ~c) = 1/3

$P(C|D) = P(C)P(D|C)/P(D)$

We omit the denominator as we are only interested in relative outcomes.

D_5 = Chinese Chinese Chinese Tokyo Japan

$P(C = c|D_5) = ¾ (⅝ * ⅝ * ⅝ * 0 * 0) = \mathbf{0}$

$P(C = \sim c|D_5) = ¼ (⅓ * ⅓ * ⅓ * ⅓ * ⅓) = 5/12 = 0.42$

LaPlace “Plus 1” Smoothing

$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t' \in V}(T_{ct'}+1)} = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

Model with smoothing

Priors

P(c) = ¾

P(~c) = ¼

Likelihoods

P(chinese | c) = (5+1)/(8+6) = 3/7
P(beijing | c) =(1+1)/(8+6) = 1/7
P(shanghai | c) = (1+1)/(8+6) = 1/7
P(macao | c) = (1+1)/(8+6) = 1/7
P(japan | c) = (0+1)/(8+6) = 1/14
P(tokyo | c) = (0+1)/(8+6) = 1/14

P(chinese | ~c) = (1+1)/(3+6) = 2/9
P(beijing | ~c) = (0+1)/(3+6) = 1/9
P(shanghai | ~c) = (0+1)/(3+6) = 1/9
P(macao | ~c) = (0+1)/(3+6) =1/9
P(japan | ~c) = (1+1)/(3+6) = 2/9
P(tokyo | ~c) = (1+1)/(3+6) = 2/9

Classification Using MNB Model With Smoothing

D_5 = Chinese Chinese Chinese Tokyo Japan

$P(C = c|D_5) = ¾ (3/7 * 3/7 * 3/7 * 1/14 * 1/14) =$ **0.0003**

$P(C = \sim c|D_5) = ¼ (2/9 * 2/9 * 2/9 * 2/9 * 2/9) = 5/12 = 0.0001$

We have now more “correctly” classified D_5 as belonging to the class China. Intuitively, this makes sense since the word China appears more times in D_5 than the other two words.

Using Logs To Prevent Underflow

$$\log(ab) = \log(a) + \log(b)$$

$$\log(a/b) = \log(a) - \log(b)$$

$$Y \leftarrow \text{argmax}\, y_k\, P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

$$Y \leftarrow \text{argmax}\, y_k\, \log P(Y = y_k) + \sum_i \log P(X_i | Y = y_k)$$

Time Complexity

- **Training time: $O(|D|L_d + |C||V|)$**
  - Where L_d is the average length of a document in D
    - Assumes V and all D_i, n_i, and n_{ij} precomputed in $O(|D|L_d)$ time during one pass through all the data
    - Generally, just $O(|D|L_d)$ since, usually, $|C||V| < |D|L_d$
- **Test time: $O(|C|L_t)$**
  - Where L_t is the average length of a test document
    - Very efficient overall, linearly proportional to the time needed to just read in all the data
    - Plus, robust in practice

More Examples

Naive Bayes examples

The End

Naive Bayes at Scale

Training an MNB Model in Parallel

- Start with a single reducer solution
- Without smoothing
- With smoothing

We Will Need To Calculate …

- Priors: N_c/N the relative frequencies of each class
- Conditional probabilities of each term: P(t|c)

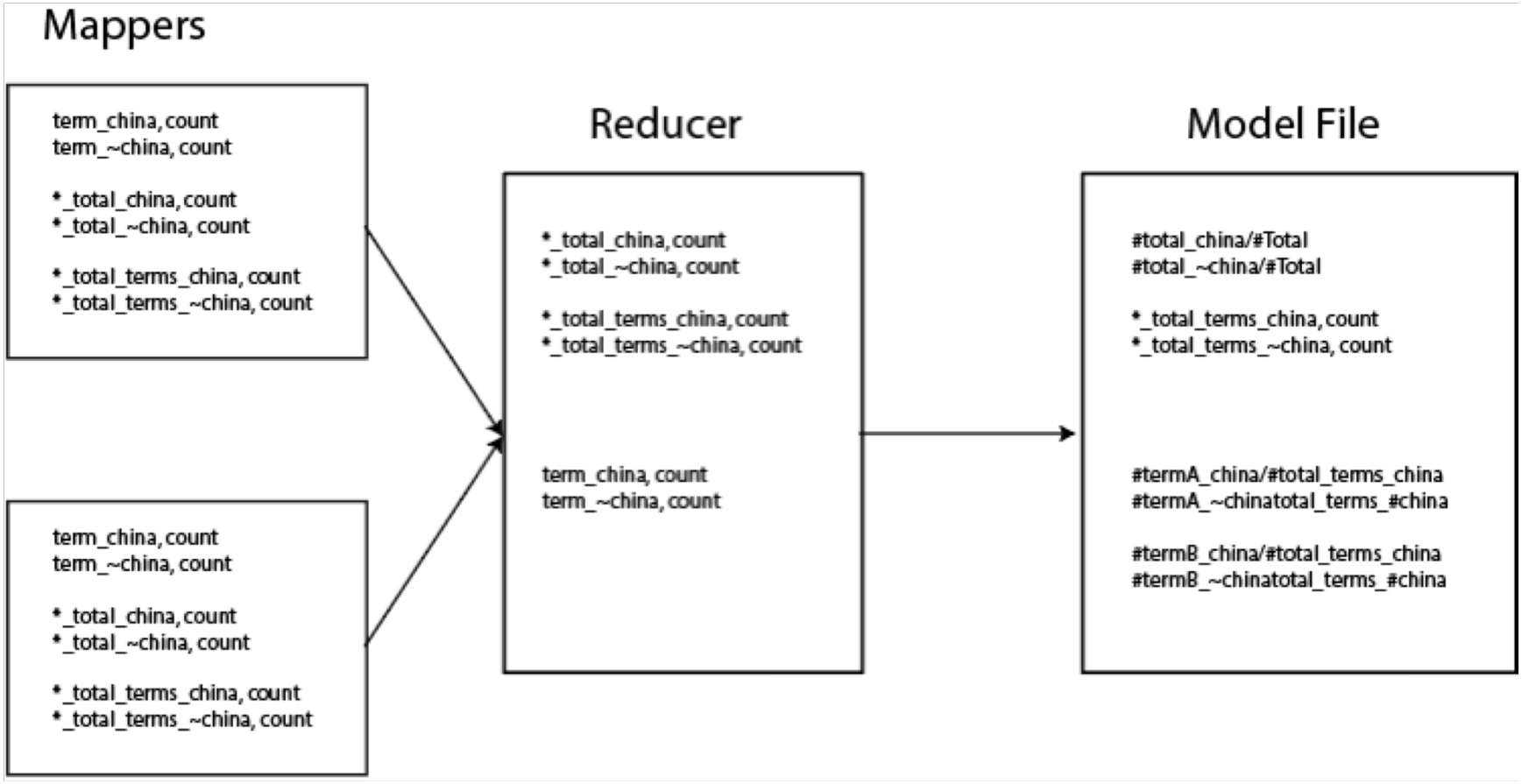
Priors

$$\frac{|\text{Class}_0|}{|\text{Docs}|} \qquad \frac{|\text{Class}_1|}{|\text{Docs}|}$$

Conditional Probabilities

$$\frac{|\text{Term}|}{|\text{Terms}_1|} \qquad \frac{|\text{Term}|}{|\text{Terms}_0|}$$

NB Training Illustrated



Mappers

- Input to Mapper 1

“1→Chinese Beijing Chinese→China”

“2→Chinese Chinese Shanghai→China”

EMIT:

<“Chinese”, (China, 1)>
<“Beijing”, (China, 1)>
<“Chinese”, (China, 1)>

<“Chinese”, (China, 1)>
<“Chinese”, (China, 1)>
<“Shanghai”, (China, 1)>

EMIT:

<“*total_China”, 2>
<“*total_notChina”, 0>
<“*total_terms_China”, 6>
<“*total_terms_notChina”, 0>

- Input to Mapper 2

“3→Chinese Macao→China”

“4→Tokyo Japan Chinese→notChina”

EMIT:

<“Chinese”, (China, 1)>
<“Macao”, (China, 1)>
<“Tokyo”, (notChina, 1)>
<“Japan”, (notChina, 1)>
<“Chinese”, (notChina, 1)>

EMIT:

<“*total_China”, 1>
<“*total_notChina”, 1>
<“*total_terms_China”, 2>
<“*total_terms_notChina”, 3>

Reducer

Input to Reducer

<“*total_China”, 2>
<“*total_China”, 2>
<“*total_notChina”, 0>
<“*total_notChina”, 1>

<“*total__terms_China”, 6>
<“*total__terms_China”, 2>
<“*total__terms_notChina”, 0>
<“*total__terms_notChina”, 3>

<“Beijing”, (China,1)>
<“Chinese”, (China,1)>
<“Chinese”, (China,1)>
<“Chinese”, (China,1)>
<“Chinese”, (China,1)>
<“Chinese”, (China,1)>
<“Chinese”, (notChina,1)>
<“Japan”, (notChina,1)>
<“Macao”, (notChina,1)>
<“Shanghai”, (notChina,1)>
<“Tokyo”, (notChina,1)>

Reducer Tallies Results in Stream:

*total_China, 3
*total_notChina, 1

total_, 4

*total_terms_China, 8
*total_terms_notChina, 3

Beijing	China 1,	notChina 0
Chinese	China 5,	notChina 1
Japan	China 0,	notChina 1
Macao	China 1,	notChina 0
Shanghai	China 1,	notChina 0
Tokyo	China 0,	notChina 1

EMIT:

Prior, $^{3}/_{4}$, $^{1}/_{4}$

Beijing, $^{1}/_{8}$, 0
Chinese, $^{5}/_{8}$, $^{1}/_{3}$
Japan, 0, $^{1}/_{3}$
Macao, $^{1}/_{8}$, 0
Shanghai, $^{1}/_{8}$, 0
Tokyo, 0, $^{1}/_{3}$

Model File

MNB With Smoothing

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

MNB With Smoothing

- Precompute the vocabulary size
- Store the vocabulary in memory on the reducer
- Two MapReduce jobs

MNB With Smoothing
Using Multiple Reducers

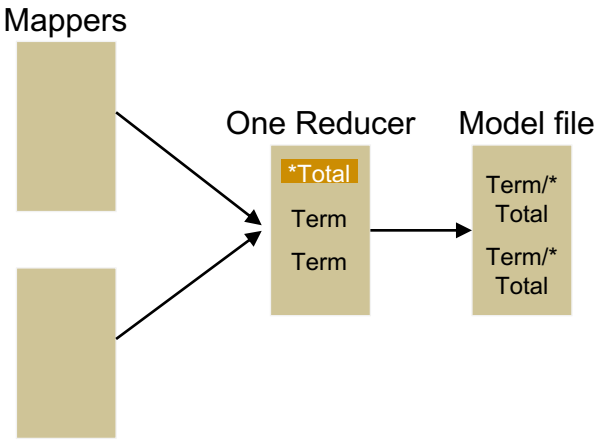
- Almost there! We now have the vocabulary size at our disposal, but we are still limited to a single reducer that is not very scalable.
- Custom partitioning to the rescue—see you in the next section!

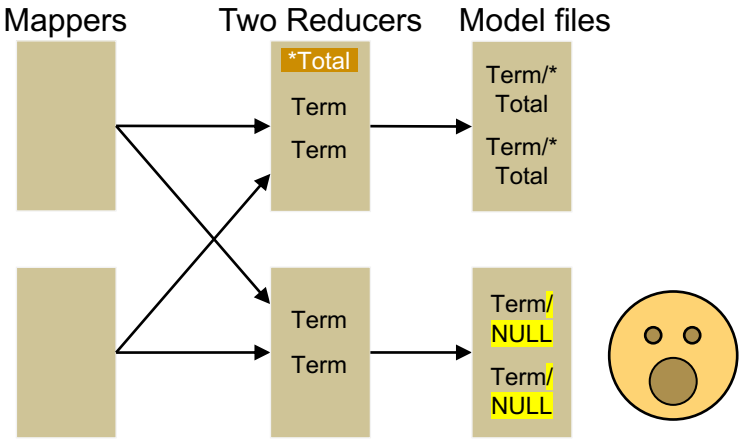
The End

Custom Partitioning

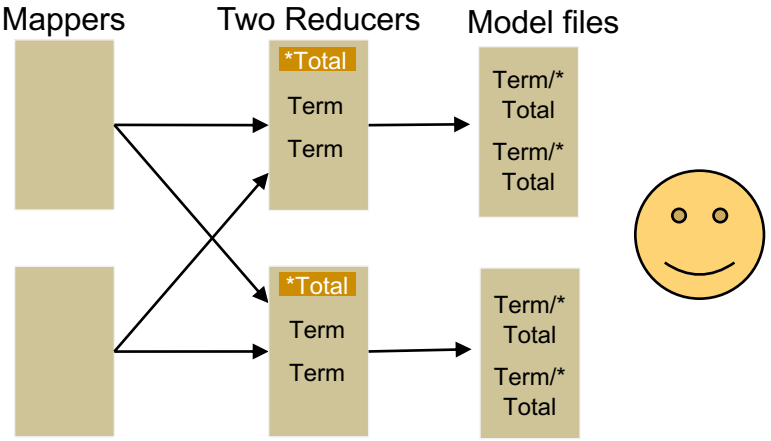
Motivation: Multiple Reducers

Previously

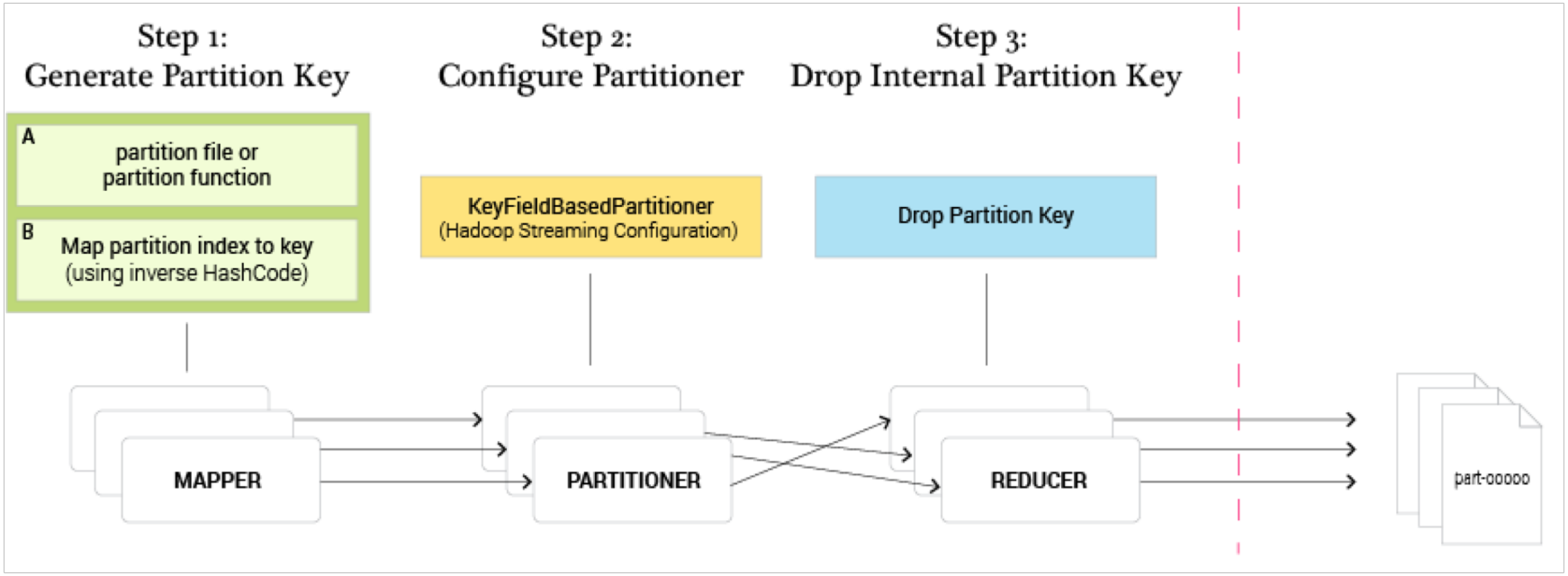




Desired



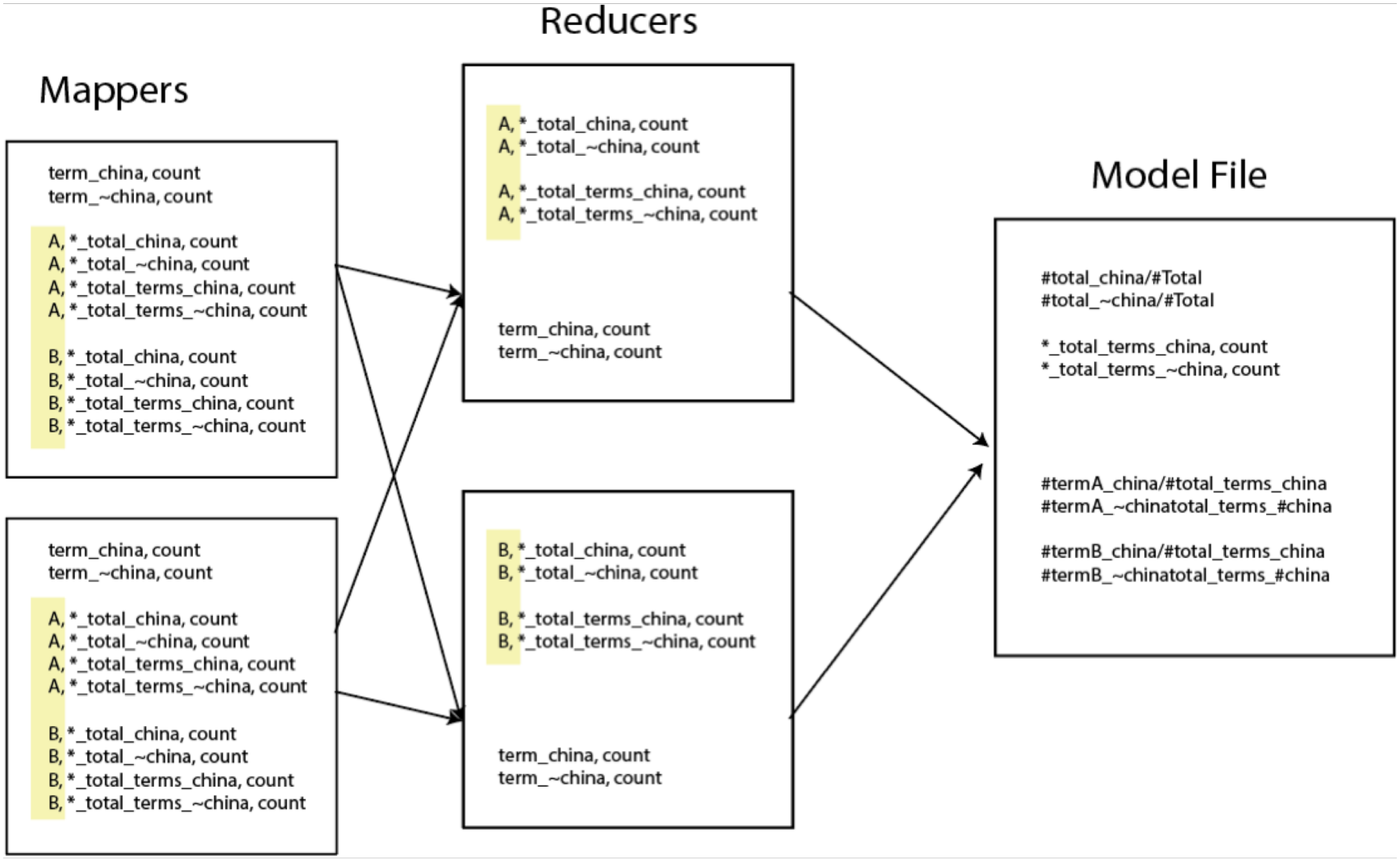
HashPartitioner



HashPartitioner

```
protected int hashCode(byte[] b, int start, int end, int currentHash) {
  for (int i = start; i <= end; i++) {
    currentHash = 31*currentHash + b[i];
  }
  return currentHash;
}

protected int getPartition(int hash, int numReduceTasks) {
  return (hash & Integer.MAX_VALUE) % numReduceTasks;
}
```

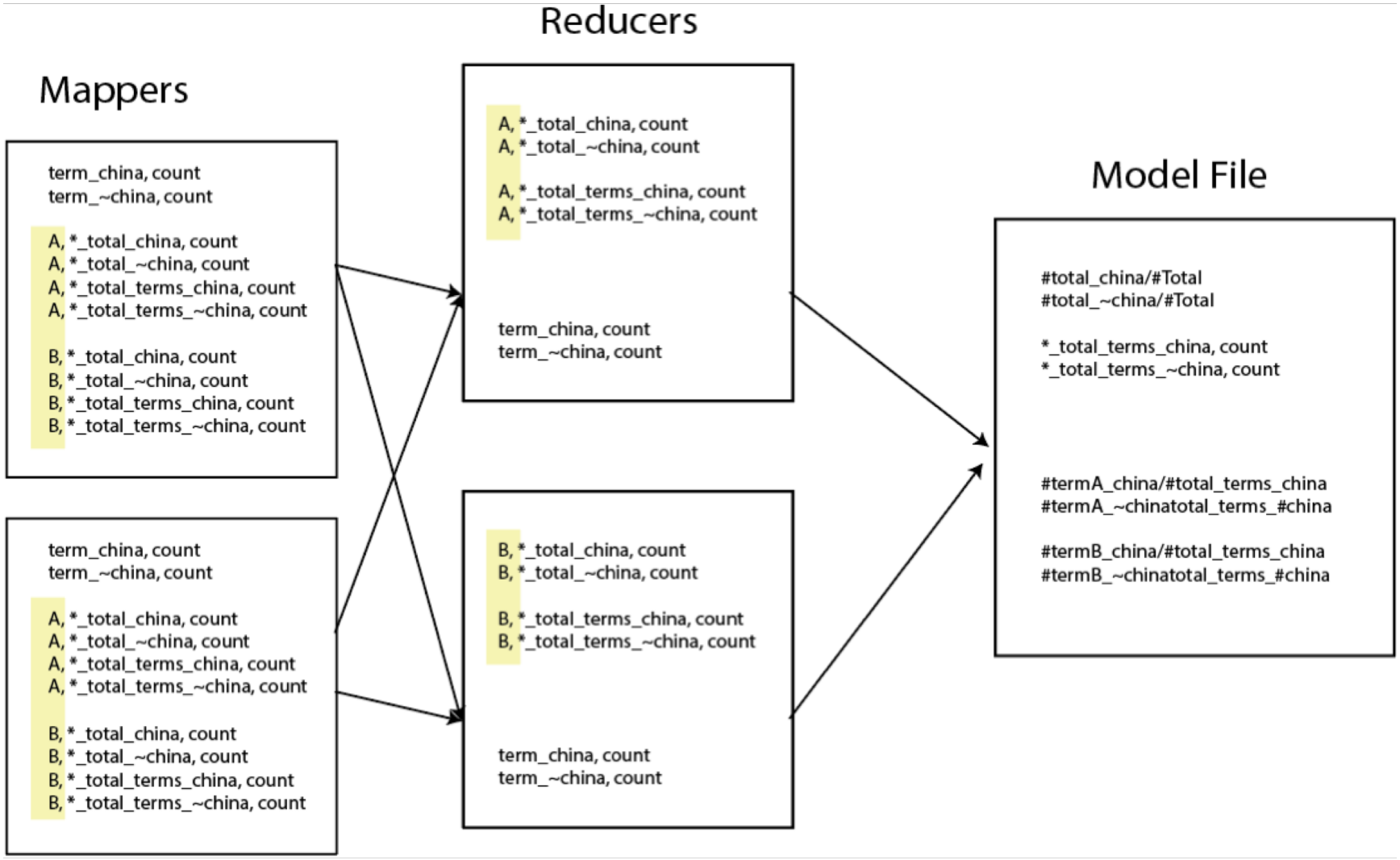
Example With Multiple Reducers Illustrated



Example With Multiple Reducers Pseudo-Code

```
numReducers = 2

# Returns a new partition key as a function of the original key and number of reducers:
partKey = partFunction(key, numReducers)
for line in sys.stdin:
    # parse input and tokenize
    docID, class_, content = line.lower().split('\t')
    words = content.split(‘ ’)

    # update class counts
    if(class_ == china):
                    docTotals_china += 1
          wordTotals_china += 1 * len(words)
      else:
          docTotals_notChina += 1
          wordTotals_notChina += 1 * len(words)

    # emit words with a count for each class
    for word in words:
        print(partKey(word), word, (class_, 1)))

# finally, emit totals with special key (order inversion)
for pk in partKeys:
   print(pk, "*docTotals_china”, docTotals_china)
   print(pk, "*docTotals_notChina”, docTotals_notChina)
   print(pk, "*wordTotals_china”, wordTotals_china)
   print(pk, "*wordTotals_notChina”, wordTotals_notChina)
```

Example With Multiple Reducers Illustrated



Smoothed MNB Summary

- One MapReduce job to calculate vocabulary size using one or more reducers
- Second MapReduce job to calculate the conditional probabilities using multiple reducers

Combiners

Can we still, and should we, use combiners?

- **Yes!**

The End