# MapReduce Project - AutoComplete

赵敏 老师

**扫描二维码关注微信/微博**
**获取最新IT面试情报及权威解答**

微信: ninechapter
知乎专栏: http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

# Outline



- What is auto-complete & Where auto-complete has been used.
- Learn what is N-Gram Model.
- Build N-Gram model.
- Run N-Gram model on MapReduce.
- Test your N-Gram result on web page.

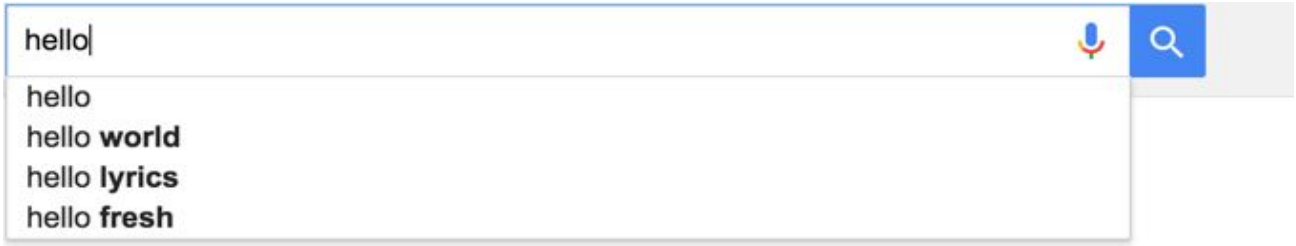
# What is Auto-Complete?



Let’s play Shannon Game!

# Where auto-complete has been deployed

- Google Suggestion




- Spelling Correction(letter)

We can implement it on N-Gram Model!

# What is N-Gram Model?

# What is N-Gram?



- An n-gram is a contiguous sequence of n items from a given sequence of text or speech

Example: I love big data

4-gram: I love big data

3-gram: I love big, love big data

2-gram?

# What is N-Gram?

2-gram?

I love, love big, big data

# What is Language Model?

- A language model is a probability distribution over entire sentences or texts.

# What is Model?

I want to eat _____ ?

- apple
- shit



Probability:

P(apple) > P(shit)

# What is N-Gram Model?

Use probability to predict next word/phrase

# Let’s build N-Gram Model

# N-Gram Model



- Predict N-Gram based on 1-Gram

I love big → data course/brother/island around

# N-Gram Model



- Predict N-Gram based on N-Gram

I love big → data course/brother/island around

Which one is better?

# N-Gram Model



- Predict N-Gram based on N-Gram

# Implement N-Gram Model

Steps:

- Read a large-scale document collections
- Build n-gram library
- Calculate probability
- Run the project on Mapreduce

| 2-gram | |
|---|---|
| want to | 200 |
| eat apple | 120 |
| eat shit | 1 |

| want | to = 200 |
|---|---|
| eat | apple = 120<br>shit = 1 |

# Fetch a large-scale document collections

- Document preprocessing:
  - Read each document
    - Line by line?
    - sentence by sentence?
  - Remove all non-alphabetical symbols.

# Build N-Gram Library

N=3

Phrase: This is cool since this is big data course.

| 1-gram | |
|---|---|
| This | 2 |
| is | 2 |
| cool | 1 |
| since | 1 |
| big | 1 |
| data | 1 |
| course | 1 |

| 2-gram | |
|---|---|
| This is | 2 |
| is cool | 1 |
| cool since | 1 |
| since this | 1 |
| is big | 1 |
| big data | 1 |
| data course | 1 |

| 3-gram | |
|---|---|
| This is cool | 1 |
| is cool since | 1 |
| cool since this | 1 |
| since this is | 1 |
| this is big | 1 |
| is big data | 1 |
| big data course | 1 |

# Build N-Gram Library



Let’s use MR to solve this problem!

- Mapper
- Reducer

# Mapper

Word Count:

This is cool since this is big data course.



一次划分

This, is, cool, since, this, is, big, data, course

N-Gram:

This is cool since this is big data course.



一次划分

This is, is cool, cool since, since this, this is, is big, big data, data course



二次划分

This is cool, is cool since, cool since this, since this is, this is big, is big data, big data course

# Reducer

| | |
|---|---|
| This is | 1 |
| This is | 1 |
| is cool | 1 |
| cool since | 1 |
| since this | 1 |
| is big | 1 |
| big data | 1 |
| data course | 1 |

Merge



| | |
|---|---|
| This is | 2 |
| is cool | 1 |
| cool since | 1 |
| since this | 1 |
| is big | 1 |
| big data | 1 |
| data course | 1 |

# Build Language Model



Probability of a word apprearing after a phrase:

$$\Pr(\text{word} \mid \text{phrase}) = \frac{\text{Count}(\text{phrase} + \text{word})}{\text{Count}(\text{phrase})}$$

# Build Language Model



Example:

| | |
|---|---|
| this | 1000 |
| this is | 500 |
| this is a | 125 |

$$\Pr(\text{is} \mid \text{this}) = \frac{\text{Count}(\text{this is})}{\text{Count}(\text{this})} = \frac{500}{1000} = 0.5$$

$$\Pr(a \mid \text{this is}) = \frac{\text{Count}(\text{this is } a)}{\text{Count}(\text{this is})} = \frac{125}{500} = 0.25$$

# Build Language Model

Can we simplify the probability computation?

# Build Language Model

$$\Pr(\text{word} \mid \text{phrase}) = \frac{\text{Count}(\text{phrase} + \text{word})}{\text{Count}(\text{phrase})}$$

| | |
|---|---|
| big | 1000 |
| big data | 800 |
| big baby | 100 |

Pr(data | big) = Count(big data) / Count(big) = 800/1000

Pr(baby | big) = Count(big baby) / Count(big) = 100/1000

Simplify

Pr(data | big) = Count(big data) = 800

Pr(baby | big) = Count(big baby) = 100

# Build Language Model

Input

| 2-gram | |
|---|---|
| Key | Value |
| This is | 2 |
| is cool | 1 |
| cool since | 1 |
| since this | 1 |
| is big | 1 |
| big data | 1 |
| data course | 1 |

| 3-gram | |
|---|---|
| Key | Value |
| This is cool | 1 |
| is cool since | 1 |
| cool since this | 1 |
| since this is | 1 |
| this is big | 1 |
| is big data | 1 |
| big data course | 1 |

Output

| Key | Value |
|---|---|
| This | is = 2 |
| This is | cool = 1 |
| cool | math = 100 |
| since I left | you = 10 |
| since | I = 20 |
| since I | left = 15 |

# Mapper

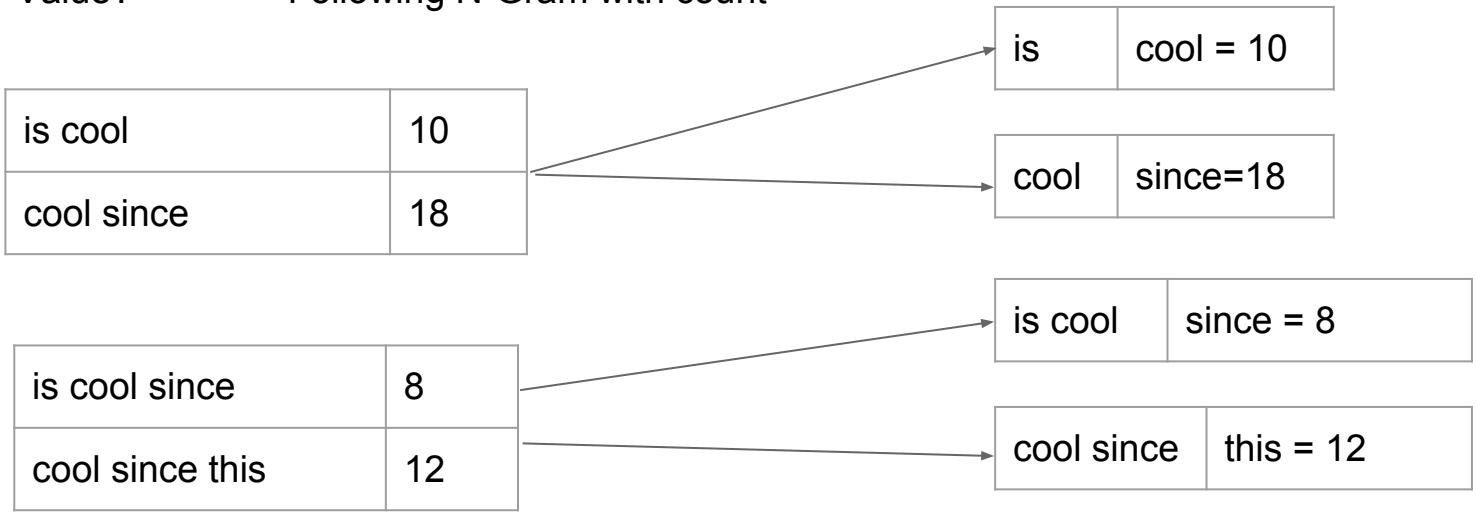


What should Mapper do?

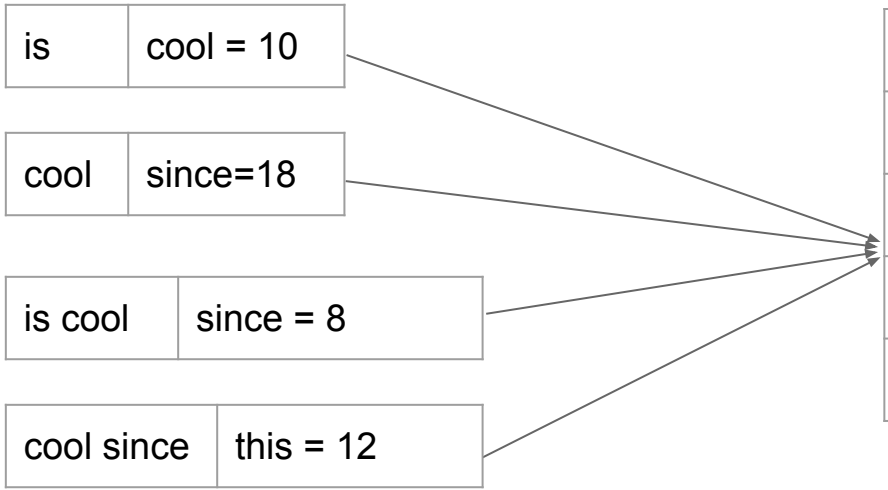
# Mapper

Key? Starting word/phrase

Value? Following N-Gram with count



# Reducer




| starting_word | following_word | probability |
|---|---|---|
| is | cool | 10 |
| cool | since | 18 |
| is cool | since | 8 |
| cool since | this | 12 |

# Build Language Model: Reducer

Hint:

For a given phrase, store only the top n words with the highest probabilities. This value should also be a command-line parameter to your MapReduce application. If two words have the same probability, choose the one which is lexicographically higher i.e. 'ab' comes before 'bc'. Use n = 5. The following figure shows an example of sorted probabilities.

# Build Language Model: Mapper

Hint:

You will want to ignore phrases that appear below a certain threshold, say t, from your n-gram count for your statistical language model to be accurate. Use t = 2 (This means that you need to ignore the phrases that fall under the category <= t).

# How to predict next n-gram?

| starting_phrase | following_word | count |
|---|---|---|
| a | man | 264 |
| a | little | 90 |
| a | great | 83 |
| a day | or | 14 |
| a day | and | 6 |

But this could only predict next 1-gram right?

# MySQL will do the magic job!

```
mysql> select * from output
    -> where starting_phrase like 'input%'
    -> order by count
    -> limit 10;
```

# What you have learned



- How to build N-Gram library
- How to build Language Model
- How to chain MR jobs
- How to write from MapReduce to Database
- PHP tools