Ruby-us Hagrid

Writing Harry Potter with Ruby

---

alexpeattie.com/hp

@alexpeattie

Writing Harry Potter with Ruby

Why should we do it?

What can we achieve?

How can we do it?

Why should we do it?

Category A

The "Potheads"

"Ouch, my heart"

Category B

The "Notters"

"Is that Yoda?"

What can we achieve?

(Spoiler!)

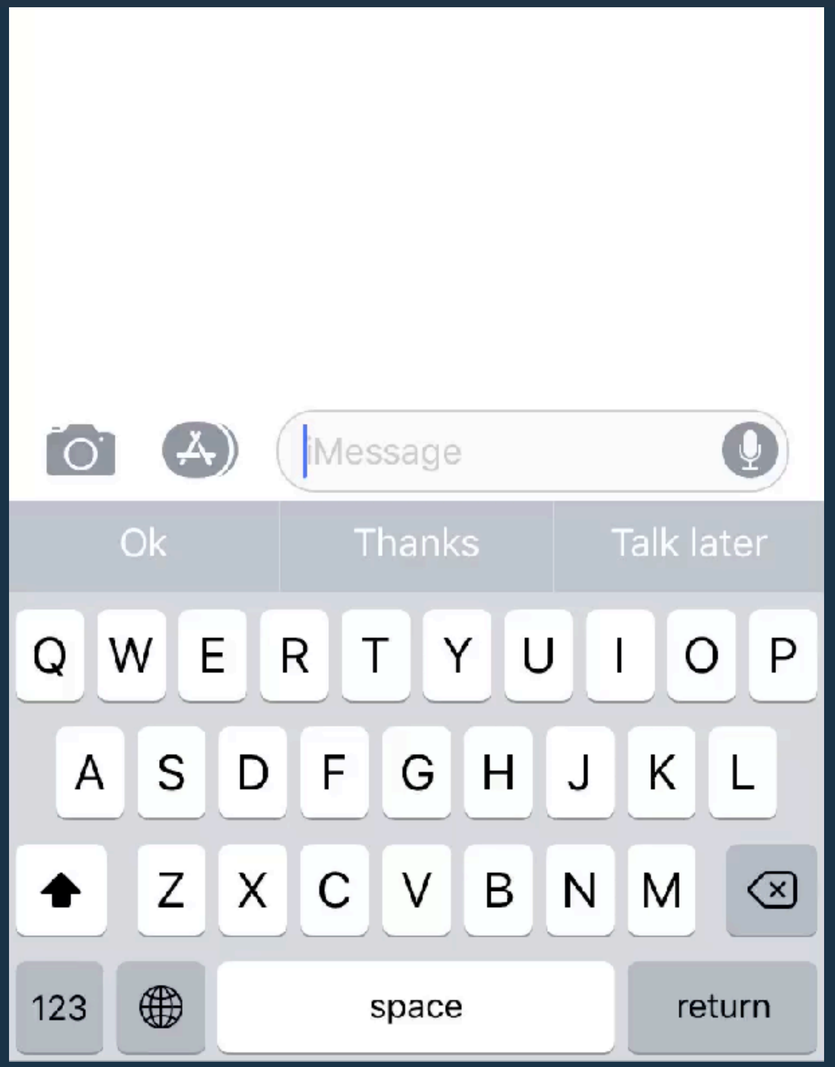
Neville, Seamus and Dean were muttering but did not speak when Harry had told Fudge mere weeks ago that Malfoy was crying, actually crying tears, streaming down the sides of their heads. “They revealed a spell to make your bludger” said Harry, anger rising once more.

How can we do it?

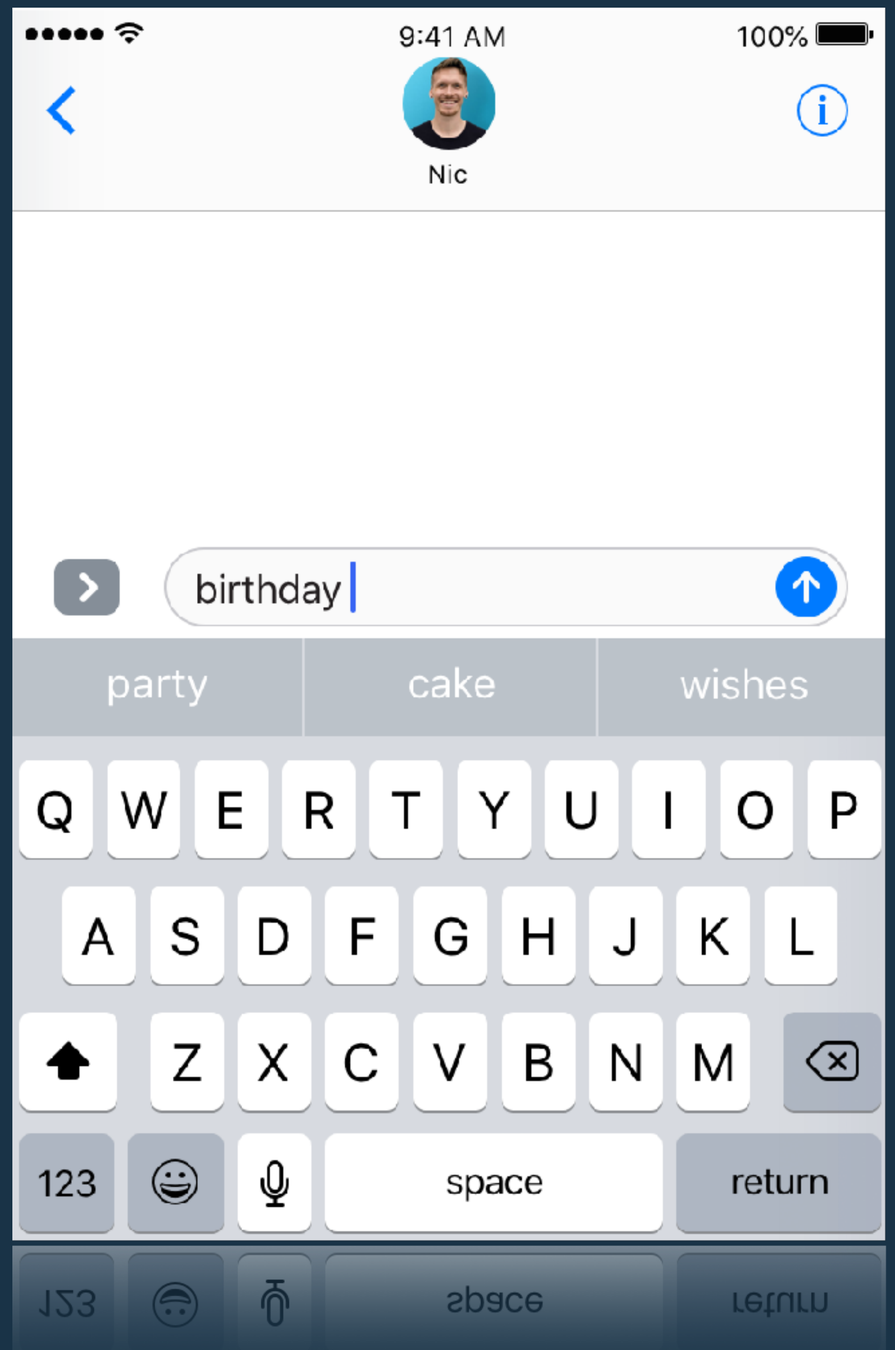
“They revealed a spell to make your bludger” said Harry, anger rising once more.

Key idea 1: Tell the story word by word

Key idea 2: Let’s take inspiration from our phones

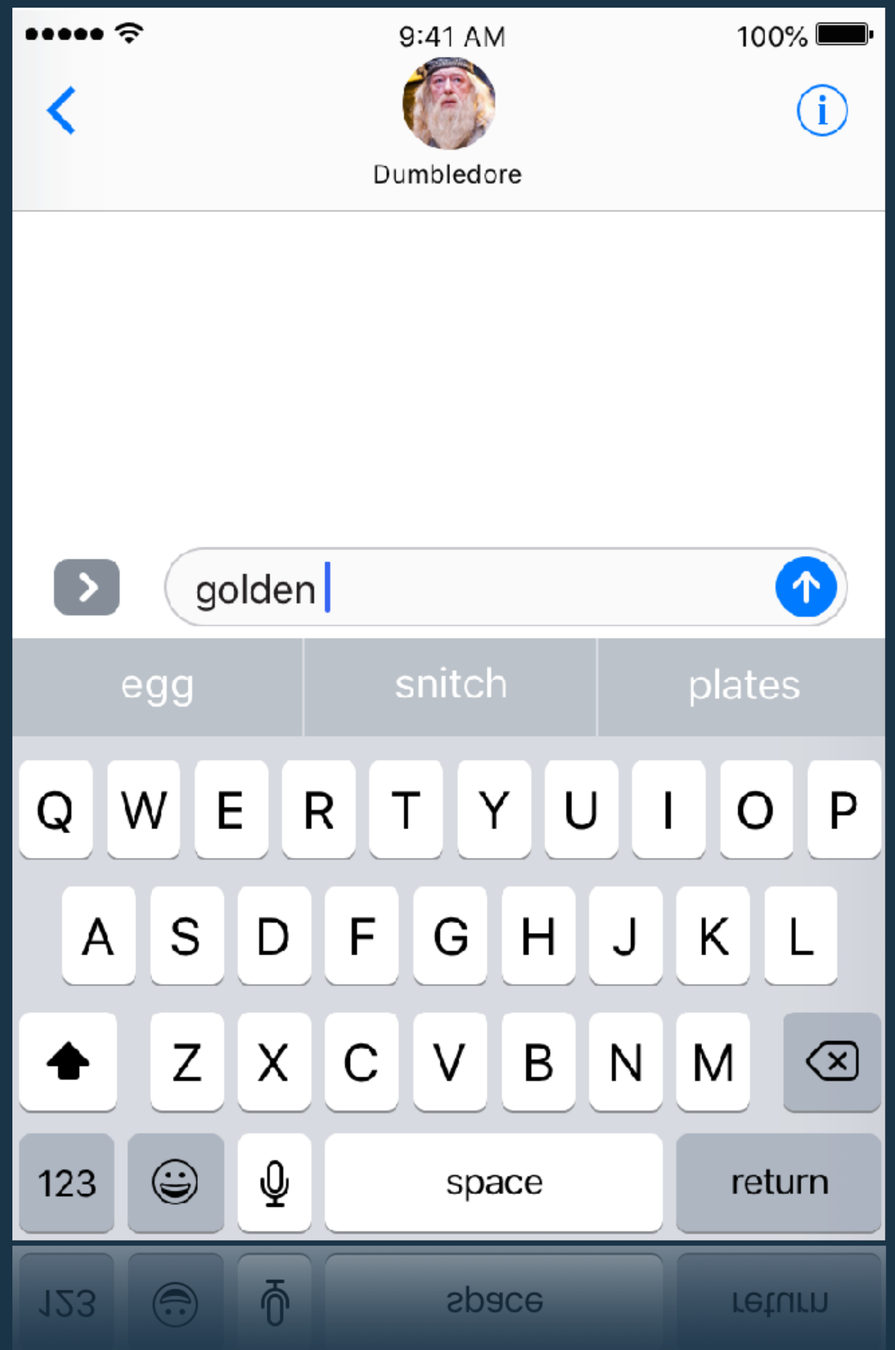


https://alexpeattie.com/assets/images/talks/hp/predictive.mp4



After “birthday”, I’ve used the word:

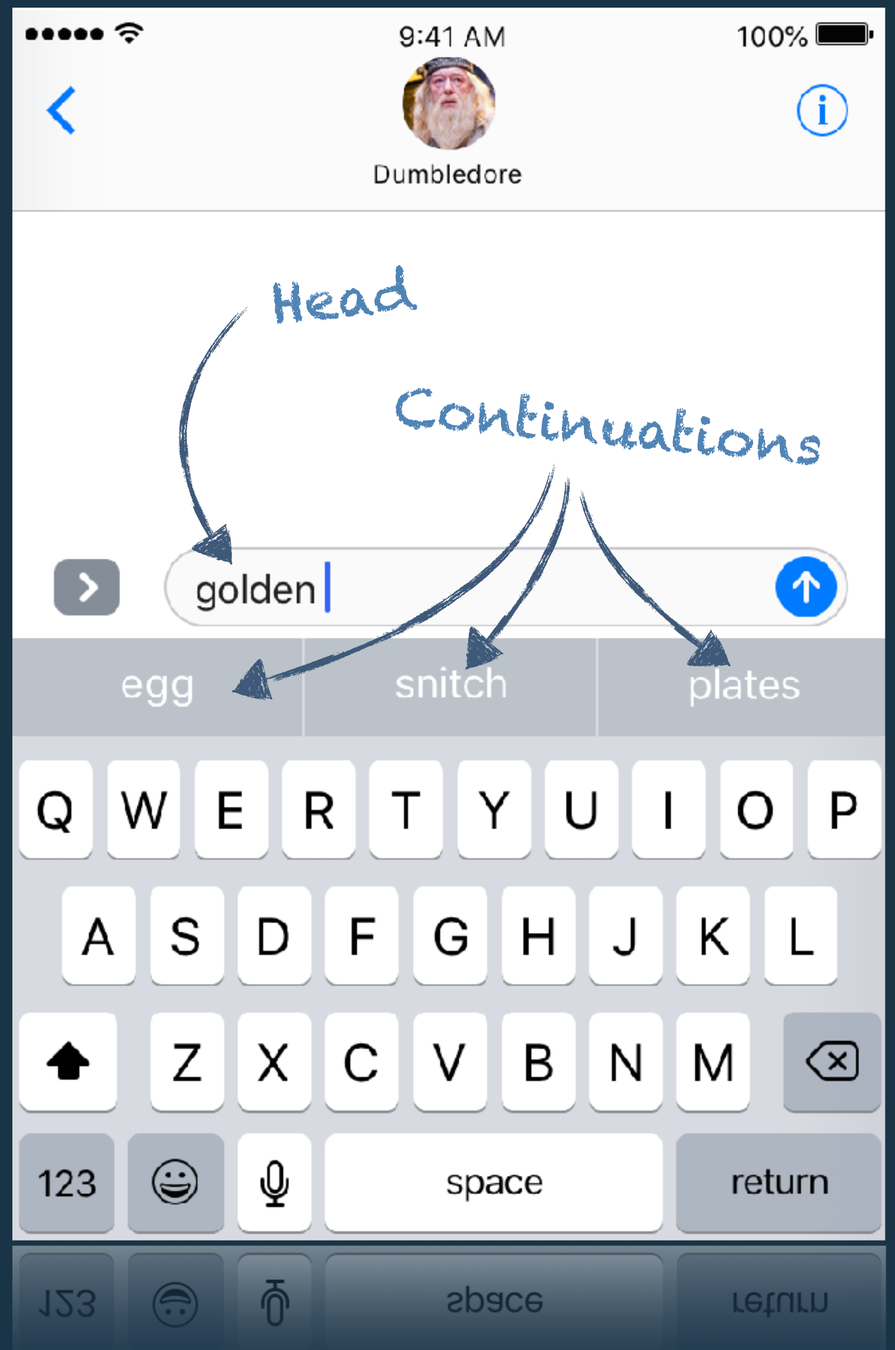
- “party” 30 times
- “cake” 20 times
- “wishes” 10 times



The world “golden” appears in the Harry Potter books 226 times.

After “golden”, J.K. used the word:

- “egg” 13 times
- “snitch” 11 times
- “plates” 10 times



The world “golden” appears in the Harry Potter books 226 times.

After “golden”, J.K. used the word:

- “egg” 13 times
- “snitch” 11 times
- “plates” 10 times

Key idea 3

Step 1

Learn

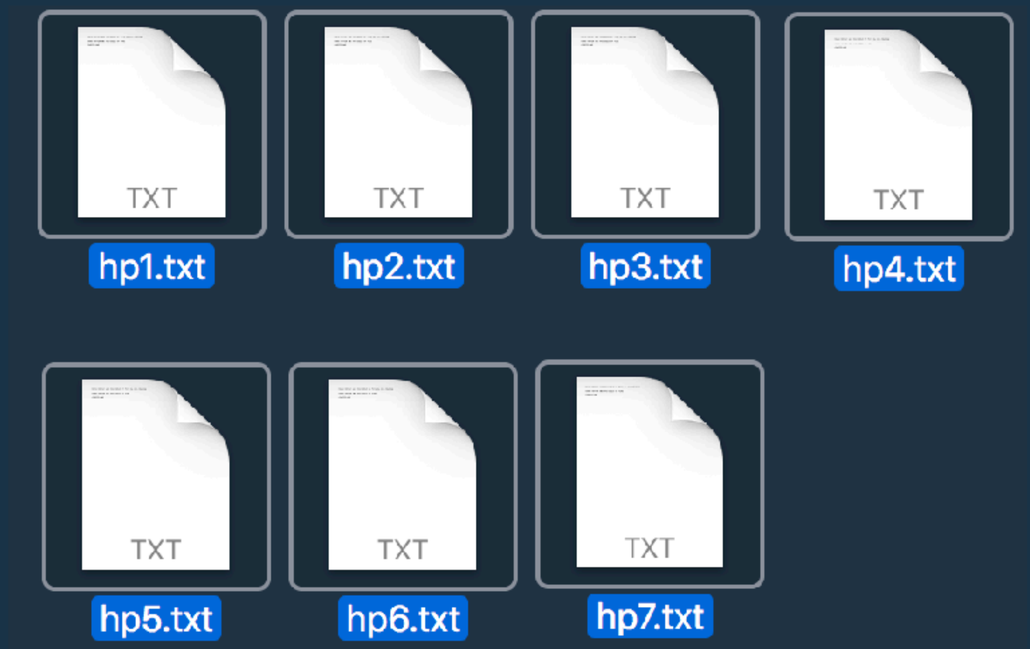
Step 2

Generate


golden
egg → 13
snitch → 11
plates → 10
light → 9
liquid → 1
goldfish
out → 1
any → 1
bowls → 1
above → 1
golf
balls → 2

21,814 words

```
{
  :golden => {
    :egg => 13,
    :snitch => 11,
    :plates => 10,
    :light => 9,
    :liquid => 1
  },
  :goldfish => {
    :out => 1,
    :any => 1,
    :of => 1,
    :bowls => 1
  },
  :golf => {
    :balls => 2
  }
}
```



alexpeattie.com/hp

```ruby
def tokenize(text)
  text.downcase.split(/[^a-z]+/).reject(&:empty?).map(&:to_sym)
end

"Mr. and Mrs. Dursley, of number four, Privet Drive,
were proud to say that they were perfectly normal"

[:mr, :and, :mrs, :dursley, :of, :number, :four, :privet, :drive,
:were, :proud, :to, :say, :that, :they, :were, :perfectly, :normal]
```

```
text = tokenize "The cat sat on the mat.
The cat was happy."

stats = {}

text.each_cons(2) do |head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end
```

```ruby
text = tokenize "The cat sat on the mat.
The cat was happy."

stats = {}

text.each_cons(2) do |head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end
```

```ruby
[:the, :cat]
```

head continuation

```ruby
{
  :the => {
    :cat => 1
  }
}
```

```ruby
text = tokenize "The cat sat on the mat.
The cat was happy."

stats = {}

text.each_cons(2) do |head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end
```

```ruby
[:cat, :sat]
```

head continuation

```ruby
{
  :the => {
    :cat => 1
  },
  :cat => {
    :sat => 1
  }
}
```

```ruby
text = tokenize "The cat sat on the mat.
The cat was happy."

stats = {}

text.each_cons(2) do |head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end
```

```ruby
{
  :the => {
    :cat => 2,
    :mat => 1
  },
  :cat => {
    :sat => 1,
    :was => 1
  },
  :sat => {
    :on => 1
  },
  :on => {
    :the => 1
  },
  :mat => {
    :the => 1
  },
  :was => {
    :happy => 1
  }
}
```

Step 1

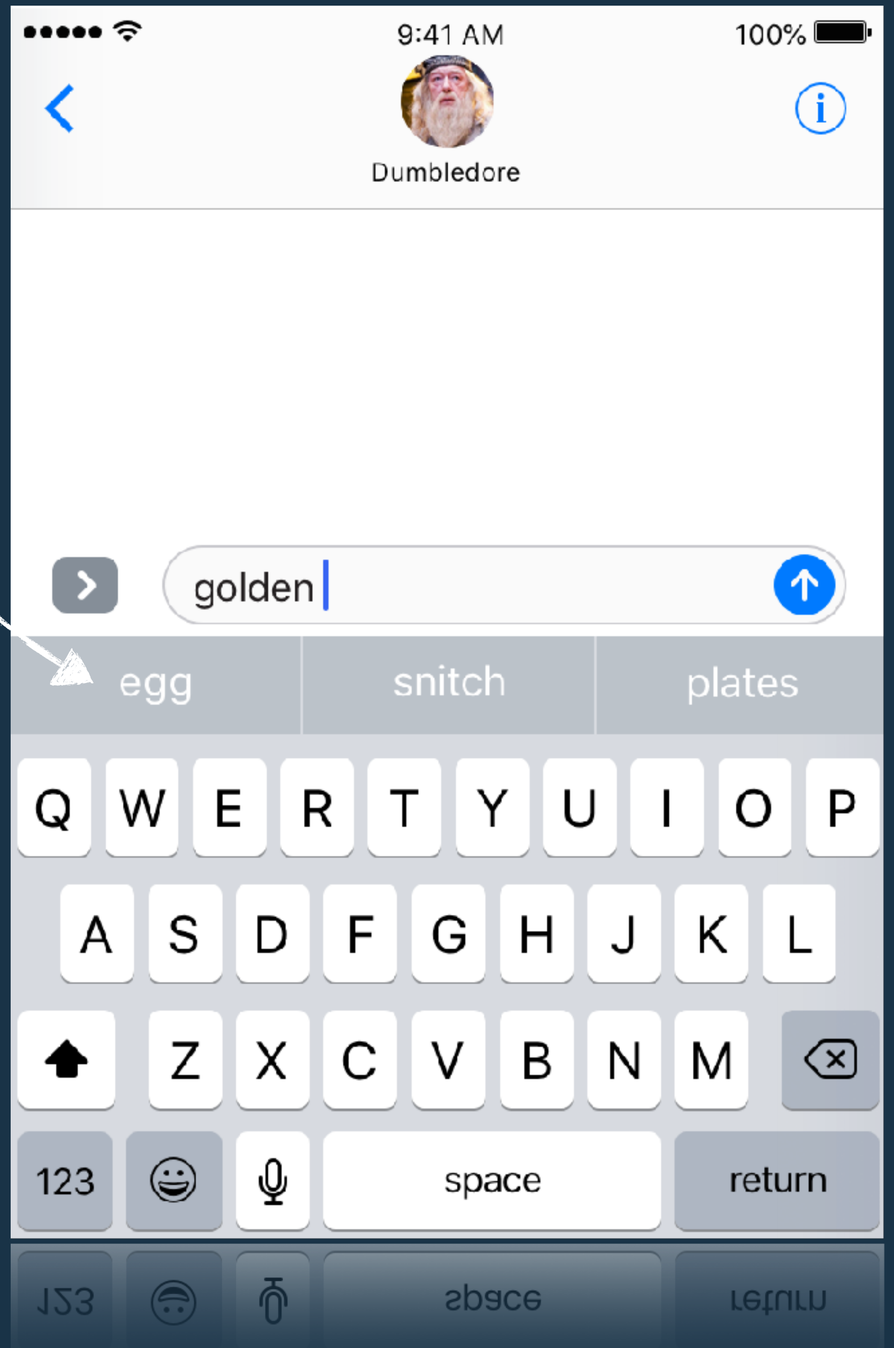
Learn ✅

Step 2

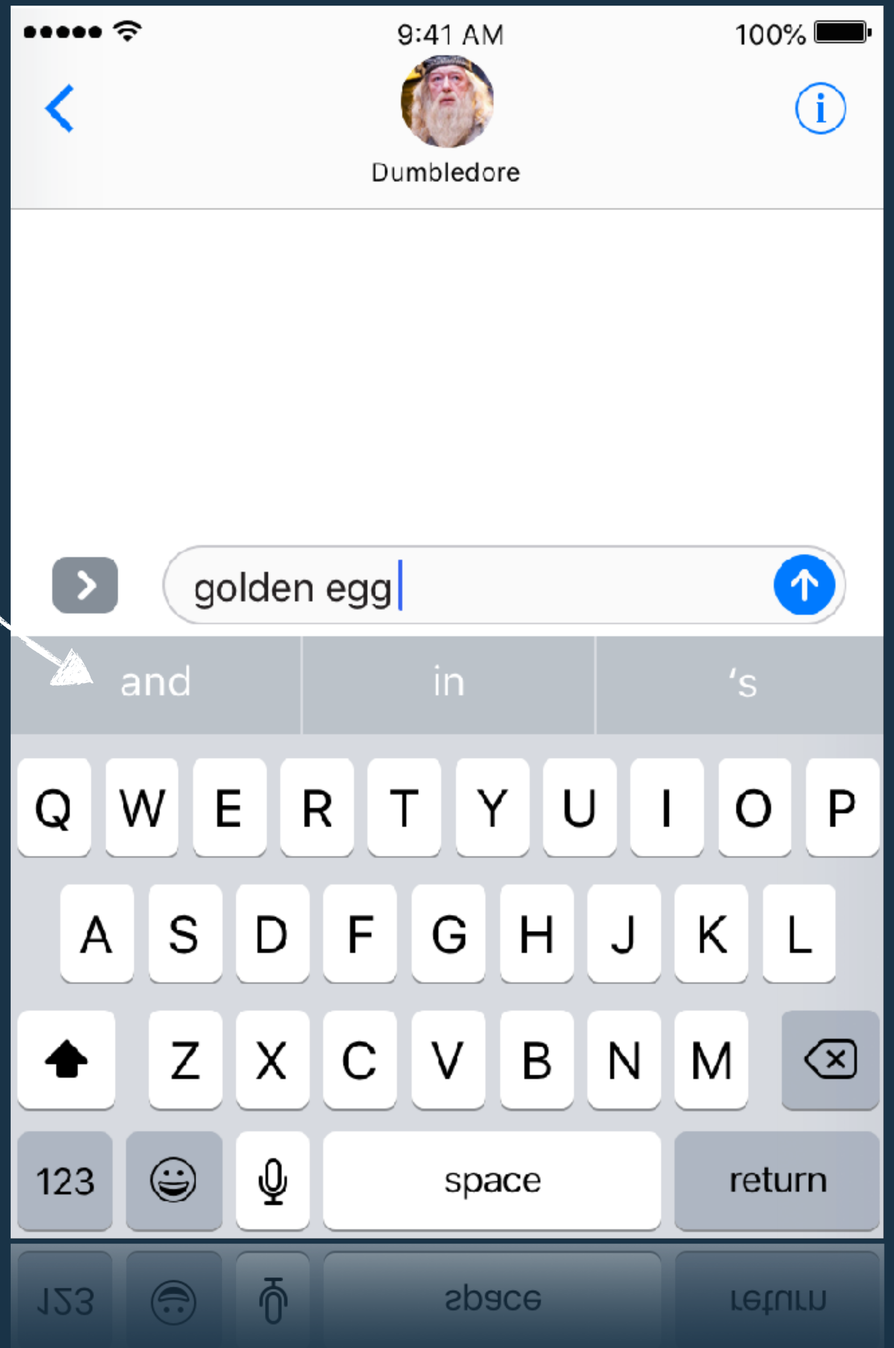
Generate

Greedy algorithm

Pick most frequent continuation



Pick most frequent continuation



```ruby
def pick_next_word_greedily(head)
  continuations = stats[head]
  chosen_word, count = continuations.max_by { |word, count| count }

  return chosen_word
end
```

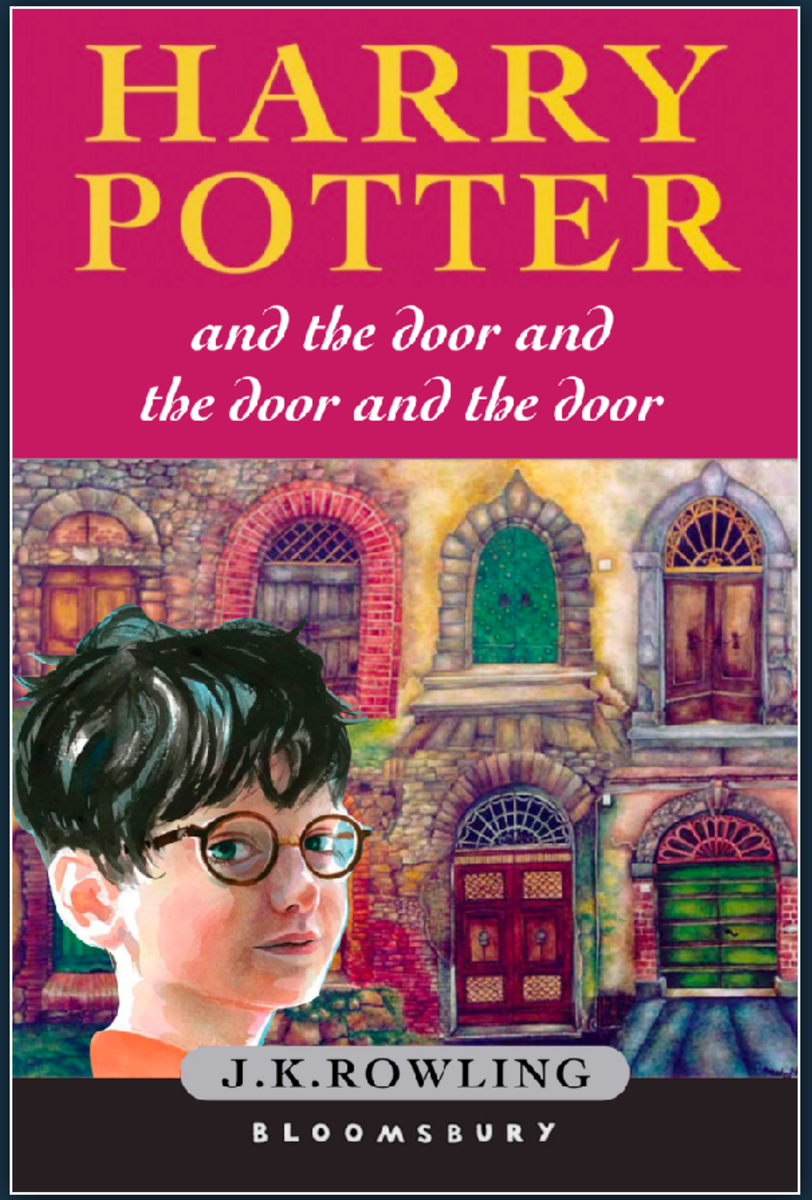
```ruby
story = [stats.keys.sample] # start with a random word from corpus

1.upto(50) do # 50 word story
  story << pick_next_word_greedily(story.last)
end
puts story.join(" ")
```

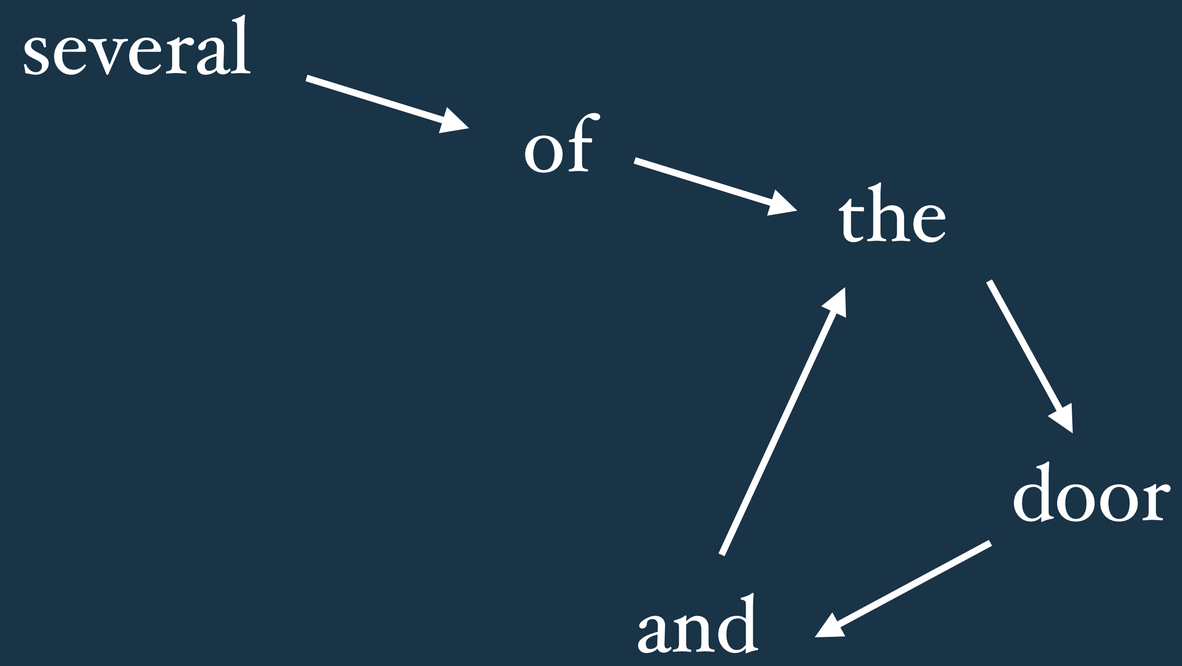
Drumroll....

“Oh no” said Harry. A few seconds later they were all the door and the door and the door and the door and the door.

Take two....

Surreptitiously, several of the door and the door and the door and the door and the door and the door and the door.


HARRY POTTER
and the door and the door and the door
J.K.ROWLING
BLOOMSBURY


several
of
the
door
and

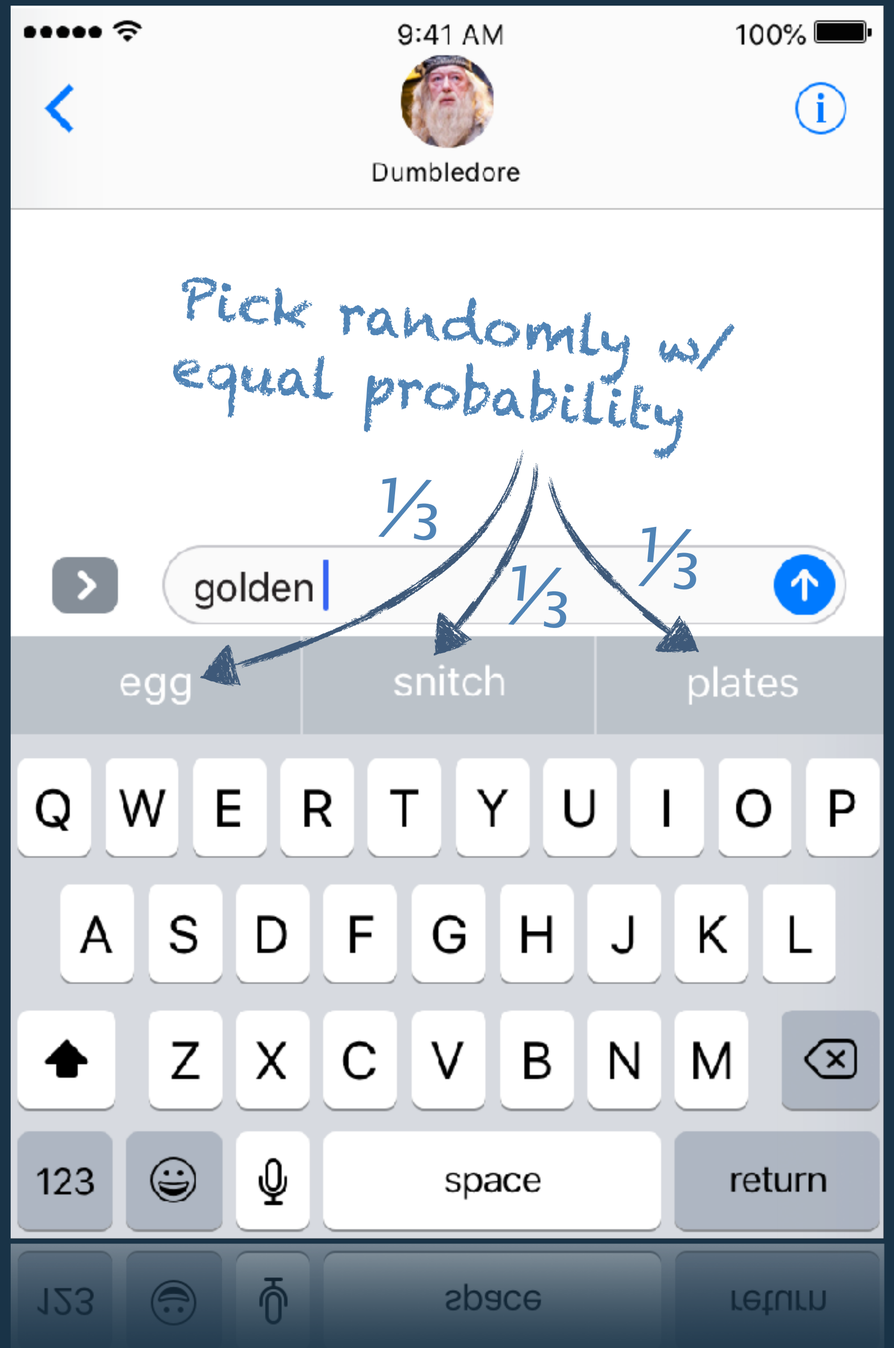
conference enchantingly nasty little more than ever since he was a few seconds later they were all the door and...

~~Greedy algorithm~~

Let’s get random

Uniform random algorithm


9:41 AM
100%
Dumbledore
Pick randomly w/ equal probability
golden
egg
snitch
plates
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return


9:41 AM
100%
Dumbledore
Pick randomly w/ equal probability
1/3
1/3
1/3
golden
egg
snitch
plates
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

egg → 1/117

snitch → 1/117

plates → 1/117

light → 1/117

⋮ 112 more

liquid → 1/117

```ruby
def pick_random_next_word(head)
  continuations = stats[head]
  return continuations.keys.sample
end
```

Debris from boys or accompany him bodily from Ron, yell the waters. Harry laughing together soon father would then bleated the smelly cloud.

What’s the problem?

house	elf	prices
	102 times	1 time
	~1/200 chance	~1/200 chance

Let’s get (a bit less) random

Weighted random algorithm

house	elf	prices
734 times	102 times	1 time
	~1/200 chance	~1/200 chance

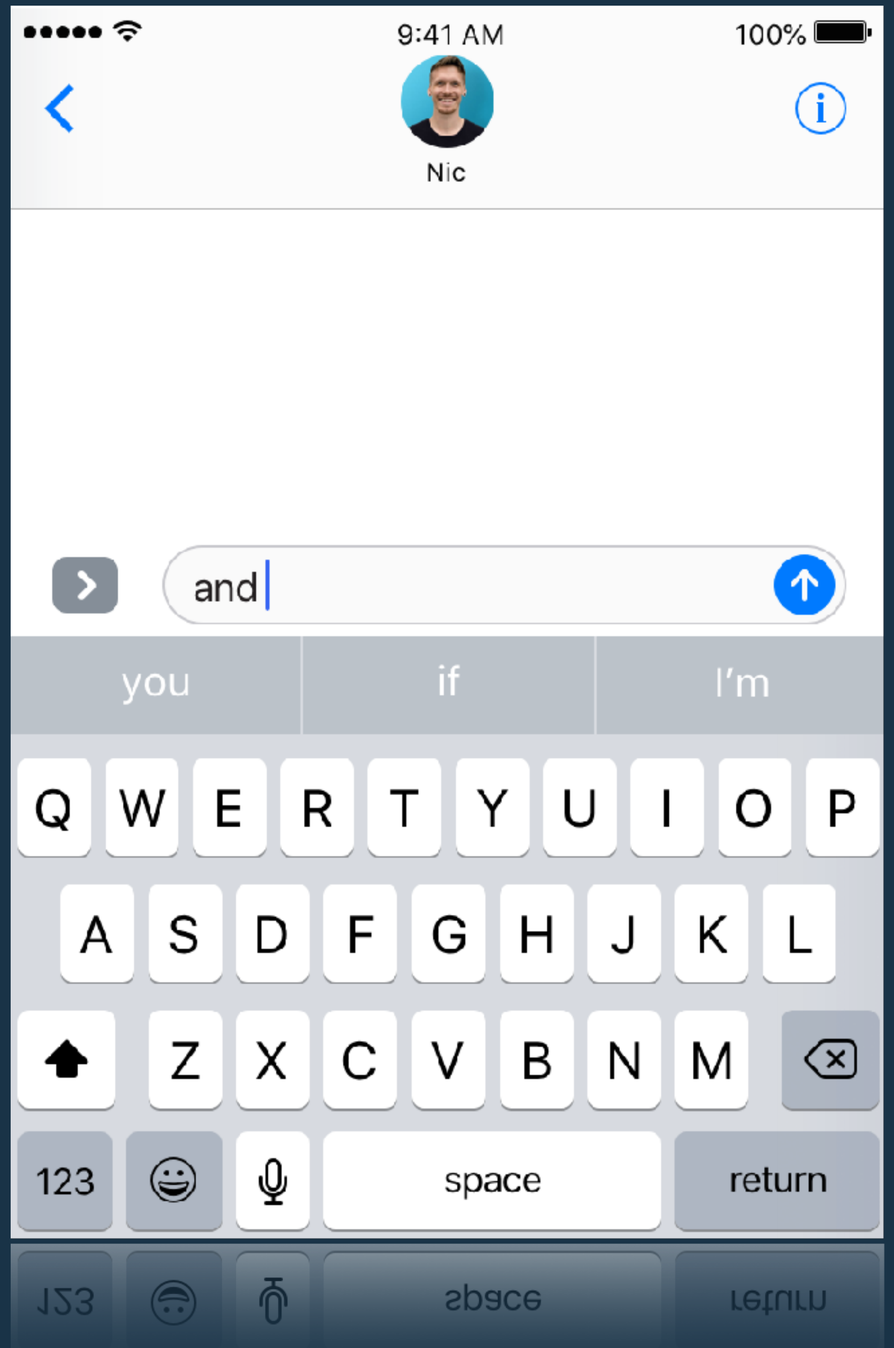
house	elf	prices
734 times	102 times	1 time
	~1/7 chance	~1/700 chance

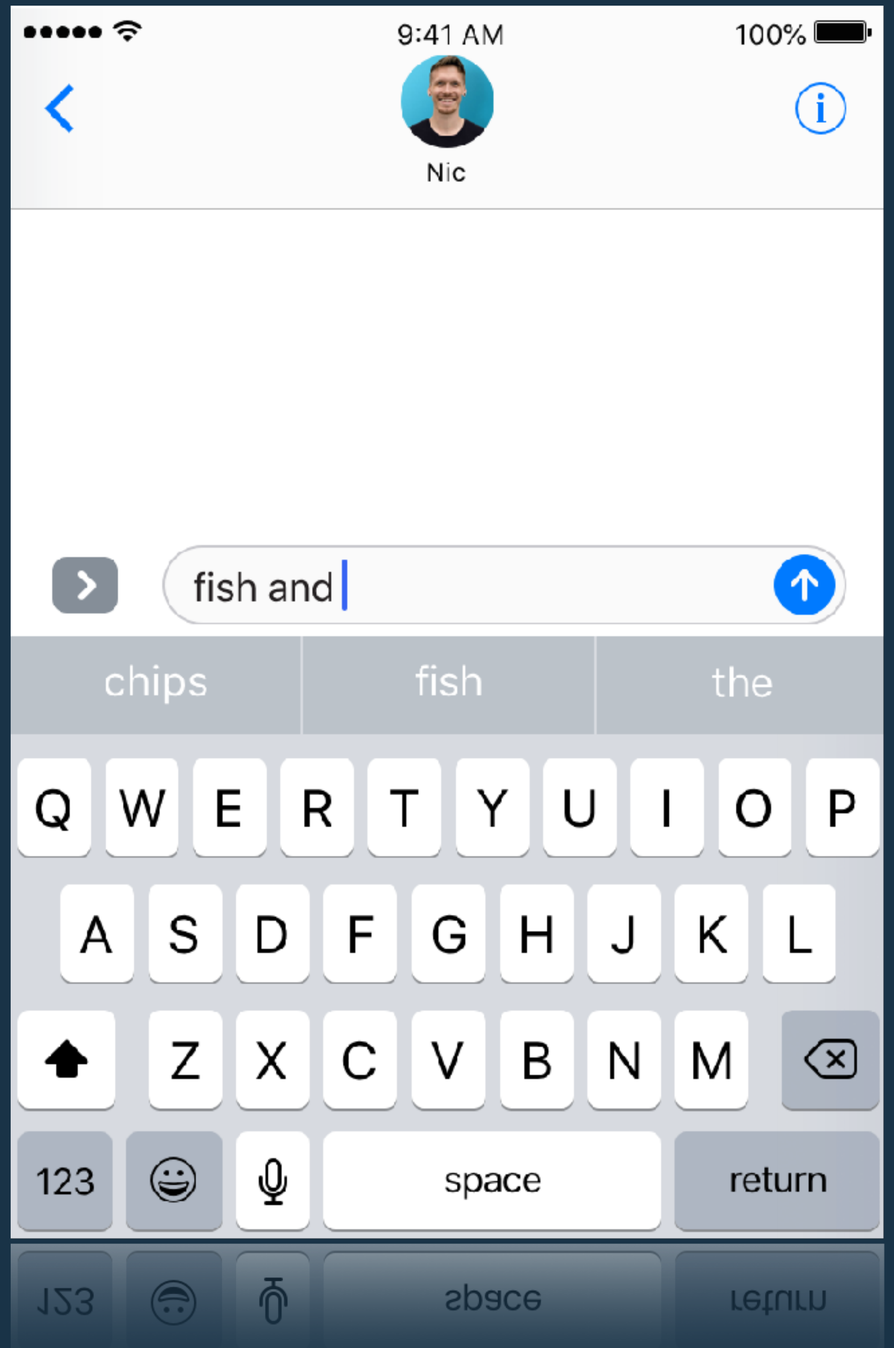

9:41 AM
100%
Dumbledore
Pick randomly w/ weighted probabilities
golden
1/2
1/3
1/6
egg
snitch
plates
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

```ruby
def pick_next_word_weighted_randomly(head)
  continuations = stats[head]
  continuations.flat_map { |word, count| [word] * count }.sample
end
```

Springing forward as though they had a bite of the hippogriff, he staggered blindly retorting Harry some pumpkin tart.

One last big idea...


9:41 AM
100%
Nic
and
you
if
I'm
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return


9:41 AM
100%
Nic
fish and
chips
fish
the
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

Key idea 4: Improve output by looking at more than just 1 previous word


```
{
  :golden => {
    :egg => 12,
    :snitch => 11,
    :plates => 10,
    :light => 9,
    :liquid => 1
  },
  :goldfish => {
    :out => 1,
    :any => 1,
    :of => 1,
    :bowls => 1
  },
  :golf => {
    :balls => 2
  }
}
```

bi·gram

two word


```
{
  [:golden, :egg] => {
    :harry => 1,
    :very => 1,
    :and => 2,
    :which => 1,
    :upstairs => 1,
    :does => 1,
    :he => 2,
    :said => 1,
    :still => 1,
    :fell => 1
  },
  [:golden, :snitch] => {
    :and => 1,
    :had => 1,
    :said => 1,
    :it => 1,
    :a => 1,
    :with => 1,
    :was => 1,
    :where => 1,
    :worked => 1
  }
}
```

tri·gram

three word

321,727 entries


```
stats = {}
n = 3

corpus.each_cons(n) do |*head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end
```

```ruby
stats = {}
n = 3

corpus.each_cons(n) do |*head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end
```

```ruby
[[:the, :cat], :sat]
```

head continuation

```ruby
{
  [:the, :cat] => {
    :sat => 1
  }
}
```

Normally when Dudley found his voice barely louder than before. “Dementors” said Dumbledore steadily, he however found all this mess is utterly worthless. Harry looked at him, put Slughorn into his bag more securely on to bigger and bigger until their blackness swallowed Harry whole and started emptying his drawers.

— trigram model

Neville, Seamus and Dean were muttering but did not speak when Harry had told Fudge mere weeks ago that Malfoy was crying, actually crying tears, streaming down the sides of their heads. “They revealed a spell to make your bludger” said Harry, anger rising once more.

— 4-gram model

```ruby
def tokenize(sentence)
  sentence.downcase.split(/[^a-z]+/).reject(&:empty?).map(&:to_sym)
end

def pick_next_word_weighted_randomly(head, stats)
  continuations = stats[head]
  continuations.flat_map { |word, count | [word] * count }.sample
end

text = tokenize(IO.read('hp.txt'))
stats = {}

n = 3
text.each_cons(n) do |*head, continuation|
  stats[head] ||= Hash.new(0)

  stats[head][continuation] += 1
end

story = stats.keys.sample

1.upto(50) do
  story << pick_next_word_weighted_randomly(story.last(n - 1), stats)
end
puts story.join(" ")
```

Key idea 1: Tell the story word by word

Key idea 2: Let's take inspiration from our phones

Key idea 3: Learn (stats about words and continuations), and generate (with weighted random algorithm)

Key idea 4: Improve output by looking at more than just 1 previous word

alexpeattie.com/hp