كتابة الحاسوب لتدريس ليسب

هذه المنشورة قد كتبت في الأصل باللغة الصينية ونشرت على □□□□، □□□□□:□□□□□□.□□□□.□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

أكبر جزء من الكود والفكر مبني على □□□□□ □□□□□□ □□□□□□138□□141.

المسألة: كيف يمكن ل الكمبيوتر أن يولد نصاً عشوائياً ولكن قابلاً للقراءة بناءً على نص إنجليزي؟ سبيل على المثال:

> تقدر الجمعية الوطنية للتمويل الاستثماري أن الثروة المرتبطة بصفقة كبيرة من الإنفاق من قبل القوانين
> التي ستنفق بعضها من البعض الأسباب الرئيسية لهذه المشاريع.

هذا النص العشوائي تم إنشاؤه بواسطة الكمبيوتر بعد تعلم بعض مقالات بول غراهام. يتوسع إلى جملة من الكلمة □□□□□ □□□□□. بشكل مفاجئ، فإن النص غالباً ما يكون قابلاً للقراءة.

الخوارزمية: تسجيل الكلمات التي تظهر بعد كل كلمة وعدد مرات ظهورها. على سبيل المثال، إذا ظهر □□ □□□□□□5 مرات في النص الأصلي و □□ □□□□□3 مرات، و لم□□□ يكن يظهر قبل أي كلمة، أخرى عندها عند إنشاء النص العشوائي، ظهور عند □□□هناك احتمال 5□8 اختيار □□□□□□□الكلمة. كالتالية إذا تم اختيار □□□□□□□، ثم فحص الكلمات التي ظهرت بعدها وكرر العملية.

الآن، دعونا حل المشكلة باستخدام □□□□.

يمكن أن تكون □□□ل من نوع الرموز تسجّل مختلف □□□□□الأ والعلامات التعبيرية، جيداً فسنستخدمها للتسجيل. سنستعمل الجدول الراسمي المتكامل لإنشاء القائمة:

```
(defparameter *words* (make-hash-table :size 10000))
```

كيف نقوم بإنشاء القائمة؟

```
(let ((prev '|.|))
  (defun see (sym)
    (let ((pair (assoc sym (gethash prev *words*))))
      (if pair
      (incf (cdr pair))
      (push (cons sym 1) (gethash prev *words*))))
    (setf prev sym)))
```

الكلمة الحالية هي المفتاح، وقائمة □□□□□ هي القيمة تحت ذلك المفتاح. على سبيل المثال، تحت □□□لدينا □ □□□□□□□□ . 5□ □□□□□□□ . 3□ □. إذا لم تكن الكلمة موجودة، ثم □□□□□ . 1□.

كيف نقوم بأخذ كلمة عشوائية؟

```
(defun random-word (word ht)
  (let* ((choices (gethash word ht))
     (x (random (reduce #'+ choices :key #'cdr))))
```

```
    (dolist (pair choices)
      (decf x (cdr pair))
      (if (minusp x)
      (return (car pair))))))
```

ذلك. بطريقة التالية وظيفة استخدام يتم هنا،

1. الجانبين؟ كل من جملة إلى معينة كلمة نمدد أن يمكن كيف نفكر دعونا الآن، أي معكوسة، قائمة على للحصول النص عكس [illegible].

2. والمعدلات السابقة الكلمات وجميع المفتاح، هي الأخيرة الكلمة حيث آخر، راسمي جدول على للحصول الراسمي الجدول عكس قائمة لكشت [illegible].

3. المعينة. الكلمة تظهر حتى عبارات عامة من الجملة تمديد في ابدأ حظك، حاول

الثانية: الطريقة استخدمت

```
(defparameter *r-words* (make-hash-table :size 10000))

(defun push-words (w1 w2 n)
  (push (cons w2 n) (gethash w1 *r-words*)))

(defun get-reversed-words () ; a cat -> cat a
  (maphash #'(lambda (k lst)
           (dolist (pair lst)
         (push-words (car pair) k (cdr pair))))
       *words*))
```

الجمل لإنتاج الكود هو هنا معكوساً. ترتيبهم مع آخر راسمي جدول في الكلمات من زوج كل إدراج ثم الأصلي، الراسمي الجدول عبور تلقائياً: المتسقة

```
(defparameter *words* (make-hash-table :size 10000))
(defconstant maxword 100)
(defparameter nwords 0)
(defconstant debug nil)
(let ((prev '|.|))
  (defun see (sym)
    (incf nwords)
    (let ((pair (assoc sym (gethash prev *words*))))
      (if pair
      (incf (cdr pair))
      (push (cons sym 1) (gethash prev *words*))))
```

```
    (setf prev sym)))

(defun check-punc (c) ; char to symbol
  (case c
    (#\. '|.|) (#\, '|,|)
    (#\; '|;|) (#\? '|?|)
    (#\: '|:|) (#\! '|!|)))

(defun read-text (pathname)
  (with-open-file (str pathname :direction :input)
    (let ((buf (make-string maxword))
      (pos 0))
      (do ((c (read-char str nil 'eof)
          (read-char str nil 'eof)))
      ((eql c 'eof))
    (if (or (alpha-char-p c)
        (eql c #\'))
        (progn
          (setf (char buf pos) c)
          (incf pos))
        (progn
          (unless (zerop pos)
        (see (intern (subseq buf 0 pos)))
        (setf pos 0))
          (let ((punc (check-punc c)))
        (if punc
            (see punc)))))))))

(defun print-ht (ht)
  (maphash #'(lambda (k v)
        (format t "~A ~A~%" k v))
         ht))

(defparameter *r-words* (make-hash-table :size 10000))

(defun push-words (w1 w2 n)
  (push (cons w2 n) (gethash w1 *r-words*)))

(defun get-reversed-words () ; a cat -> cat a
  (maphash #'(lambda (k lst)
```

```
          (dolist (pair lst)
            (push-words (car pair) k (cdr pair))))
        *words*))

(defun print-a-word (word ht)
  (maphash #'(lambda (k lst)
          (if (eql k word)
          (format t "~A ~A~%" k lst)))
        ht))

(if debug
    (print-a-word '|leave| *r-words*))

(defun punc-p (sym) ; symbol to char, nil when fails.
  (check-punc (char (symbol-name sym) 0)))

(defun random-word (word ht)
  (let* ((choices (gethash word ht))
     (x (random (reduce #'+ choices :key #'cdr))))
    (dolist (pair choices)
      (decf x (cdr pair))
      (if (minusp x)
      (return (car pair))))))

(defun gen-former (word str)
  (let ((last (random-word word *r-words*)))
    (if (not (punc-p last))
    (progn
      (gen-former last str)
      (format str "~A " last)))))

(defun gen-latter (word str)
  (let ((next (random-word word *words*)))
    (format str "~A " next)
    (if (not (punc-p next))
        (gen-latter next str))))

;(gen-latter '|leave| t)

(defun get-a-word (ht) ; get a random word
```

```
  (let ((x (random nwords)))
    (maphash #'(lambda (k v)
         (dolist (pair v)
           (decf x (cdr pair))
           (if (minusp x)
               (return-from get-a-word (car pair)))))
         ht)))
;(get-a-word *words*)
(defun gen-sentence (word str)
  (gen-former word str)
  (format str "~A " word)
  (gen-latter word str))

(defun test ()
  (setf nwords 0)
  (read-text "essay.txt")
  (get-reversed-words)
  (let ((word (get-a-word *words*)))
    (print word)
    (gen-sentence word t)))
(test)
```