Training a Markov Process on Pride and Prejudice

Justin Pearson
Oct 26, 2014

In[1]:=
```
Clear["Global`*"]
SetDirectory[NotebookDirectory[]];
```

For simplicity, we'll just chop up the tokens by whitespace and keep all the punctuation. Consequently the word "Elizabeth" will be different from the word "Elizabeth," (with a comma). Fine.

In[3]:=
```
t = ExampleData[{"Text", "PrideAndPrejudice"}, "Words"]
Length[t]
```

Out[3]=
```
{Chapter, 1, It, is, a, truth, universally, acknowledged,, that, a, single,
 man, in, possession, of, a, good, fortune,, must, be, in, want, of, a,
 wife., However, little, known, the, feelings, or, views, of, such, a,
 man, may, be, on, his, first, entering, a, neighbourhood,, this, truth,
 is, so, ··· 121 457 ···, from, the, city., With, the, Gardiners,, they,
 were, always, on, the, most, intimate, terms., Darcy,, as, well, as,
 Elizabeth,, really, loved, them;, and, they, were, both, ever, sensible,
 of, the, warmest, gratitude, towards, the, persons, who,, by, bringing,
 her, into, Derbyshire,, had, been, the, means, of, uniting, them.}
```

large output | show less | show more | show all | set size limit...

Out[4]= 121 553

Enumerate the unique words

In[5]:=
```
uniq = DeleteDuplicates[t];
Short[uniq]
n = Length[uniq]
```

Out[6]//Short=
```
{Chapter, 1, It, is, a, truth, universally, acknowledged,,
  that, single, <<13 046>>, end., persuasion,, offence,,
  reconciliation;, pollution, mistress,, city., terms., uniting}
```

Out[7]= 13 065

Here's an association to convert from words to their index in the unique-word list:

In[8]:=
```
With[{rules = MapThread[Rule, {uniq, Range[n]}]},
 ass = Association[rules];
 (* much faster to apply an Ass than a list of Rules *)
 unass = Association[Reverse /@ rules];
]
Short[ass]
Short[unass]
```

Out[9]//Short=
```
⟨| Chapter → 1, 1 → 2, It → 3, is → 4, a → 5, truth → 6, universally → 7,
  <<13 051>> , offence, → 13 059, reconciliation; → 13 060, pollution → 13 061,
  mistress, → 13 062, city. → 13 063, terms. → 13 064, uniting → 13 065 |⟩
```

Out[10]//Short=
```
⟨| 1 → Chapter, 2 → 1, 3 → It, 4 → is, 5 → a, 6 → truth, 7 → universally,
  <<13 051>> , 13 059 → offence,, 13 060 → reconciliation;, 13 061 → pollution,
  13 062 → mistress,, 13 063 → city., 13 064 → terms., 13 065 → uniting |⟩
```

In[11]:=
```
ass /@ StringSplit@"Mary went into the house to get an axe"
```

Out[11]=
```
{581, 5237, 255, 24, 145, 57, 361, 268, Missing[KeyAbsent, axe]}
```

In[12]:=
```
unass /@ {581, 5237, 255, 24, 145, 57, 361, 268}
```

Out[12]=
```
{Mary, went, into, the, house, to, get, an}
```

The text of *Pride and Prejudice*, converted to word indices:

In[13]:=
```
x = ass /@ t;
Short[x]
```

Out[14]//Short=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 5, 10, 11, 12, 13, 14, 5, 15, 16, 17, 18, 12, 19, 14, 5, 20, 21,
  22, 23, <<121 498>> , 1718, 84, 280, 707, 1839, 650, 2298, 14, 24, 12 916, 1758,
  735, 24, 3133, 2026, 119, 6841, 241, 255, 4875, 70, 82, 24, 1370, 14, 13 065, 271}
```

In[15]:=
```
Grid[{t, x}ᵀ[[1 ;; 40]], Frame → All]
```

Out[15]=

Chapter	1
1	2
It	3
is	4
a	5
truth	6
universally	7
acknowledged,	8
that	9
a	5
single	10
man	11
in	12
possession	13
of	14
a	5
good	15
fortune,	16
must	17
be	18
in	12
want	19
of	14
a	5
wife.	20
However	21
little	22
known	23
the	24
feelings	25
or	26
views	27
of	14
such	28
a	5
man	11
may	29
be	18
on	30
his	31

Method 1: Build our own first-order Markov chain

Build the adjacency matrix for our discrete markov process:

In[16]:=
```
times = {};
A = SparseArray[{}, {n, n}];

(* Go through the text. For each word pair,
increment the appropriate entry in the adjacency matrix. *)
AppendTo[times, Now];
For[i = 1, i ≤ Length[x] - 1, i++, A[[Sequence @@ x[[{i, i + 1}]]]]++]
AppendTo[times, Now];

(* Normalize each row to sum to 1 to make it a true pmf. *)
For[j = 1, j ≤ Length[A], j++, A[[j]] = N[A[[j]] / Total[A[[j]]]]];
AppendTo[times, Now];
Differences[times]
```

Out[23]= { 12.5302 s , 1.10767 min }

In[24]:=
```
A
```

Out[24]= SparseArray[]

In[25]:=
```
MinMax[A]
```

Out[25]= {0, 1.}

In[26]:=
```
ByteCount[A]
```

Out[26]= 2 217 824

In[27]:=
```
next[s_] := RandomChoice[A[[ass[s]]] → uniq]
```

In[28]:=
```
next["Mary"]
```

Out[28]= very

In[29]:=
```
NestList[next, "Elizabeth", 200] // StringRiffle
```

Out[29]=
```
Elizabeth coloured, and grateful to Miss Elizabeth listened most abundantly
  regret that the business, too! How strange it incumbent on Mr. Bennet
  followed this comfort to himself to matrimony, in my old housekeeper, who
  were engaged Mr. Bennet had once sit in distressed to the horror of talking,
  my dear uncle had reached the study of importance which gave a young woman
  whom it is really believed to have had. My father had been able," said
  Bingley; what motives has no longer importune my godfather, and useless
  to have no questions." "Thank you--but are over-scrupulous, surely. I saw
  them to be much to remain the principal, if he was shocked. Lydia went by
  the Little more gentlemanlike manner." Kitty, I am sure, was satisfied
  with the words, or disgust of my family. Everybody said too hasty, sir,"
  was involved in the honour you were violently caring no suspicion, the
  former interference in all that would have been written with the account
  and their kind as Mr. Bennet, beyond the same time for Jane," said she,
  "as to wait till you speak to join in a week, and now, when they would
  have another disappointment. Her answer, and sometimes convey much the
```

Method 2: Use Mathematica's build-in DiscreteMarkovProcess

We can use Mathematica's built-in functions to train a discrete Markov process from our text. Unfortunately, it's quite slow for datasets of this size, so we use a subset of the text.

Here we use Mathematica's "EstimatedProcess" function to estimate the weights of an *n*-state discrete Markov chain:

In[30]:=
```
t2 = t[[ ;; 1000]];
uniq2 = DeleteDuplicates[t2];
n2 = Length[uniq2];
With[{rules = MapThread[Rule, {uniq2, Range[n2]}]},
  ass2 = Association[rules];
  unass2 = Association[Reverse /@ rules];
 ];
x2 = ass2 /@ t2;
{time, dmp} = AbsoluteTiming[EstimatedProcess[x2, DiscreteMarkovProcess[n2]]];
time
```

Out[36]=
```
6.471
```

Let's run the discrete markov process for 200 timesteps:

In[37]:=
```
SeedRandom[1234];
RandomFunction[dmp, {1, 200}]["Values"] // Map[unass2] // StringRiffle
```

Out[38]=
```
2 Mr. Bennet," said his second daughter employed in love with one of
  three-and-twenty years at least." "Ah, you talk so! But it will not
  pretend to be sure! A single man in vexing me. You take possession
  before Michaelmas, and live to be for as Jane, nor half so odd a wife.
  However little known the place, and that Netherfield is not go; and I
  have none of her daughters married; its solace was less difficult to
  introduce him." "I do not." "You are as he is a chaise and caprice,
  that account, for my little information, and some one of her own. She
  was so odd a wife. However little known the rightful property of her
  daughters married; its solace was then disclosed in trimming a year come
  into the earliest of them, and therefore you must know what I have no
  such thing. She is let at least." "Ah, you must go, merely on that he
  agreed with Mr. Bennet made no answer. "Do you not half so odd a single
  man in trimming a woman of England; that Netherfield is let at least."
  "Ah, you are as Jane, nor half so good-humoured as he suddenly addressed
```

Appendix: Word-frequency analysis

Wordcounts often follow a Zipf distribution: “The top 10% most-frequent words account for 90% of all words.” Let’s see if *Pride and Prejudice* follows this trend.

Wikipedia:

https://en.wikipedia.org/wiki/Zipf%27s_law

Zipf’s law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.: the rank-frequency distribution is an inverse relation. For example, in the Brown Corpus of American English text, the word “the” is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million). True to Zipf’s Law, the second-place word “of” accounts for slightly over 3.5% of words (36,411 occurrences), followed by “and” (28,852). Only 135 vocabulary items are needed to account for half the Brown Corpus.[4]

Zipf’s law is most easily observed by plotting the data on a log-log graph, with the axes being log (rank order) and log (frequency). For example, the word “the” (as described above) would appear at x = log(1), y = log(69971). It is also possible to plot reciprocal rank against frequency or reciprocal frequency or interword interval against rank.[1] The data conform to Zipf’s law to the extent that the plot is linear.

In[39]:=
```
wordcounts = Counts[t] // Normal;
Short[wordcounts]
```

Out[40]//Short=
```
{Chapter → 61, 1 → 1, It → 198, is → 781, a → 1891, truth → 16, universally → 3,
 acknowledged, → 7, <<13 050>>, offence, → 1, reconciliation; → 1,
 pollution → 1, mistress, → 1, city. → 1, terms. → 1, uniting → 1}
```

What are the rarest words? What are the most common?

In[41]:=
```
sortedwordcounts = wordcounts ~ SortBy ~ Last;
```

In[42]:=
```
(sortedwordcounts /. {s_String :> InputForm[s]}) // Short
```

Out[42]//Short=
```
{"____" → 1, "1" → 1, "10" → 1, "11" → 1, "12" → 1, "13" → 1, "14" → 1, "15" → 1,
 "15th" → 1, "16" → 1, "17" → 1, "18" → 1, "18th," → 1, "19" → 1, "2" → 1,
 "2." → 1, "20" → 1, "21" → 1, "22" → 1, "23" → 1, "24" → 1, <<13 023>>,
 "you" → 937, "it" → 942, "for" → 982, "with" → 993, "he" → 1037, "as" → 1109,
 "had" → 1127, "his" → 1168, "be" → 1191, "she" → 1306, "not" → 1337,
 "that" → 1405, "I" → 1739, "in" → 1758, "was" → 1795, "her" → 1858,
 "a" → 1891, "and" → 3240, "of" → 3552, "the" → 4047, "to" → 4051}
```

(The numbers on the left are chapter numbers.)

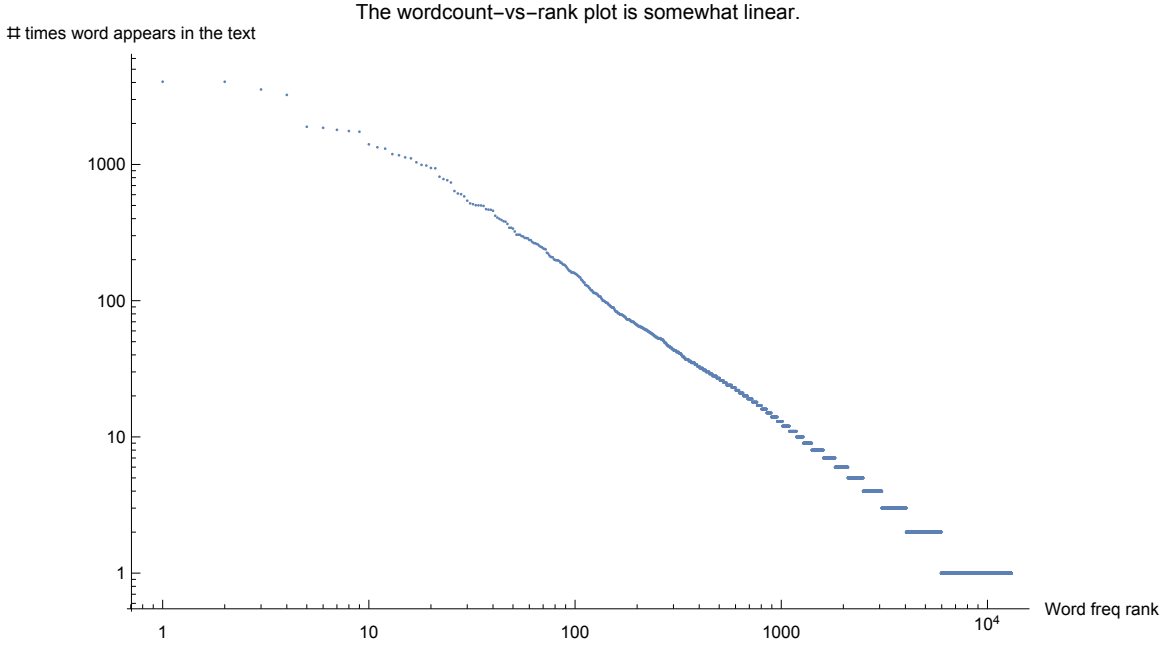
Plot log-log to assess eligibility for Zipf distribution

Let's plot log-rank and log-count as described above, so the word "to" would appear at x = log(1), y = log(4051), "the" would appear at x = log(2), y = log(4047), etc.

In[43]:=
```
counts = Last /@ sortedwordcounts;
deccounts = Reverse@counts;
ranks = Range@Length@deccounts;
pts = {ranks, deccounts}^T;
Short[pts]
ListLogLogPlot[pts,
 AxesLabel → {"Word freq rank", "# times word appears in the text"},
 PlotLabel → "The wordcount-vs-rank plot is somewhat linear.", ImageSize → 600]
```

Out[47]//Short=
```
{{1, 4051}, {2, 4047}, {3, 3552}, {4, 3240}, {5, 1891}, {6, 1858}, {7, 1795},
  {8, 1758}, {9, 1739}, {10, 1405}, <<13 046>>, {13 057, 1}, {13 058, 1}, {13 059, 1},
  {13 060, 1}, {13 061, 1}, {13 062, 1}, {13 063, 1}, {13 064, 1}, {13 065, 1}}
```

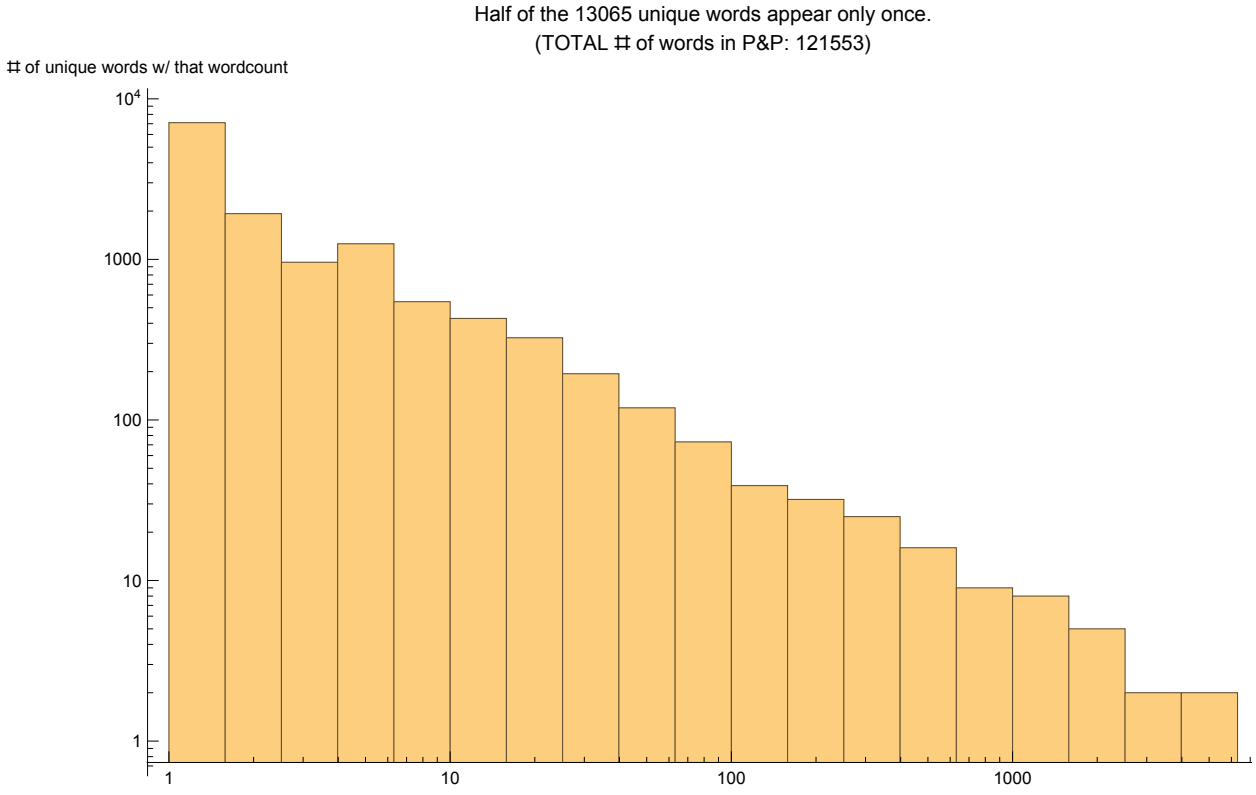
Out[48]=



Looks pretty linear in the middle, not so great at the ends. What's up with the long tail -- all the words that only appear once?

In[49]:=
```
Histogram[counts, "Log", "LogCount", AxesLabel →
  {"wordcount (# times a word occurs)", "# of unique words w/ that wordcount"},
 PlotLabel → "Half of the " <> ToString[Length@counts] <>
   " unique words appear only once.\n(TOTAL # of words in P&P: " <>
   ToString[Length@t] <> ")", ImageSize → 800]
```

Out[49]=



A lot of words occur only once:

In[50]:=
```
Count[wordcounts, HoldPattern[_ → 1]]
```

Out[50]= 7104

It is mainly due to us not erasing punctuation:

In[51]:=
```
Cases[wordcounts, HoldPattern[_ → 1]]
```

Out[51]=
```
{1 → 1, rightful → 1, last?" → 1, Morris → 1, Michaelmas, → 1, week." → 1, name?" → 1,
 "Bingley." → 1, single?" → 1, Single, → 1, tiresome! → 1, here?" → 1,
 "Design! → 1, Nonsense, → 1, comes." → 1, grown-up → 1, beauty." → 1, cases, → 1,
 newcomers. → 1, _us_ → 1, over-scrupulous, → 1, surely. → 1, preference." → 1,
 _can_ → 1, vexing → 1, least." → 1, "Ah, → 1, suffer." → 1, ··· 7049 ··· ,
 cheap → 1, ought. → 1, her's → 1, profession. → 1, Bath; → 1, overcome, → 1,
 retain → 1, dropt → 1, resentment; → 1, heretofore, → 1, arrear → 1, see. → 1,
 sportive, → 1, He, → 1, pleasantry. → 1, instructions, → 1, liberties → 1,
 abusive, → 1, end. → 1, persuasion, → 1, offence, → 1, reconciliation; → 1,
 pollution → 1, mistress, → 1, city. → 1, terms. → 1, uniting → 1}
```

large output | show less | show more | show all | set size limit...

How many words account for half the text?

In[52]:=
```
LengthWhile[Accumulate@deccounts, # < Length[t] / 2 &]
```

Out[52]= 84

In[53]:=
```
Total@deccounts[[ ;; 84]]
```

Out[53]= 60 714

Wow, one could write half of Pride and Prejudice with only 84 words!

Fit a Zipf dist to it.

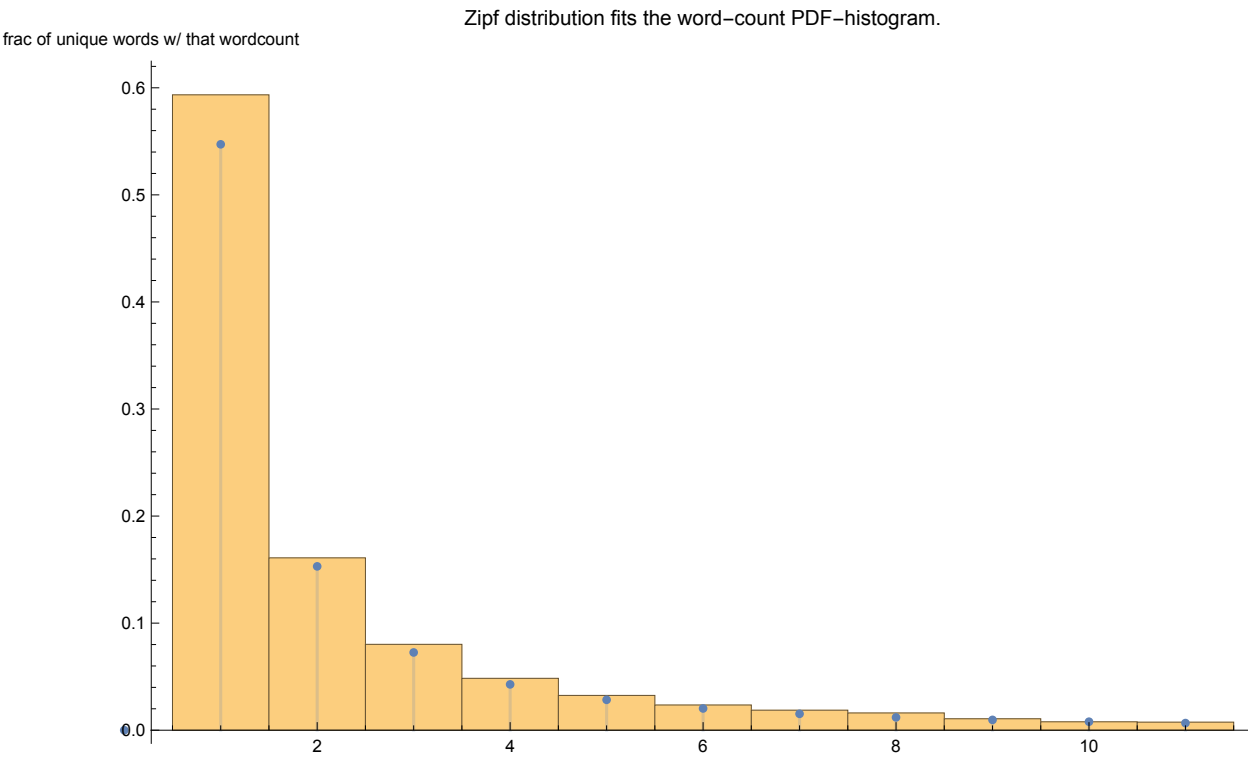
In[54]:=
```
edist = EstimatedDistribution[counts, ZipfDistribution[ρ]]
```

Out[54]= ZipfDistribution[0.838398]

The estimated distribution seems to line up well with the histogram of wordcounts:

In[55]:=
```
Show[Histogram[counts, {0.5, 11.5, 1}, "PDF"],
 DiscretePlot[PDF[edist, x], {x, 0, 11}, PlotStyle → PointSize[Medium]], AxesLabel →
  {"wordcount (# times a word occurs)", "frac of unique words w/ that wordcount"},
 PlotLabel → "Zipf distribution fits the word-count PDF-histogram.",
 ImageSize → 800]
```

Out[55]=



With this, you can answer questions like: What's the probability that a word occurs more than 100 times in the text?

In[56]:=
```
Probability[X ≥ 100, X ≈ edist]
```

Out[56]= 0.0137952

Hopefully this matches the fraction of the unique words that occur more than 200 times:

In[57]:=
$$\frac{\texttt{Count[counts, x\_ /; x ≥ 100]}}{\texttt{Length[counts]}}\ \texttt{// N}$$

Out[57]= 0.0105626

Not too bad.

In[58]:=
```
Speak["End of file."]
```