

# Markov Jabberwocky: Through the Sporking Glass

John Kerl

~~Department of Mathematics, University of Arizona~~ Two Sigma Investments

~~August 26, 2009~~ January 25, 2012

## Unnatural Language Processing for the Uninitiated:

~ Why, and what ~

~ The abstract how ~

~ The concrete how: back to words! ~

~ Results, and a little (but not too much) head-scratching ~

~ Some applications, and conclusion ~

## Why

I finished **grad school** in May 2010 and started work at **Two Sigma** in June 2010. The summer before that, I was hard at work<sup>1</sup> writing my dissertation, and beginning to put the **Big Job Search** into gear.

I've always been enchanted by Lewis Carroll's *Jabberwocky*, including a few translations; foreign languages have, as well, also fascinated me as long as I can remember<sup>2</sup>. Moreover, *Jabberwocky* is only 28 lines long; one is left **wanting more**. At some point, I realized that Markov-chain techniques might give me a tool to explore creating more **not-quite-words**.

Results:

- It works, well enough.
- It has some power to classify written utterances in various languages.
- Really, though, it was just a two-day lark project. Then I went back to more serious work (such as finding a job).

---

<sup>1</sup>While playing online Scrabble, have you ever checked (hoping-hoping-hoping) that *motch*, *say*, or *filious*, or *helving*, was some rare but legitimate English word? (One of those three is.)

<sup>2</sup>Then I became a programmer and realized I could make a living learning new languages. Groovy, man!

## What: Lewis Carroll's *Jabberwocky* / *le Jaseroque* / *der Jammerwoch*

'Twas brillig, and the slithy toves  
Did gyre and gimble in the wabe;  
All mimsy were the borogoves,  
And the mome raths outgrabe.

«Garde-toi du Jaseroque, mon fils!  
La gueule qui mord; la griffe qui prend!  
Garde-toi de l'oiseau Jube, évite  
Le frumieux Band-à-prend!»

*Er griff sein vorpals Schwertchen zu,  
Er suchte lang das manchsam' Ding;  
Dann, stehend unterm Tumtum Baum,  
Er an-zu-denken-fing. . . .*

Many of the above words do not belong to their respective languages — yet look like they *could*, or *should*. It seems that each language has its own **periphery of almost-words**. Can we somehow capture a way to generate words which look Englishy, Frenchish, and so on?

It turns out **Markov chains** do a pretty good job<sup>3</sup> of it. Let's open up that particular black box and see how it works.

<sup>3</sup>The method Carroll used for some of his neologies was the *portmanteau*, the packing or splicing together, of pairs of words: the same process gives us *bromance* and *spork*.

## The abstract how

## Probability spaces (the first of a half-dozen mathy slides)

A **probability space**\* is a set  $\Omega$  of possible **outcomes**\*\*  $X$ , along with a **probability measure**  $P$ , mapping from **events** (sets of outcomes) to numbers between 0 and 1 inclusive. Example:  $\Omega = \{1, 2, 3, 4, 5, 6\}$ , the results of the toss of a (fair) die.

What would you want  $P(\{1\})$  to be? Given that, what about  $P(\{2, 3, 4, 5, 6\})$ ? And of course, we want  $P(\{1, 2\}) = P(\{1\}) + P(\{2\})$ .

The axioms for a probability measure encode that intuition. For all  $A, B \subseteq \Omega$ :

- $P(A) \in [0, 1]$  for all  $A \subseteq \Omega$
- $P(\Omega) = 1$
- $P(A \cup B) = P(A) + P(B)$  if  $A$  and  $B$  are disjoint.

Any function  $P$  from subsets of  $\Omega$  to  $[0, 1]$  satisfying these properties is a probability measure. Connecting that to real-world “randomness” is an **application** of the theory.

(\*) Here's the fine print: these definitions work if  $\Omega$  is finite or countably infinite. If  $\Omega$  is uncountable, then we need to restrict our attention to a  $\sigma$ -field  $\mathcal{F}$  of  $P$ -measurable subsets of  $\Omega$ . For full information, you can take Math 563.

(\*\*) Here's more fine print: I'm taking my random variables  $X$  to be the identity function on outcomes  $\omega$ .

## Independence of events

Take a pair of fair coins. Let  $\Omega = \{HH, HT, TH, TT\}$ . What's the probability that the first or second coin lands heads-up? What do you think  $P(HH)$  ought to be?

	H	T	
H	1/4	1/4	$A = 1\text{st is heads}$
T	1/4	1/4	$B = 2\text{nd is heads}$

Now suppose the coins are welded together — you can only get two heads, or two tails: now,  $P(HH) = \frac{1}{2} \neq \frac{1}{2} \cdot \frac{1}{2} = P(H*) \cdot P(*H)$ .

	H	T	
H	1/2	0	$A = 1\text{st is heads}$
T	0	1/2	$B = 2\text{nd is heads}$

We say that events  $A$  and  $B$  are **independent** if  $P(A, B) = P(A)P(B)$ .

## PMFs and conditional probability

A list of all outcomes  $X$  and their respective probabilities is a **probability mass function** or **PMF**. This is the function  $P(X = x)$  for each possible outcome  $x$ .

1/6	1/6	1/6	1/6	1/6	1/6
-----	-----	-----	-----	-----	-----

Now let  $\Omega$  be the people in a room such as this one. If 9 of 20 are female, and if 3 of those 9 are also left-handed, what's the probability that a randomly-selected female is left-handed? We need to scale the fraction of left-handed females by the fraction of females, to get  $1/3$ .

	L	R
F	3/20	6/20
M	2/20	9/20

We say

$$P(L | F) = \frac{P(L, F)}{P(F)} \quad \text{from which} \quad P(L, F) = P(F) P(L | F).$$

This is the **conditional probability** of being left-handed **given** being female.



## Die-tipping and stochastic processes

Repeated die rolls are independent. But suppose instead that you first roll the die, then **tip it** one edge at a time. Pips on opposite faces sum to 7, so if you roll a 1, then you have a  $1/4$  probability of tipping to 2, 3, 4, or 5 and zero probability of tipping to 1 or 6.

A **stochastic process** is a sequence  $X_t$  of outcomes, indexed (for us) by the integers  $t = 1, 2, 3, \dots$ : For example, the result of a sequence of coin flips, or die rolls, or die tips.

The probability space is  $\Omega \times \Omega \times \dots$  and the probability measure is specified by all of the  $P(X_1 = x_1, X_2 = x_2, \dots)$ . Using the conditional formula we can always split that up into a **sequencing** of outcomes:

$$\begin{aligned} P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) &= P(X_1 = x_1) \\ &\quad \cdot P(X_2 = x_2 \mid X_1 = x_1) \\ &\quad \cdot P(X_3 = x_3 \mid X_1 = x_1, X_2 = x_2) \\ &\quad \cdot P(X_n = x_n \mid X_1 = x_1, \dots, X_{n-1} = x_{n-1}). \end{aligned}$$

Intuition: How likely to start in any given state? Then, given all the history up to then, how likely to move to the next state?

## Markov matrices

A **Markov process** (or **Markov chain** if the state space  $\Omega$  is finite) is one such that the

$$P(X_n = x_n \mid X_1 = x_1, X_2 = x_2, \dots, X_{n-1} = x_{n-1}) = P(X_n = x_n \mid X_{n-1} = x_{n-1}).$$

If probability of moving from one state to another depends only on the previous outcome, and on nothing farther into the past, then the process is Markov. Now we have

$$P(X_1 = x_1, \dots, X_n = x_n) = P(X_1 = x_1) \cdot P(X_2 = x_2 \mid X_1 = x_1) \cdots P(X_n = x_n \mid X_{n-1} = x_{n-1}).$$

We have the **initial distribution** for the first state, then **transition probabilities** for subsequent states.

Die-tipping is a Markov chain: your chances of tipping from 1 to 2, 3, 4, 5 are all  $1/4$ , regardless of *how* the die got to have a 1 on top. We can make a **transition matrix**. The rows index the from-state; the columns index the to-state:

$$\begin{bmatrix} & (1) & (2) & (3) & (4) & (5) & (6) \\ (1) & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 \\ (2) & 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 \\ (3) & 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 \\ (4) & 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 \\ (5) & 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 \\ (6) & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 \end{bmatrix}$$

## Markov matrices, continued

What's special about Markov chains? (1) Mathematically, we have matrices and all the powerful machinery of eigenvalues, invariant subspaces, etc. If it's reasonable to use a Markov model, we would want to. (2) In applications, Markov models are often reasonable<sup>4</sup>.

Each row of a Markov matrix is a conditional PMF:  $P(X_2 = x_j \mid X_1 = x_i)$ .

The key to making linear algebra out of this setup is the following **law of total probability**:

$$\begin{aligned} P(X_2 = x_j) &= \sum_{x_i} P(X_1 = x_i, X_2 = x_j) \\ &= \sum_{x_i} P(X_1 = x_i)P(X_2 = x_j \mid X_1 = x_i). \end{aligned}$$

**PMFs are row vectors.** The PMF of  $X_2$  is the PMF of  $X_1$  times the Markov matrix  $M$ . The PMF of  $X_8$  is the PMF of  $X_1$  times  $M^7$  (assuming the same matrix is applied at each step), and so on.

---

<sup>4</sup>For the current project, a Markov model produces decent *results* for language-specific Jabberwocky words, while [I claim] bearing only slight resemblance to only one of the ways in which people actually form new words.

The concrete how: back to words!

## Phase 1 of 2: read the dictionary file

Word lists (about a hundred thousand words each) were found on the Internet: English, French, Spanish, German. The state space is  $\Omega \times \Omega \times \dots$  where  $\Omega$  is all the letters found in the dictionary file:  $a$ - $z$ , perhaps  $\hat{o}$ ,  $\beta$ , etc.

After experimenting briefly with different setups, I settled on a probability model which is **hierarchical** in word length:

- I have  $P(\text{word length} = \ell)$ .
- Letter 1:  $P(X_1 = x_1 \mid \ell)$ . Then  $P(X_k = x_k \mid X_{k-1} = x_{k-1}, \ell)$  for  $k = 2, \dots, \ell$ .
- I use separate Markov matrices ("non-homogeneous Markov chains") for each word length and each letter position for that word length. This is a lot of data! But it makes sure we don't end words with *gr*, etc.

PMFs are easy to populate. Example: dictionary is *apple, bat, bet, cat, cog, dog*.

Histogram of word lengths:

$$\begin{bmatrix} 0 & 0 & 5 & 0 & 1 \\ (\ell = 1) & (\ell = 2) & (\ell = 3) & (\ell = 4) & (\ell = 5) \end{bmatrix}$$

Then just normalize by the sum to get a PMF for word lengths:

$$\begin{bmatrix} 0 & 0 & 5/6 & 0 & 1/6 \\ (\ell = 1) & (\ell = 2) & (\ell = 3) & (\ell = 4) & (\ell = 5) \end{bmatrix}$$

## Example

Dictionary is *apple*, *bat*, *bet*, *cat*, *cog*, *dog*. Word-length PMF, as above:

$$\begin{bmatrix} 0 & 0 & 5/6 & 0 & 1/6 \\ (\ell = 1) & (\ell = 2) & (\ell = 3) & (\ell = 4) & (\ell = 5) \end{bmatrix}$$

Letter-1 PMF for three-letter words:

$$\begin{bmatrix} 2/5 & 2/5 & 1/5 \\ (b) & (c) & (d) \end{bmatrix}$$

Letter-1-to-letter-2 transition matrix for three-letter words:

$$\begin{bmatrix} & (a) & (e) & (o) \\ (b) & 1/2 & 1/2 & 0 \\ (c) & 1/2 & 0 & 1/2 \\ (d) & 0 & 0 & 1 \end{bmatrix}$$

Letter-2-to-letter-3 transition matrix for three-letter words:

$$\begin{bmatrix} & (t) & (g) \\ (a) & 1 & 0 \\ (e) & 1 & 0 \\ (o) & 0 & 1 \end{bmatrix}$$

## Phase 2 of 2: generate the words using CDF sampling

How can we sample from a non-uniform probability distribution? Think of the PMF as a dartboard. We throw a uniformly wild dart. Outcomes with bigger  $P$  should take up bigger area on the dartboard.

Theorem: This works. Technically:

- We write a **cumulative distribution function**, or **CDF**. Whereas the PMF is  $f(x) = P(X = x)$ , the CDF is  $F(x) = P(X \leq x)$ . (Put some ordering on the outcomes.)
- Let  $U$  (the dart) be **uniformly distributed** on  $[0, 1]$ .
- Then  $F^{-1}(U)$  (appropriately interpreted) has the distribution we want. (See my September 2007 grad talk *Is 2 a random number?* for full details.)

Example: PMF for letter 1 of three-letter words is

$$\begin{bmatrix} 0.4 & 0.4 & 0.2 \\ (b) & (c) & (d) \end{bmatrix}.$$

CDF for letter 1 of three-letter words is

$$\begin{bmatrix} 0.4 & 0.8 & 1.0 \\ (b) & (c) & (d) \end{bmatrix}.$$

If  $U$  comes out to be 0.6329, then I pick letter 1 to be  $c$ . If  $U$  comes out to be 0.1784, then I pick letter 1 to be  $b$ . Etc. I also make a CDF for each row of each Markov matrix.

To generate a word, given the Markov-chain data obtained from a specified dictionary file:

- Use CDF sampling to pick a word length  $\ell$  from the word-length distribution.
- Use the letter-1 CDF for word length  $\ell$  to pick a first letter.
- Go to that letter's row in the letter-1-to-letter-2 transition matrix for word length  $\ell$ . Sample that CDF to pick letter 2.
- Keep going until the  $\ell$ th letter.
- Print the word out.



## Three-letter memory

The non-Markov part of the story: Using Markov chains, as described here, I got decent words, but not always. Real-word correlations go more than one letter deep.

Example: Using a German dictionary, my program generated the 5-letter word *bller*. This made sense: There are *b l \_ \_ \_* words in German, e.g. *bleib*. There are *\_ l l \_ \_* words in German, e.g. *alles*. But my Markov model only looks at correlations between adjacent letters, and thus it didn't detect that *bll \_ \_* never happens in German.

For revision two of the project, I did all the steps described in the previous slides, but now with the following data:

- I have  $P(\text{word length} = \ell)$  as before.
- For first letters,  $P(X_1 = x_1 \mid \ell)$ .
- For second letters,  $P(X_2 = x_2 \mid X_1 = x_1, \ell)$ .
- For the rest,  $P(X_k = x_k \mid X_{k-2} = x_{k-2}, X_{k-1} = x_{k-1}, \ell)$ .

## Results, and a little (but not too much) head-scratching

## Full output vocabulary with a tiny word list

Dictionary is *bake, balm, bare, cake, calm, care, cart, case, cave*. Here are all possible outputs (all of  $\Omega \times \Omega \times \dots$ ) using two-letter and three-letter memory, respectively. Words appearing in the output but not in the input word list are marked with \*.

$\omega$	$P(\omega)$	$\omega$	$P(\omega)$
<i>bake</i>	0.0740741	<i>bake</i>	0.1111111
<i>balm</i>	0.0740741	<i>balm</i>	0.1111111
<i>bare</i>	0.0740741	<i>bare</i>	0.0740741
<i>bart*</i>	0.0370370	<i>bart*</i>	0.0370370
<i>base*</i>	0.0370370	<i>cake</i>	0.1111111
<i>bave*</i>	0.0370370	<i>calm</i>	0.1111111
<i>cake</i>	0.1481481	<i>care</i>	0.1481481
<i>calm</i>	0.1481481	<i>cart</i>	0.0740741
<i>care</i>	0.1481481	<i>case</i>	0.1111111
<i>cart</i>	0.0740741	<i>cave</i>	0.1111111
<i>case</i>	0.0740741		
<i>cave</i>	0.0740741		

When larger word lists are used,  $\Omega$  is far larger than the input word list: i.e. there are far more *mimsy* and *mome* than *were* and *the*. (More on this below.)

## Results with real word lists

For full-size word lists, I don't try to enumerate all possible outputs — I just generate 100 or so at a time. (From here on out I use the **three-letter-memory** model.)

When I feed word lists from different languages into the same computer program, I get different outputs. Hopefully, you can tell which is which.

*churency kingling supprotophated doconic linictoxly stewalorties murine hawkinesses*

*texueux roseras plaçâtes exhumèrent orileffé cinquetassions laissez regre-nèses  
sauceptant montrenards résaïsmez enjupillâmes ratît fausive*

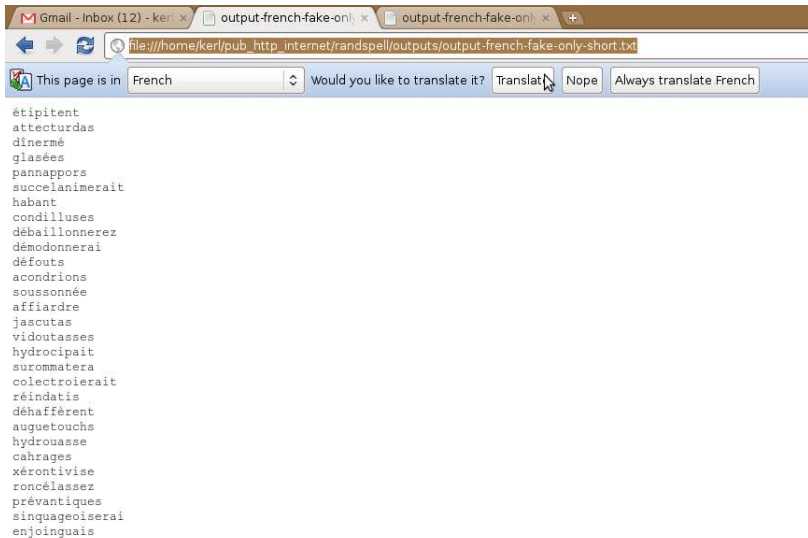
*perónimo bolón sanfija morricete esmotorrar bisfato filamberecer estempolí mícleta  
zarífero senestrosia desalificapio*

*Böservolle techtausfälle Nah wohlassee verschützen Probinus träßcher Postenpland  
einprückt Bußrfere höhegendeter*

*occlamo domitor nestum inhibeo prohisus equino eribro obvolla exteptor exhibro abduco  
loci equa occasco*

For **medium-sized word lists** (a few thousand words), we can enumerate all possible outputs and compute their probabilities ... any guesses as to the top ten?

... a moment for some external validation ...



The screenshot shows a web browser window with three tabs: "Gmail - Inbox (12) - kerl", "output-french-fake-onl", and "output-french-fake-onl". The address bar contains the file path: "file:///home/kerl/pub\_http\_internet/randspell/outputs/output-french-fake-only-short.txt". Below the address bar, a translation prompt is displayed: "This page is in French" with a dropdown menu showing "French". To the right of the dropdown is the question "Would you like to translate it?" followed by three buttons: "Translate", "Nope", and "Always translate French". The main content of the page is a list of 25 nonsense words in French, such as "étipitent", "atteurdas", "dinermé", "glasées", "pannappors", "succelanimerait", "habant", "condilluses", "débaillonerez", "démonnerai", "défouts", "acondrions", "soussonée", "affiardre", "jascutas", "vidoutasses", "hydrocipait", "surommater", "colectroierait", "réindatis", "déhaffèrent", "auguetouchs", "hydrouasse", "cahrages", "xérontivise", "roncélassez", "prévantiques", "sinquageoiserai", and "enjoinguais".

## Top-ten (and bottom-ten) lists

Input corpus size and output  $\#\Omega$ :

	English "GSL-2000"	Deutsch	Français	Español
In	2,284	999	8,410	997
Out	20,920	4,417	2,178,894	5,262

Top and bottom ten:

Word	<i>P</i>	Wort	<i>P</i>	Mot	<i>P</i>	Palabra	<i>P</i>
mill	0.0008756	einen	0.0023929	administration	0.0002272	así	0.0030120
attraction	0.0008295	zum	0.0020100	dire	0.0001982	ya	0.0020080
hold	0.0008209	Vor	0.0020100	chutes	0.0001952	único	0.0020080
baste	0.0007505	vor	0.0020100	peux	0.0001921	tal	0.0020080
tide	0.0007297	nur	0.0020100	jette	0.0001911	por	0.0020080
suppose	0.0007005	ihren	0.0020100	aides	0.0001911	para	0.0020080
sour	0.0006567	alte	0.0020100	sage	0.0001902	ni	0.0020080
come	0.0006567	allem	0.0020100	modes	0.0001698	este	0.0020080
stain	0.0006129	nicht	0.0015075	salle	0.0001661	con	0.0020080
then	0.0005837	neute	0.0015075	plan	0.0001585	ante	0.0020080
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
depertist	8.189e-8	heides	4.54762e-6	goutertiel	2.586e-12	abserén	2.3241e-6
depertism	8.189e-8	heidel	4.54762e-6	coutertiel	2.586e-12	antreír	1.9124e-6
depertian	8.189e-8	beidet	4.54762e-6	boutertiel	2.586e-12	instinta	1.8592e-6
depertial	8.189e-8	beides	4.54762e-6	séertiemiél	2.508e-12	histinta	1.8592e-6
misertist	6.142e-8	beidel	4.54762e-6	géertiemiél	2.508e-12	enserír	1.8592e-6
misertism	6.142e-8	Kintet	3.98819e-6	féertiemiél	2.508e-12	matencil	1.7113e-6
misertian	6.142e-8	Kintem	3.98819e-6	sononarios	1.691e-12	catencil	1.7113e-6
misertial	6.142e-8	Kintel	3.98819e-6	mononarios	1.691e-12	enserén	1.5494e-6
hescestive	4.114e-8	Kintei	3.98819e-6	lononarios	1.691e-12	sontinta	9.960e-7
descestive	4.114e-8	wolitel	3.94127e-6	hononarios	1.691e-12	enstinta	9.296e-7

## Note on the top-ten lists

Reassuring things:

- Probabilities do, in fact, sum to one (bug check!)
- Real words tend to have higher probability.

Troubling things:

- Why are so many  $P$ 's the same?
- Why are the values of  $NP$  (where  $N$  is corpus size) so often integers or simple fractions, such as 2,  $7/4$ ,  $5/6$ ? (E.g.  $P(\text{mill}) = 2/2284$ .)
- If **mill** and **administration** are numbers 1 and 2 on the list, *how is that not a bug???*

The answer to all three questions is the same.

## Fractions which simplify

Let's walk through computation of  $P$  for **mill**, **hold**, and **attraction**.

$P(\_\_\_\_)$	$= 493 / 2284$		$P(\_\_\_\_)$	$= 493 / 2284$	
$P(m\_\_\_   \_\_\_\_)$	$= 27 / 493$		$P(h\_\_\_   \_\_\_\_)$	$= 30 / 493$	
$P(mi\_\_\_   m\_\_\_\_)$	$= 7 / 27$	(mail, mend, mile, milk)	$P(ho\_\_\_   h\_\_\_\_)$	$= 8 / 30$	(half, hope, hunt)
$P(mil\_   mi\_\_\_\_)$	$= 4 / 7$	(mile, mind)	$P(hol\_   ho\_\_\_\_)$	$= 3 / 8$	(holy, home)
$P(\_ill   \_il\_\_\_)$	$= 7 / 14$	(wild, will)	$P(\_old   \_ol\_\_\_)$	$= 5 / 8$	(fold, hole, holy, roll)
$P(\_\_\_\_\_\_\_\_\_\_\_)$	$= 87 / 2284$				
$P(a\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_\_\_\_\_\_\_)$	$= 5 / 87$		+-----+   There are only five 10-letter words starting with 'a'		
$P(at\_\_\_\_\_\_\_\_\_\_\_   a\_\_\_\_\_\_\_\_\_\_\_)$	$= 2 / 5$		in the English GSL-2000 corpus. They are:		
$P(att\_\_\_\_\_\_\_\_\_\_\_   at\_\_\_\_\_\_\_\_\_\_\_)$	$= 2 / 2$		altogether, appearance, artificial, attraction, attractive.		
$P(\_ttr\_\_\_\_\_\_\_\_\_\_\_   \_tt\_\_\_\_\_\_\_\_\_\_\_)$	$= 2 / 2$		+-----+		
$P(\_tra\_\_\_\_\_\_\_\_\_\_\_   \_tr\_\_\_\_\_\_\_\_\_\_\_)$	$= 2 / 2$				
$P(\_\_rac\_\_\_\_\_\_\_\_\_\_\_   \_\_ra\_\_\_\_\_\_\_\_\_\_\_)$	$= 2 / 2$				
$P(\_\_\_act\_\_\_\_\_\_\_\_\_\_\_   \_\_\_ac\_\_\_\_\_\_\_\_\_\_\_)$	$= 2 / 2$				
$P(\_\_\_\_cti\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_ct\_\_\_\_\_\_\_\_\_\_\_)$	$= 9 / 9$				
$P(\_\_\_\_\_\_tio\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_\_ti\_\_\_\_\_\_\_\_\_\_\_)$	$= 18 / 19$	(only exception: attractive)			
$P(\_\_\_\_\_\_\_\_\_\_\_ion   \_\_\_\_\_\_\_\_\_\_\_\_io\_\_\_\_\_\_\_\_\_\_\_)$	$= 26 / 26$				

$$P(\text{mill}) = \frac{2}{2284}; \quad P(\text{hold}) = \frac{15}{8 \cdot 2284}; \quad P(\text{attraction}) = \frac{2 \cdot 18}{19 \cdot 2284}.$$



## War and Peace

An apparent issue with using a word list as input is that words like **the** and **attraction** each appear equally often — namely, **just once** in the word list. Using a **word list with repeats** (e.g. an English translation of *War and Peace*), the top ten output words are more like the top ten input words.

```
$ wc -l wap-input-distinct.txt  
565,620 wap-input-distinct.txt
```

```
$ wc -l wap-input-freq.txt  
17,531 wap-input-freq.txt
```

```
$ wc -l wap-output-distinct.txt  
27,795,523 wap-output-distinct.txt
```

Rank	Input word	Count	Rank	Output word	$P$
1	the	34544	1	the	0.061073
2	and	22225	2	and	0.039293
3	to	16673	3	to	0.029477
4	of	14888	4	of	0.026321
5	a	10548	5	a	0.018648
11	with	5663	11	it	0.009897
12	it	5598	12	with	0.009712
38	have	1975	38	pierre	0.003189
39	pierre	1963	39	were	0.003177
40	prince	1928	40	an	0.002876
127	napoleon	583	127	us	0.000747
220	oh	317	220	napoleon	0.000372
136	am	545	136	whem	0.000647
137	long	544	137	princh	0.000636
236	wife	295	236	firse	0.000344

## Some applications, and conclusion

## An application: language scoring

Aramian Wasielak's idea: run a word (real or not) through the Markov-chain data for all tabulated languages, computing the probability of the word:

$$P(\text{word length} = \ell) \cdot P(X_1 = x_1 | \ell) \cdot P(X_2 = x_2 | X_1 = x_1, \ell) \cdots$$

(last four columns.) Then, for each word, normalize those numbers to get a score between zero and one (first four columns).

Word	En score	Fr score	Sp score	De score	En $P$	Fr $P$	Sp $P$	De $P$
cat	1.000	0.000	0.000	0.000	$5.5 \cdot 10^{-6}$	0	0	0
baguette	0.015	0.985	0.000	0.000	$4.7 \cdot 10^{-9}$	$3.1 \cdot 10^{-7}$	0	0
wurst	0.180	0.000	0.000	0.820	$1.2 \cdot 10^{-7}$	0	0	$5.5 \cdot 10^{-7}$
palapa	0.014	0.056	0.930	0.000	$9.0 \cdot 10^{-9}$	$3.6 \cdot 10^{-8}$	$6.0 \cdot 10^{-7}$	0
fesh	1.000	0.000	0.000	0.000	$9.3 \cdot 10^{-7}$	0	0	0
location	0.719	0.098	0.000	0.181	$1.9 \cdot 10^{-7}$	$2.6 \cdot 10^{-8}$	0	$4.8 \cdot 10^{-8}$
xyzy	0.000	0.000	0.000	0.000	0	0	0	0
brillig	0.000	0.000	0.000	1.000	0	0	0	$2.5 \cdot 10^{-9}$
slithy	1.000	0.000	0.000	0.000	$2.1 \cdot 10^{-7}$	0	0	0
toves	0.000	0.000	0.000	0.000	0	0	0	0
outgrabe	0.000	0.000	0.000	0.000	0	0	0	0
frumieux	0.067	0.895	0.000	0.037	$4.5 \cdot 10^{-11}$	$6.0 \cdot 10^{-10}$	0	$2.5 \cdot 10^{-11}$
griff	0.742	0.139	0.000	0.118	$7.4 \cdot 10^{-7}$	$1.3 \cdot 10^{-7}$	0	$1.1 \cdot 10^{-7}$
vorpal	1.000	0.000	0.000	0.000	$1.3 \cdot 10^{-9}$	0	0	0
muggle	1.000	0.000	0.000	0.000	$1.5 \cdot 10^{-6}$	0	0	0
expecto	0.000	0.000	1.000	0.000	0	0	$8.1 \cdot 10^{-7}$	0
patronum	1.000	0.000	0.000	0.000	$2.0 \cdot 10^{-10}$	0	0	0

## Other possibilities

In this project, my goal was to construct words out of letters, using language-specific empirical knowledge of transition probabilities from **one letter to the next**. One can do something similar, constructing sentences out of (true) words, using language-specific empirical knowledge of transition probabilities from **one word to the next**. Google for **Garkov** and **Rooter**. See also **Cam McLeman's** page on language/math experiments.

Shane Passon's idea: Using more languages (e.g. German, Dutch, Swedish; French, Spanish, Catalan, Italian; Polish, Czech, Russian; etc.) can we adapt the scoring mechanism to measure **relatedness of languages**?

All the machinery here works on letters — specifically on written language. Better results might be obtained by using not letters, but units such as *e*, *n*, *ou*, *gh*. This requires a language expert to decide what the pieces are. Or does it? Can we automate detection of these digraphs, trigraphs, and so on?

While Markov chains are merely phenomenological in this context, they are fully legitimate in the study of how words change over time [Modi].

Lastly, what separates mere portmantarkov arithmetic from art? What makes ***Long time the manxome foe he sought*** so satisfying — and how does *that* work?

Vielen Dank für Ihre Aufmerksamkeit!

Je vous remercie de votre attention!

Gracias por su atención!

Thank you for attending!

## Extra slide(s) in case of questions

### Jabberwocky (Lewis Carroll)

'Twas brillig, and the slithy toves  
Did gyre and gimble in the wabe;  
All mimsy were the borogoves,  
And the mome raths outgrabe.

'Beware the Jabberwock, my son!  
The jaws that bite, the claws that catch!  
Beware the Jubjub bird, and shun  
The frumious Bandersnatch!'

He took his vorpal sword in hand:  
Long time the manxome foe he sought —  
So rested he by the Tumtum tree,  
And stood awhile in thought.

And as in uffish thought he stood,  
The Jabberwock, with eyes of flame,  
Came whiffling through the tulgey wood,  
And burbled as it came!

One, two! One, two! And through and through  
The vorpal blade went snicker-snack!  
He left it dead, and with its head  
He went galumphing back.

'And has thou slain the Jabberwock?  
Come to my arms, my beamish boy!  
O frabjous day! Callooh! Callay!  
He chortled in his joy.

'Twas brillig, and the slithy toves  
Did gyre and gimble in the wabe;  
All mimsy were the borogoves,  
And the mome raths outgrabe.

### Le Jaseroque (Frank L. Warrin)

Il brilgue: les tôves lubricilleux  
Se gyrent en vrillant dans le guave.  
Enmîmés sont les gougebosqueux  
Et le mômerade horsgrave.

«Garde-toi du Jaseroque, mon fils!  
La gueule qui mord, la griffe qui prend!  
Garde-toi de l'oiseau Jube, évite  
Le frumieux Band-à-prend!»

Son glaive vorpal en main il va-  
T-à la recherche du fauve manscant;  
Puis arrivé; à l'arbre Té-Té,  
Il y reste, réfléchissant.

Pendant qu'il pense, tout uffusé,  
Le Jaseroque, à l'œil flambant,  
Vient siblant par le bois tullegeais,  
Et burbule en venant.

Un deux, un deux, par le milieu,  
Le glaive vorpal fait pat-à-pan!  
La bête défaite, avec sa tête,  
Il rentre gallompnant.

«As-tu tué le Jaseroque?  
Viens à mon coeur, fils rayonnant!  
Ô Jour frabjejeais! Calleau! Callai!»  
Il cortule dans sa joie.

Il brilgue: les tôves lubricilleux  
Se gyrent en vrillant dans le guave.  
Enmîmés sont les gougebosqueux  
Et le mômerade horsgrave.

### Der Jammerwoch (Robert Scott)

Es brillig war. Die schlichte Toven  
Wirrten und wimmelten in Waben;  
Und aller-mümsige Burggoven  
Die mohmen Râth' ausgraben.

»Bewahre doch vor Jammerwoch!  
Die Zähne knirschen, Krallen kratzen!  
Bewahr' vor Jubjub-Vogel, vor  
Frumiösen Banderschnätzchen!«

Er griff sein vorpals Schwertchen zu,  
Er suchte lang das manchsam' Ding;  
Dann, stehend unterm Tumtum Baum,  
Er an-zu-denken-fing.

Als stand er tief in Andacht auf,  
Des Jammerwochen's Augen-feuer  
Durch tulgen Wald mit Wiffek kam  
Ein burbelnd Ungeheuer!

Eins, Zwei! Eins, Zwei! Und durch und durch  
Sein vorpals Schwert zerschnifer-schnück,  
Da blieb es todt! Er, Kopf in Hand,  
Geläumfig zog zurück.

»Und schlugst Du ja den Jammerwoch?  
Umarme mich, mein Böhm'sches Kind!  
O Freuden-Tag! O Halloo-Schlag!«  
Er schortelt froh-gesinnt.

Es brillig war. Die schlichte Toven  
Wirrten und wimmelten in Waben;  
Und aller-mümsige Burggoven  
Die mohmen Râth' ausgraben.

## Afternote 1

Matt A., Vikram, and I asked another question about language scoring: what are the most **cosmopolitan** (evenly-scored) words, and the most **language-characteristic** words? For both, I ran all four full word lists through the scoring routine. For the former, I sorted to find least difference between max and min normalized score. For the latter, I found those with highest (i.e. 1.0) scores for each language, but since there were so many 1.0's, I looked at the non-normalized column to break ties.

Cosmopolitan				English	French	Spanish	German				
1.	obliger	21.	porter	1.	rested	1.	moirai	1.	tapadero	1.	gestalten
2.	fade	22.	trine	2.	mucks	2.	bouillons	2.	mosquito	2.	verboten
3.	tramel	23.	mille	3.	tuff	3.	abaisses	3.	habanero	3.	schlieren
4.	vertus	24.	satin	4.	haes	4.	abaissions	4.	desperado	4.	befallen
5.	plane	25.	orache	5.	pike	5.	abaque	5.	caballero	5.	auslander
6.	cause	26.	rapider	6.	hows	6.	abaissent	6.	comprador	6.	schnecke
7.	modeler	27.	genie	7.	buffs	7.	abaisserons	7.	armadillo	7.	schottische
8.	place	28.	have	8.	tuffs	8.	abaques	8.	caudillo	8.	klatzsches
9.	sonder	29.	mimer	9.	dowed	9.	abaissiez	9.	sabadilla	9.	einstains
10.	filmer	30.	lauras	10.	copped	10.	abaissaient	10.	cuadrilla	10.	gesundheit
11.	trace	31.	niche	11.	jibs	11.	abaisse	11.	quebracho	11.	schottisches
12.	ford	32.	lancer	12.	rapped	12.	boudoir	12.	cascarilla	12.	zeitgeber
13.	folia	33.	interne	13.	prow	13.	abaisserais	13.	amontillado	13.	gegenschien
14.	rasper	34.	intender	14.	skin	14.	abaisserait	14.	picadillo	14.	autobahnen
15.	absorber	35.	probe	15.	ripped	15.	abaisserent	15.	pimiento	15.	wunderkind
16.	hetaira	36.	framer	16.	recked	16.	abaissasses	16.	impresario	16.	lebensraum
17.	sente	37.	normal	17.	dogging	17.	abaisserez	17.	enchilada	17.	hausfrauen
18.	robustas	38.	postiche	18.	daws	18.	gridirons	18.	burladero	18.	gesellschaft
19.	rote	39.	vagus	19.	yaws	19.	abaissasse	19.	pistolero	19.	gemeinschaft
20.	rase	40.	intoner	20.	dyes	20.	abaissez	20.	guayabera	20.	kindergarten