

The Portmantout

Dr. Tom Murphy VII Ph.D.*

1 April 2015

1 Introduction

A *portmanteau*, henceforewith non-italicized, is a stringin'-together of two words to make a new word, like “brogrammer” (brother + programmer; a programmer who is your brother), “hupset” (hungry + upset; a bit more passive than hangry), or “webinar” (web + nerd). Portmanteaus were invented by Lewis Carroll, the Jabberwock of wordplay.

It is natural to think of generalizations of the portmanteau, such as the *portmantois*,¹ (itself a portmanteau of portmanteau + trois, French-language for three) the human-centipedification of three words, such as “anachillaxis” (anaphylaxis + chill + relax; a severe allergic reaction to idleness) or “brogrammermaid” (brother + programmer + mermaid; a programmer who is your brother and a mermaid).

In this paper I present the world’s first (?)²³ *portmantout*, a portmanteau of all English-language words (*tout* means “all” in French-language). I also considered calling this a *portmantotal*, *portmantotale*, etc., as well as *portmantoutal* (a portmantois of the first three) or even *portmantoutale*. You kind of see how this can get out of hand. The word is 630,408 letters long and contains all 108,709 words in a particular wordlist called `wordlist.asc`.⁴ Even though nobody can really agree what all the words in English are, the technique used to generate this portmantout should work for most very long word lists, although we will see in Section 3.2 that a handful of words are very important.

Since the word itself is 11 pages long in 4pt type with .75 linespacing, and this SIGBOVIK proceedings is positively overfull hbox with content, we should probably get on with it.

2 Computationalizing “portmanteau”

A real portmanteau is usually phonetic, like “portmantotally” is about the sound of “—teau” being the same as “to—”. It’s also not unusual for part of the word to be completely dropped, as in “chillax”, which drops the “re—” from relax. They are also usually clever or meaningful. For the sake of computing a portmantout, we need to make some rules about what it is, and it can’t require cleverness or semantic/phonemic interpretation of words if I’m going to start and complete this project on the day of the SIGBOVIK deadline.

Generalized portmanteau. For a set of strings L , a string s is a generalized portmanteau if the entire string can be covered by strings in L . A cover is a set of word occurrences $W = \langle m, n \rangle$, where s_m-s_n (the substring starting at offset m and ending at n , inclusive) is in L , and, taken sorted by the m component, $W_{i.n} \geq W_{i+1.m}$ for each i in range. For example,

**temper
red
rewrotempered
rewrote**

This string can be covered by *rewrote*, *temper*, and *red*, so it is a generalized portmanteau if these three strings are in L . (Spoiler alert: L is English-language, so they are.) Importantly, the covering strings overlap: *rewro*, *temper* and *ed* on their own would not cover this string (and *rewro* is not a word). Therefore, we cannot simply concatenate the entire dictionary.

Portmantout. A generalized portmanteau is a portmantout if it contains every string in L as a substring. The words need not be in its cover, as there may be mulple covers (In fact I conspicuously did not choose the simpler one *rewrote* + *tempered*.) Other words, like *wrote*, *rote* and *rot* are in there too “for free”, even though they may not be able to participate in a legal cover.

A word may appear multiple times; we just have to get them all. This is fairly unavoidable—the word *a* appears 60,374 times in the portmantout. Perhaps more surprising is that the word *iraq* appears 315 times.

*Copyright © 2015 the Regents of the Wikiplia Foundation. Appears in SIGBOVIK 2015 with the **etaoin shrldu** of the Association for Computational Heresy; *IEEEEE!* press, Verlag-Verlag volume no. 0x40-2A. BTC 0.00

¹Graham Smith, personal communication.

²I did a couple Google searches; seems good enough.

³Enjoy source code: <http://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/portmantout>

⁴Tom Murphy VII, “What words ought to exist?”, SIGBOVIK 2011

Note that a “generalized” portmanteau does not actually include most colloquial portmanteaus; **rogrammer** cannot be covered since we dropped the “p—” in **rogrammer**. **rogrammer** is also not a generalized portmanteau since **bro** and **grammar** do not overlap, yo, but I think it would be accepted colloquially by most dudes. Disrupt!

3 Generating a portmantout

It’s fairly straightforward handwaving to see that generating the shortest portmantout is NP-complete. Seeing that it is in NP is easy; we just need to check the cover and look up all the substrings, which is clearly polynomial. It is probably NP-hard because the traveling salesman problem can be reduced to it; for each node in the graph, generate a two-symbol word xy where x and y are fresh symbols; for each edge between cities $x_i y_i$ and $x_j y_j$ generate a string $y_i u^w x_j$, where u is also a fresh symbol repeated w times, the cost of the edge. This allows us only to join two city words by using a corresponding edge word.⁵

OK but good news! Since it’s NP-complete, we know that we can come up with a solution that’s both non-optimal and slow, and we can still feel pretty good about it. We proceed in two steps: Generating particles eagerly, and then joining them together.

3.1 Generating particles

For the first step, we load up all the words, and insert each word into a multimap, keyed by each of its non-empty prefixes. We then start by initializing a particle from any word; we choose **portmanteau** to start, obviously. Then, repeatedly:

- Check each suffix of the particle in descending length-order,
 - If we have a word that has not already been used, strip the suffix from its start and append the remainder to the particle
- Otherwise, emit the particle and start a new one with any unused word.

As an additional optimization, we discard words that are substrings of any particle. This search makes the program much slower, but it produces a much more efficient portmantout.

This always makes progress, using up one word at each step: We either append it to our current particle, or we start a new particle with a word. The particles are all generalized portmanteaus by construction, because each added word has non-empty overlap with the previous one. Here’s an example particle: **overmagnifyinggearlessshrimpierabbinicalcicadaeratorshrimpiestandardizableleaseholdershrimpingementshrimpsychedelically** (overmagnifying + gearless + shrimpier + rabbinical + calc + cicada + aerators + shrimpiest + standardizable + leaseholders + shrimping + impingements + shrimps + psychedelically; presumably having something to do with shrimp).

At this point, we have about 38,000 particles, many of which are a single word. English contains many imbalances, like vastly more words ending with “—y” (10,071) than beginning “y—” (only 338), so it is not hard to see how we may get stuck with no new words to add to a particle. We’ve also used each word only once, and locally maximized the amount of overlap. If we can join these particles together in a valid way, we’ll have a portmantout.

3.2 Joining particles

Since we’ve already used every word, and, by construction, these particles cannot be adjoined directly, we know we will need to reuse some words to join them together. A simple way to do this is to construct a table of size 26^2 that for every pair of letters a and b , contains a short word that starts with a and ends with b . 86% of the table entries can be filled in, but some letters are very tricky: For example, almost no words end with “q” (we have only **colloq** and **iraq** in our dictionary), and there are no words that start with “v” and end with “f”. Fortunately, if we consider all two-word (generalized) portmanteaus, using basically the same algorithm as in Section 3.1, we can fill the table completely (Figure 1).

It is lucky for the existence of words like **iraq**; they are used for many of the entries in the “q” column. In fact, without a handful of such words, it might be the case that English would not permit a portmantotal! There are probably some less irregular languages that cannot achieve this lexical feat. :’-(

This table alone would allow us a very simple algorithm for generating a valid portmantotal: Just take words from the dictionary and concatenate them, but when putting e.g. **tv** and **farm** together, we use the v-f linker **vetof** (**veto** + **of**) from the table. We can’t ever fail! However, this would produce a portmantout that’s bigger than the dictionary itself, which isn’t very economical. Rather than use the whole dictionary this way, we instead join all of the 38,000 particles from the previous section. These are much more compact. And now we are done!

4 The portmantout

This portmantout is 630,408 letters long; there are 931,823 total letters in the dictionary so this is a compression ratio of 1.47:1. In comparison, “**rogrammermaid**” (although an illegal generalized portmanteau) has a compression ratio of $2^4/14 = 1.71:1$. So it is fair to say that we are in the ballpark of a “solid portmanteau.” Of course, the gold standard is a compression ratio of $\infty : 1$ —for the case that we have the word **portmanteau**, a totally overlapping portmanteau of **portmanteau** + **portmanteau**,⁶ iterated infinitely.

⁵There are some rubs: TSP requires that nodes only be visited once but a portmantout can use words multiple times. I believe that the multi-visit generalization of TSP is also NP-hard. The portmantout solution also requires visiting every edge, but we can relax this by concatenating all edge words e into a new mega-long word like $e_0 z e_1 z \dots e_k$ where z is also a fresh symbol; since this word must appear and all edges are substrings of it, we now have no requirement that the rest of the solution (containing our TSP embedding) contains all edge words. This kind of trick also lets us set the start node for TSP.

⁶Cara Gillotti, personal communication, 2015.

—	a	b	c	d	e	f	g	h	i	j	k	l	m
a	anna	arab	arc	add	ace	agof	aleg	ash	ansi	arconj	ark	ail	am
b	boa	bib	barc	bad	be	barf	bag	bach	bassi	baconj	back	bail	bum
c	cia	cab	calc	cod	cue	calf	cog	cash	cacti	conj	calk	call	cum
d	dada	dab	doc	dad	die	dof	dig	dash	deli	deconj	dank	deal	dam
e	era	ebb	etc	end	eve	elf	egg	each	elhi	econj	elk	eel	elm
f	feta	fib	farc	fad	fee	fief	fag	fish	fbi	falconj	fink	fail	farm
g	gaga	gab	getc	god	gage	gof	gag	gash	genii	garconj	gawk	gal	gem
h	hula	hub	hetc	had	he	half	hag	hash	haji	hadj	hack	hail	ham
i	idea	iamb	isac	ibid	ice	if	ifag	inch	ifbi	iconj	ilk	ill	ibm
j	java	jab	jarc	jaded	joe	jiff	jig	josh	jinni	jehadj	jack	jail	jam
k	kaka	kerb	kepic	kid	kale	kerf	keg	kith	kepi	kashadj	kick	keel	kakam
l	lava	lab	letc	lad	lie	leaf	lag	lash	levi	loconj	lack	loll	loam
m	mama	mob	mac	mad	me	miff	meg	mach	magi	maconj	mack	mail	mom
n	nasa	nab	narc	nod	name	nof	nag	nigh	nazi	narconj	nark	nil	nam
o	ova	orb	orc	odd	ode	of	orig	ooh	ofbi	oohadj	oak	oil	ovum
p	pea	pub	proc	pad	pee	pelf	peg	path	padri	poconj	pack	pal	palm
q	qiana	qaidab	quebec	qaid	quake	quaff	quahog	qoph	quasi	qophadj	quack	quail	quam
r	raga	rib	rabic	rad	rue	ref	rag	rash	rani	reconj	rack	rail	ram
s	sea	sob	sac	sad	see	sof	sag	sash	ski	shadj	sack	sail	sam
t	tea	tub	talc	tad	tee	tof	tag	tach	taxi	taconj	tack	tail	tim
u	usa	upub	uric	used	use	ufof	ufog	ugh	ugli	ughadj	umiak	ural	unum
v	via	verb	vetc	veld	vade	vetof	vying	vetch	verdi	vaticonj	vailk	vail	viam
w	whoa	web	warc	wad	we	waif	wag	wash	wadi	washadj	wok	wail	warm
x	xenia	xmasob	xebec	xyloid	xylene	xmaself	xmasag	xmash	xmaski	xebeconj	xmask	xylitol	xylem
y	ymca	yamob	yetc	yard	yale	yaref	yang	yeah	yeti	yeahadj	yak	yawl	yam
z	zeta	zagab	zinc	zend	zone	zoof	zag	zooH	zuni	zinconj	zooak	zeal	zoom
—	n	o	p	q	r	s	t	u	v	w	x	y	z
a	an	ago	amp	airaq	air	as	at	adieu	atv	anew	apex	any	abuzz
b	ban	bio	bop	bassiraq	bar	bus	bat	beau	batv	bow	box	by	buzz
c	can	ciao	cap	colloq	car	cabs	cat	chou	catv	cow	calx	coy	chez
d	dan	do	dip	deairaq	dor	dis	dot	dayou	dotv	dew	deux	day	doyez
e	eon	ego	emup	emiraq	ear	ears	eat	emu	eatv	eraw	eaux	easy	elfez
f	fan	faro	fop	firaq	far	fads	fat	flu	fatv	few	fax	fey	fez
g	gin	go	gap	geniiraq	gor	gas	get	gnu	getv	glow	galax	gay	grosz
h	hen	halo	hip	hairaq	her	has	hat	hemu	hatv	how	hex	hay	hertz
i	in	ino	imp	iraq	intr	is	it	iflu	itv	ifew	ibex	icy	ifez
j	jean	jato	jeep	jinniraq	jar	jabs	jet	juju	jetv	jaw	jeux	jay	jazz
k	ken	kayo	keep	kafiraq	kafir	kays	kit	kudu	kiev	knew	knox	key	klutz
l	lain	leo	lap	liraq	lair	labs	let	lieu	letv	law	lax	lay	leviz
m	man	moo	map	magiraq	moor	macs	mat	menu	mirv	mow	max	my	machez
n	non	no	nap	noiraq	nor	nabs	net	nehru	netv	new	nix	nay	nertz
o	on	ono	ofop	obeliraq	or	oafs	oat	oflu	oatv	odew	onyx	obey	oyez
p	pan	paso	pep	pairaq	par	pus	pat	peru	patv	paw	pox	pay	phiz
q	quean	quito	quip	quasiraq	qatar	ques	quit	quipu	quitv	qaidew	qophex	quaky	quiz
r	ran	redo	rap	raniraq	rear	rads	rat	ragnu	ratv	raw	roux	ray	razz
s	sin	so	sap	siraq	sir	sos	sat	situ	shiv	saw	sax	say	sitz
t	tan	to	tap	tapiraq	tar	tis	tit	tabu	tv	tow	tax	thy	tviz
u	urn	ufo	up	ugliraq	user	us	unit	uperu	univ	upaw	unix	ugly	uphiz
v	van	veto	vamp	viziraq	veer	vans	vat	virtu	vatv	vow	vex	vary	viz
w	win	who	warp	weiraq	war	was	wet	wemu	wetv	wow	wax	way	whiz
x	xenon	xmaso	xmasp	xebecolloq	xyster	xmas	xmast	xylemu	xmashiv	xmasaw	xerox	xenicy	xmasitz
y	yen	yeno	yep	yetiraq	year	yes	yet	you	yetv	yaw	yalex	yamy	yawhiz
z	zen	zoo	zap	zuniraq	zoor	zags	zest	zebu	zestv	zapaw	zapox	zany	zaphiz

Figure 1: Minimal joining strings for every letter (rows) to every other letter (columns).

