Mark Rosenfelder

How likely are chance resemblances between languages?

The first rule is, you must not fool yourself. And you are the easiest person to fool.
--Richard Feynman

Contents

On sci.lang we are often presented with lists of resemblances between far-flung languages (e.g. Basque and Ainu, Welsh and Mandan, Hebrew and Quechua, Hebrew and every other language, Basque and every other language), along with the claim that such resemblances "couldn't be due to chance", or are "too many" to be due to chance.

Linguists dismiss these lists, for several reasons. Often a good deal of work has gone into them, but little linguistic knowledge. Borrowings and native compounding are not taken into account; the semantic equivalences proffered are quirky; and there is no attempt to find systematic sound correspondences. And linguists know that chance correspondences do happen.

All this is patiently explained, but it doesn't always convince those with no linguistic training-- especially the last point. Human beings have been designed by evolution to be good pattern matchers, and to trust the patterns they find; as a corollary their intuition about probability is abysmal. Lotteries and Las Vegas wouldn't function if it weren't so.

So, even one resemblance (one of my favorites was ***gaijin*** vs. ***goyim***) may be taken as portentous. More reasonably, we may feel that one resemblance may be due to chance; but some compilers have amassed dozens of resemblances. Such lists may be criticized on other grounds, but even linguists may not know if the chance argument applies. ***Could*** a few dozen resemblances be due to chance? If not, what is the approximate cutoff?

The same question comes up in evaluating the results of Greenbergian mass comparisons; or proposals relating language families (e.g. Japanese and Tibeto-Burman) based on very small numbers of cognates. Again, it would be useful to know how many chance resemblances to expect.

I will propose a simple but linguistically informed statistical model for estimating the probability of such resemblances, and show how to adjust it to match the specific proposal being evaluated.

A trivial model: Abstract languages sharing a phonology

Let's start with a simplified case (we'll complicate it later). We will compare two unrelated languages A and B, each of which has 1,000 lexemes of the form CVC, and an identical semantics and phonology. That is, if there is a lexeme ***a*** in A with some meaning M, there will be a lexeme ***bp*** in B ***phonetically*** identical to ***a***, and a lexeme ***bs*** with the same ***meaning*** as ***a***.

What is the probability that ***bp*** is ***bs***?-- that is, that there is a chance resemblance with ***a***? It can be read off from the phonology of the typical root. Supposing there are 14 consonants and 5 vowels, it is 1/14 * 1/5 * 1/14, or 1 in 980. (This assumes that the vowels and consonants are equiprobable, which of course they are not.) For ease of calculation we'll round this to 1 in 1000.

How many chance resemblances are there? As a first approximation we might note that with a thousand chances at 1 in 1000 odds, there's a good chance of getting one match.

However, this is not enough. How likely is it exactly that we get one match? What's the chance of two matches? Would three be quite surprising? Fortunately probability theory has solved this problem for us; the chance that you'll find ***r*** matches in ***n*** words where the probability of a single match is ***p*** is the binomial probability

$$(n! / (r! (n-r)!)) p^r (1 - p)^{(n-r)}$$

or in this case

$$(1000! / (r! (1000-r)!)) .001^r .999^{(1000-r)}$$

For the first few ***r***:

p(1) = .368
p(2) = .184
p(3) = .0613
p(4) = .0153
p(5) = .00305
p(6) = .000506

So the probability of between 1 and 6 matches is .368 + .184 + .0613 + .0153 + .00305 + .000506 = .632, or about 63%. It would be improbable, in other words, if we found no exact matches in the entire dictionary. (But not ***very*** improbable; p(0), which we can find by subtracting the above ***p***'s from 1.0, is 37%.)

Semantic and phonetic leeway

Proffered resemblances are rarely exact. There is always some **phonetic and semantic leeway**. Either can be seen as increasing the set of words in B we would consider as a match to a given word ***a*** in A.

For instance, suppose for each consonant we would accept a match with 3 related consonants, and for each vowel, 3 related vowels. Since we're assuming a CVC root structure, this gives 3*3*3 = 27 words in B which might match any given ***a***.

And suppose for each word ***a*** we will accept 10 possible meanings for ***b***. This applies to each of the 27 phonetic matches; so ***a*** can now match a pool of 27*10 = 270 lexemes. The probability that it does so is of course 270 in 1000, or .27. Every lexeme in A, in other words, has a better than 1 in 4 chance of having a random match in B!

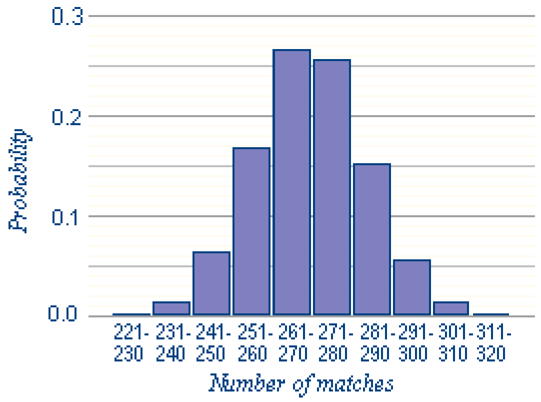
How many chance resemblances are there now? The same formula can be used, with the revised estimate for ***p***:

$$(1000! / (r! (1000-r!))) .27^r .73^{(1000-r)}.$$

There is a significant probability for very high numbers of matches, so we must continue calculating for ***r*** well into the hundreds. The results can be summarized as follows:

p(up to 210) - negligible
p(211 to 220) = .0002
p(221 to 230) = .0020
p(231 to 240) = .0148
p(241 to 250) = .0647

p(251 to 260)	= .1686
p(261 to 270)	= .2661
p(271 to 280)	= .2571
p(281 to 290)	= .1536
p(291 to 300)	= .0573
p(301 to 310)	= .0134
p(311 to 320)	= .0020
p(over 320)	- negligible



This looks a lot like a normal distribution, and in fact it is one, if ***np*** and ***n(1-p)*** are both over 5. (For typical lexicon sizes, the distribution will be normal if p > .01.) The 'expected case'-- the number of matches with the highest probability-- will be ***np***; in the above case, 270, with a probability of .0284.

I will suggest refinements to this model below, but the basic features are in place: a probability for a single match; a calculation for number of expected matches; and an adjustment for phonetic and semantic leeway.

Vocabulary size and other variables

What happens as various parameters of the model are changed?

In the model above, **vocabulary size** is not independent of the sample phonology. If syllables are all CVC, with 5 vowels and 14 consonants, then there are only 980 phonologically possible words.

If we use a larger vocabulary size in the formula, we are conceptually re-using some of these possible words. This might not be a bad model for a language with many homonyms, or if we are simply ignoring some of the phonology of the language (e.g. tones, or final vowels).

We can also try different **phonologies**. For instance, here are a the numbers of expected matches for a few different word types, again allowing 3 matches per sound, and 10 semantic matches per word:

Form	#C	#V	***Lexicon size***	***Phon matches / word***	***Expected matches***
CV	14	5	70	9	90
CVC	14	5	980	27	270
CVCV	14	5	4900	81	810

The number of matches isn't increasing as fast as the vocabulary size. For instance, as we go from CVC to CVCV, we increase the lexicon fivefold (for each CVC syllable, there are 5 CVCV syllables, one with each vowel), while the number of matches only increases threefold (the extra vowel can be matched only 3 ways).

(For the first (CV) case, the expected number of matches is larger than the lexicon size! This is because, with the rules as stated, we're often going to find more than one match per word. We'll come back to this later.)

What about **exact matches**? Curiously, the number of exact matches averages about 1 for any phonology, and therefore for widely differing vocabulary sizes. The reason is not hard to find: as the lexicon size increases, the chance of a given match goes down, but we get more chances.

(The average of 1 exact match needs to be qualified, because of homonymy: a word can have several meanings, and thus match several possible words. In effect, languages come with a built-in semantic leeway; a handful of exact matches is therefore no surprise.)

What about different **phoneme inventories**? What if we have 20 consonants and 10 vowels? There are now 4000 possible CVC syllables; yet without varying any other parameters, we would still expect only 270 matches.

But the number of phonetic matches is not really independent of the phonology we use. In a five-vowel system, a phonetic leeway of 3 per sound matches means (e.g.) accepting ***pat*** and ***pit*** as matches for ***pet***. In a ten-vowel system, might we not also accept ***pEt*** (open e) and ***pöt*** (rounded e) as well?

Contrariwise, suppose there are only 3 vowels, as in Quechua or Classical Arabic. A 3-vowel leeway means that all vowels match all vowels!

Here's some 20C/10V languages with a few different phonetic leeways (and the same semantic leeway of 10):

Form	#C	#V	***Lexicon size***	***Phonetic leeway***	***Expected matches***
CV	20	10	200	2	40
CV	20	10	200	3	90
CV	20	10	200	5	250
CVC	20	10	4000	2	80
CVC	20	10	4000	3	270
CVC	20	10	4000	5	1250
CVCV	20	10	40000	2	160
CVCV	20	10	40000	3	810
CVCV	20	10	40000	5	6250

The reader may well conclude that with the appropriate choice of parameters, any number of matches is possible! That would be more or less accurate; but note:

- The parameters are not simply selected according to taste; they must be taken from the particular comparison we are evaluating. For instance, if the comparer matches ***t*** with ***th*** and ***ch*** and ***d***, the phonetic leeway is ***at least*** 4.

- If there's ***any*** phonetic and semantic leeway at all-- and I've never seen a relationship proposal with none-- then one can well expect dozens or even hundreds of matches, rather than the bare handful intuition might suggest.

A computer model

I've written a computer program which generates two random languages and compares them for matches. Phonology, phoneme inventory, matching algorithm, phonetic leeway, and semantic leeway can all be varied.

You can get the source code for the program here.

The program gives several counts, corresponding to slightly differing **match-counting methodologies**. For instance, here's the output of a sample run simulating one of the models discussed above: 980 possible words with phonetic leeway of 3 and semantic leeway of 10.

```
Matches in 980 words:
```

For each word in A report:	Allow a word in B to match:	
	multiple words in A	only globallyonce
all matches	290	257
1st match	256	230

The program generates two random languages A and B. Then, for each word in language A, the program tests each word in language B. There is obviously a chance that this will result in more than one match for each word in A. The total number of matches is reported in the top left; the value here (290) is close to our expectation based on ***np*** (980 * .275 = 270). In statistical terms, this is a binomial probability distribution with replacement.

The bottom row counts only one match for each word in A. So, even if we found three matches for a given word in A, we count only one. So in the example, though there were 290 total matches, there were 256 words in A that had at least one match, and some had more than one.

The right-hand column counts matches if we allow each word in B to be used only once. That is, if any word B is ever counted as a match for a word in A, it's withdrawn from the pool, and no further word in A can match it.

It might seem obvious that we only want one match per word; but again, we must follow the practice of the comparison we're evaluating. A comparison of Basque and Ainu posted to sci.lang, for instance, matched Ainu ***poro monpeh*** 'big finger (=thumb)' to ***erpuru***, and ***poro aynu*** 'big man (=adult)' to ***porrokatu*** 'tired'. Yes, the semantic leeway there is formidable; but the point is that one Ainu word, ***poro***, is being used to match two entirely different Basque words.

It's not hard to find the expected **number of words in A with one or more matches** (the number in the lower left of the program output). Let's start by subtracting out the cases where we found two matches in B. How many of these will there be?

The probability that word A_i has the same meaning as B_i is of course 1 in ***n*** (the lexicon size). If we allow phonetic leeway, we get additional chances for a match; each of these phonetic near-matches is multiplied further by the semantic leeway. In the example above we effectively get 270 chances for each word in A.

This is really another binomial probability problem: we have 270 trials, each with probability 1 in 980, and we want to know the chances for 2 successes. Using the formula given, the chance is (270! / (2! (270-2)!)) $(1/980)^r$ $(979/980)^{(270-r)}$ = .0288. Now since there are 980 words in A, we would expect about .0288 * 980 = 28 two-word matches. Since we expected 270 total matches, this leaves about 242 words with one or more matches. This is not far from the 256 found in the sample run.

The same formula will tell us how often a word in A will match ***three*** words in B-- the probability is .00262, leading us to expect 2.57 matches. Since these would count three times in the total count and we want them to count only once, we have to subtract another 2 * 2.57 = 5 from that 242, giving us a more accurate estimate of 237. A four-word match has a probability of .00018, which doesn't affect the final number any. In practice it's not really worth correcting for anything beyond two-word matches.)

(Some readers may worry that the numbers from the sample run (290, 256) are 'too far' from the expected values (270, 237). But no, these numbers are well within the variance. If I run the program a hundred times, I get an average of 270.6 total matches, 240.5 words matched.)

Better phonological models

We'd like to remove the unrealistic assumptions in this model, starting with the absurdly simplified phonologies. Fortunately this is not hard to do; it amounts to finding a better ***p***.

The model above assumes that both languages have the same phonology, which obviously isn't usually the case. If you're evaluating a proposed set of resemblances, it's usually fairly evident what the compiler counted as a match. For instance, the Quechua/Semitic comparison discussed below normally just (loosely) matches initial and medial consonants. To estimate ***p***, then, we multiply the probability of a match on initial consonants with that of a match on medial consonants.

To do this, we need to know how many possible consonants there are, and how many of them are considered a match. The latter of course must be taken from the proposed resemblances; if there's enough of them the number of consonants can be too, otherwise we need to find out the language's phonology.

Phonemes don't actually appear with equal frequencies, and this can be important. For instance, Quechua has just three vowels, a/i/u (though allophonic o/e are found in some dictionaries). The chance that the main vowel of a root is an **a**, however, is not 33.3% but 56%. A match with medial **a** is therefore correspondingly less surprising. In a sidebar, I've estimated the chances of random matches between Quechua and Chinese taking **phoneme frequencies** into account.

What if we're only given **a word or two**-- say, that cute Japanese ***gaijin*** / Hebrew ***goyim*** pair? Even with one word, we can estimate phonetic leeway segment by segment:

- first consonant matches; Japanese has about 14 consonant phonemes, so let's say the chance is 1 in 14
- first vowel is basically ignored
- medial consonant is extremely loose-- a dental affricate versus a palatal approximant. To be as fair as possible, let's say the chance is 1 in 3 that we get even this close.
- second vowel matches; Japanese has 5 vowel phonemes, so the chance here is 1 in 5
- final consonant is nasal in both cases; but Japanese words can ***only*** end in a vowel or **n**-- so let's assign a chance of 1 in 6.

That gives us (1/14) * (1/3) * (1/5) * (1/6) or 1 in 1260 (= .00079). This pair doesn't of course come with the size of the comparer's dictionary; but from the probability alone we can say if the lexicon has at least 1260 words, we're likely to find at least one Japanese/Hebrew match that's this close. In a dictionary of 10,000 entries we could expect about 8 such matches.

Of course it's worse than that, since there's semantic leeway as well. ***Goyim*** means 'nations' (from ***goy*** 'nation'); though it's metaphorically used for 'non-Jews', it's certainly not an exact semantic match for a word meaning 'foreigner'. (***Gaijin*** is a borrowing from Chinese ***wairén*** 'outside-person'.) I can't emphasize enough that inexact matches greatly multiply the chances for finding random resemblances.

Semantic matches

Unfortunately there's no rigorous way to improve our estimate of semantic leeway-- the number of meanings that will be considered to match a given word. Classifying meanings has been a parlor pastime for several centuries, but this has never produced any generally useful way of analyzing meanings, much less determining whether two meanings are close or not.

About all that can be done is to emphasize that most actual attempts to find "cognates" accept very considerable semantic leeway, and that this greatly increases the chance of random matches.

It's helpful to see just what a semantic leeway of 10 or 100 meanings looks like. For instance, let's take the word "**eat**".

If we allow 10 matches, we'd accept something like:

eat, dine, bite, chew, swallow, feed, hungry, edible, meal, meat

If we allow 25 matches, we'd accept:

eat, dine, bite, nibble, chew, munch
consume, devour, taste, drink, gulp, swallow,
meal, dinner, supper, snack, meat, food
hungry, thirsty, fast, edible, tasty, mouth,
corrode

If we allow 100 matches, we'd accept:

eat, dine, sup, stuff, gorge, nibble, ruminate, take in, tuck in, gobble, bolt, swill
chew, munch, slurp, bite, gnaw, masticate, -vorous
consume, devour, ingest, nourish, regale, taste, partake
drink, guzzle, suck, gulp, quaff, swallow, imbibe
morsel, mouthful, serving, helping, entree
food, victuals, fare, fodder, provender, diet
meal, repast, dinner, lunch, luncheon, supper, breakfast, snack, feast
meat, soup, fowl, vegetables, bread, trough, plate, hearth,
eat (of animals), feed, provide, forage, grub, chow
cuisine, board, table, mess, restaurant, cook
hungry, thirsty, peckish, famine, glutton, fast, full
edible, comestible, potable, tasty, delicious
mouth, cheek, teeth, tongue, stomach
deplete, waste, destroy, ravage, corrode, etch, erode, subsist (on), live, prey

Ten matches may seem like a lot; but nothing in even the list of 100 matches is a real stretch. The real question is: looking for matches for 'to eat' in A, **would our language comparer take a word meaning 'nourish' or 'delicious' or 'gnaw' as a match**--indicating that his semantic leeway is more like 100 than like 10? The temptation is almost irresistible.

It's worth pointing out that semantic leeway is really ***multidimensional***. The 100-word list shows several directions we can explore starting from the word 'eat': types of eating; other types of ingestion; types of food; types of feeding or food preparation; qualities or states associated with eating; associated parts of the body; differences in connotation or register; differences in the type of eater; metaphorical extensions. You may accept only a little variation along each dimension, but because there are many dimensions the total variation is high.

(This may suggest a rough method for estimating semantic leeway: count the dimensions in which the words differ. For instance, Greenberg & Ruhlen's ***mekeli*** ('nape', from Faai) and ***amu'ul*** ('to suck', from Mixe) seems to follow the semantic links ***nape / (nearby body part) neck or throat / (associated verb) swallow / (another bodily verb) suck***; if there are as few as five alternatives at each of these four steps, the semantic leeway is at least 5 x 5 x 5 = 125. (The reader may enjoy tracing how many steps are required to reach their proposed Indo-European cognate, 'milk'.)

Effect of semantic leeway

What happens to the number of expected matches as semantic leeway increases?

We are traipsing through language A, word by word. For each word ***a***, the probability ***p*** of finding a match in B is, say, .01. But now suppose the semantic leeway ***m*** is 10. So for word ***a***, we have to check for a match 10 times.

What does this mean numerically? It depends on how we count matches.

- We want to know if ***a*** matches **any word**. We check for matches for ***a*** 10 times. What are the chances that we'll find a match?

  The easiest way to compute this is backwards: find the probability that we ***won't*** find a match, which is simply $(1 - \boldsymbol{p})^{\boldsymbol{m}}$. To see this, think of checking the 3rd word. To not find a match on words

1, 2, 3, we must find no match on word b_1 (probability .99), no match on word b_2 (prob .99), and no match on word b_3 (prob .99). Since these events are independent, the cumulative probability of .99 * .99 * .99--that is, $(1 - p)^m$.

So the probability of finding a match is $1 - ((1 - p)^m)$. In our example, this is $1 - .99^{10} = .095618$.

- We want to know the **number of words *a* matches**. That is, we might find more than one phonetic match in our set of ***m*** semantic matches. The average number of matches is easy, since it's ***p*** for each word, and thus ***mp*** total. We could also go on to find the probability for one match, two matches, etc.

Below, I'll pursue the first of these alternatives, on the grounds that it most closely resembles the methodology of the comparisons we want to evaluate. As we've seen with the Basque/Ainu comparison, comparers generally don't scruple at using the same word twice. My impression, however, is that this only applies to one of the languages.

In other words, their method is, I take it, to loop through the words of language A and for each one find, if possible, one match in B. This methodology avoids multiple matches per word in A, but doesn't prevent us from using the same word in B more than once. The formulas suggested here model this methodology.

Vocabulary size

Semantic leeway is not really independent of vocabulary size.

As the vocabulary grows into the thousands, we can expect finer and finer distinctions of meaning; but the language comparer is likely to ignore them. A language of 10,000 words may well distinguish 'eat (of humans)' vs. 'eat (of animals)' vs. 'devour' vs. 'bite' and so on. Yet the language comparer, looking for a match for 'eat (of humans)', is not going to skip over 'eat (of animals)'.

If the comparer is working only with (say) a 200-item wordlist, not much semantic variation will be available. Quite a bit is available in a 2000 or 20,000-word lexicon.

Mismatched lexicons

I've assumed so far that the languages being compared share word meanings. What if they don't?

Let's say we're comparing two lexicons of 1000 words each, and the chance of a single phonetic match is .03. If we know that the meaning of a word in language A is also found in language B, then we can expect about 30 exact semantic matches.

Now let's say that 1/2 of the meanings recorded in A ***aren't*** found in B. In effect, as we're comparing two words, there's only a 1 in 2 chance that the meaning we're looking at is even found in the other lexicon. So in our example, we'd expect 15 matches.

Another complication is that words are polysemous, and the particular set of meanings a word has in language A are not likely to be found in language B. At first glance this might be expected to lower the chances for a match; but from what I've seen of alleged resemblances, it increases it. For instance, Quechua ***simi*** means 'mouth, word, language'; a comparer is liable to feel free to match on ***any*** of these senses. In effect, polysemy brings with it its own increased semantic leeway.

Analyzing a claim

To analyze a claim about language relationship based simply on resemblances (as opposed, of course, to one based on the comparative method), we can apply the principles and formulas developed above.

We will need to know:

- The probability of a phonetic match (implicit in the claimant's comparisons)
- The semantic leeway: how many words in B the claimant will allow to match one in A
- The vocabulary size (we may know this directly, or estimate it from the phonology)

I analyze two examples below: Greenberg & Ruhlen's "world etymology" ***maliq'a*** 'swallow', and a comparison of Quechua and Hebrew posted to sci.lang.

Greenberg & Ruhlen

Greenberg & Ruhlen are comparing multiple languages. For a truly rigorous judgment we would need phoneme and root structure frequency analyses, as well as vocabulary size estimates, for each language concerned. This information is not available; but fortunately G&R's resemblance list alone is enough to deduce most of the parameters required.

To estimate the amount of phonetic leeway they allow, we can simply count the correspondences they allow. For instance, the vowels seem to be completely ignored. The middle consonant can be one of l, ly, lh, n, r, or zero-- (at least) 6 possibiliites. The end consonant can be one of g, j, d, k, q, q', kh, k', X, zero-- (at least) 10 possibilities.

The initial consonant must, it seems, be **m**. There is a reason for that, I think. Our brains, as psycholinguists have found, respond very strongly to initials. Find words in A and B that begin with the same letter, and the battle is half-won-- the brain is predisposed to find them very similar. Indeed, posters to sci.lang sometimes offer "resemblances" that correspond in nothing ***but*** the first letter. (The fact that they find the coincidence remarkable is more evidence for human beings' lousy intuition about probabilities.)

(It's also worth pointing out that a very high 7% of all words in my Quechua sample text begin with **m**. This initial isn't particularly common in Chinese, but one has to wonder if this choice of initial doesn't increase the possibility of false cognates, simply by being more common.)

Let's consider (arbitrarily) that G&R's languages have about 25 phonemes each. Then the ***minimum*** probability of a single, exact-meaning random match is 1/25 * 6/25 * 10/25 or .004. It must be emphasized that this is a minimum; if we saw more of G&R's potential cognates we might find that they allow more leeway than this. And of course it's only a loose estimate, because it's an abstract measure intended to cover any two arbitrary languages.

As for semantic matches, we can also form a lower bound on the semantic leeway they allow by counting the separate meanings in their glosses. I count at least 12; but I would consider it highly misleading not to at least double this number, and based on our lists of words related to 'eat' I'd consider 100 to be quite reasonable. If you can accept 'breast', 'cheek', 'swallow', 'drink', 'milk', and 'nape of the neck' as cognates, it's hard to seriously claim that you wouldn't accept 'eat', 'mouth', 'guzzle', 'vomit', 'suckle', and 'stomach'.

Let's say the semantic leeway is 25 words. Then the probability that we'll find at least one match for a word is $1 - ((1 - \boldsymbol{p})^{\boldsymbol{m}} = 1 - 0.996^{25} = .0953$.

How many random matches can R&G expect to find?

Let's assume the lexicon is 2000 words-- hopefully a good estimate of the size of the lexicons available for many of the obscure languages they work with. Our formula produces (omitting ranges with minimal probabilities):

p(126 to 150) = .0080

p(151 to 175) = .1222
p(176 to 200) = .6509
p(201 to 225) = .2214
p(226 to 250) = .0047

In other words we should expect close to ***200 random matches***, or a ***tenth of the lexicon***.

Of course, that's between two languages. What's the likelihood of finding matches **across languages**?

If the languages are related, of course, we would expect many non-random resemblances (though with G&R-style phonetic and semantic laxity we will find many random matches as well). We should not be cowed by the size of G&R's cognate list; there are multiple entries per family. They are not really comparing hundreds of languages, but a much smaller number of language ***families***.

There's no general number of random matches to expect this way, since language families vary in number of languages, and in how similar their lexicons are. But even closely related languages, like Spanish and French, have significant numbers of non-cognate words (***chien*** vs. ***perro, chapeau*** vs. ***sombrero, manger*** vs.***comer, regarder*** vs. ***mirar***, etc.). So if you don't find enough resemblances in language A1, you can try A2, A3, etc. Or even search dialects for cognates, as it seems they've done with Quechua and Aymara. The list of random resemblances between ***families*** will be larger than the list of random resemblances between ***languages***.

Another key question is how many families they're comparing. If G&R are right about certain high-level categories, such as Amerind and Eurasiatic, this turns out to be "three or four". If families A and B have 500 random resemblances, they'll have about 125 with family C and 31 with family D-- a quite respectable listing for "proto-World".

Quechua & Semitic

Now let's look at a list of Quechua and Semitic resemblances posted to sci.lang (and a level of scholarship which will make us miss Ruhlen & Greenberg).

Quechua		Semitic	
llama	llama	**gamal**	camel (Heb.)
qollana	leader	**kohen**	priest (Heb.)
t'eqe	rag doll	**degem**	model, specimen (Heb.)
qhapa	rich, powerful	**gabar**	become strong (Heb.)
qhoyu	group	**goi**	nation, people (Heb.)
qhoruy	cut off	**garaz**	cut (Heb.)
qhasay	burp	**gasah**	burp (Heb.)
q'enti	shorten, shrink	**gamad**	shrink (Heb.)
amaru	snake, sage	**amaru**	know, teach (Assyr.)
anta	copper	**homnt**	copper (Coptic)
atoq	fox	**bachor**	fox (Coptic)
aysana	basket	**tsina**	basket (Aramaic)
ch'olqe	wrinkle	**chorchi**	wrinkles (Coptic)
charki	jerky	**charke**	drought (Heb.)
cholo	Andean person	**chlol**	folk (Coptic)
wanaqo	guanaco	**anaqate**	she-camel (Assyr.)
churi	father's son	**chere**	son (Coptic)
illa	light, jewel	**ille**	brightness, light (Coptic)
k'ayrapin	pancreas	**kaire**	gullet, belly (Coptic)
kinuwa	quinoa	**knaao**	sheaf (Coptic)

k'ayra	frog	**krur**	frog (Coptic)
kutuna	blouse	**kutunet**	garment, tunic
onqoy	sickness, illness	**thomko**	ill use, afflict
punku	door	**brg**	be open (Coptic)
tarpuy	planting	**sirpad**	plant (Heb.)
hamuna	entrance	**amumuna**	city gate (Assyr.)
hillu	sweet tooth	**akkilu**	glutton (Assyr.)
huku	owl	**akku**	owl (Assyr.)
qoleq	silver	**purku**	gold (Assyr.)
p'uru	bladder, gourd	**buru**	vessel (Assyr.)
ch'enqo	small thing	**enegu**	suck (Assyr.)
watuq	diviner	**baru**	seer (Assyr.)
waliq	abundant	**baru**	become full (Assyr.)
ch'aphra	brush, twigs	**abru**	brush pile (Assyr.)
raphra	wing	**abru**	wing, fin (Assyr.)
apu	god, mountain lord	**abu**	father (Assyr.)
hatarichiy	build, incite, disturb	**adaru**	worried, disturbed (Assyr.)
hayk'api	how much?	**akka'iki**	how much? (Assyr.)
taruka	type of deer	**barih'a**	antelope (Assyr.)
umiqa	jewel	**banu**	headgear, diadem (Assyr.)
wawa	baby, child	**babu**	child (Assyr.)
p'uytu	well, puddle	**buninnu**	pond (Assyr.)
walla	domineering person, soldier	**baxalu**	ripe, youthful, manly (Assyr.)
wayra	wind, air	**billu**	low wind (Assyr.)
wanqara	drum	**balangu**	kettle-drum (Assyr.)
phiri	thick; dish made of flour and water	**buranu**	meal? (Assyr.)
perqa	wall	**birtu**	fetter; fortress (Assyr.)
phasi	steamed	**bashalu**	boil, roast (Assyr.)
maqt'a	young man	**batulu**	youth (Assyr.)
patana	stone seat	**apadana**	throne room (Assyr.)
qhapaq	rich, powerful	**gabshu**	massive, powerful (Ethiopic)
qocha	lake, pond	**gubshu**	mass of water (Assyr.)
llantin	type of herb	**gam-gam**	a herb (Assyr.)

There are a number of obvious problems with this list.

- The compiler (whose name I omit out of charity) knows nothing about the comparative method; no regular correspondences are presented.
- Nor does he know much about Quechua; he has for instance consistently taken the regular nominalizing suffix **-na** as part of the root. (Note what this does to a 'resemblance' like ***aysana / tsina***.)
- **-mu**- in ***hamuy*** is a movement suffix, leaving the rather unconvincing ***ha-/amumuna***. Likewise **-pi** in ***hayk'api*** is a locative suffix, leaving ***hay'ka/akka'iki.***
- Many of the resemblances are based on secondary meanings of Quechua roots. For instance, 'disturb' is a very secondary meaning of the causative ***hatarichiy***; the meaning of the basic root ***hatariy*** is 'rise up'.
- It's quite naive to compare individual Semitic languages with modern Cuzqueño dialect. On the Semitic side proto-Semitic or proto-Afro-Asiatic should be used; and on the Quechua side, reconstructed proto-Quechua. We also know some words in an even earlier form; for instance ***qocha***

is related to Aymara ***qota***-- which looks even less like the proposed cognate ***gubshu***.

However, my only concern here is to answer the compiler's question: "Is it a mere coincidence that there are so many correspondances between these languages?"

The above criticisms cannot answer this question; but the statistical model developed here can.

First let's estimate the degree of **phonetic** laxness the compiler is allowing. I'll use my Quechua frequency table, but I don't have similar data for the Semitic languages.

- He's fairly strict on initials. Some have two alternatives (h, q, t, w, ch'), but others match just one consonant (k, m, p, qh, r, ll, ch). Initial **a** matches four sounds. The pool of Semitic consonants matched is small: a, b, ch, d, g, k, p, s, 0. So the probability of an initial match is ***p***(h, q, t, w, ch') * 2/9 + ***p***(k, m, p, qh, r, ll, ch) * 1/9 + ***p***(a) * (4/9) which works out to .14.
- He'll allow about any medial vowels to match; but over half the time he does have a close vowel match. To estimate the probability of this we'd need to know the Semitic vowel probabilities, which we can hardly take as equiprobable-- in Quechua, for instance, 56% of medial vowels are **a**'s.
- Medial consonants match between 1 and 6 consonants: e.g. **k** matches k, g, h; **q** matches g, ch, q, k, t, 0. The pool of Semitic consonants matched is larger, with 16 members. The cumulative probability of a medial consonant match comes out to .524.
- Almost all the Quechua words end in a vowel, and this can match most anything as well. Where the Quechua word ends in a consonant, the comparer ignores it, except in the single instance ***llantin/gam-gam***.

37 out of the 54 matches involve just two matches (initial and medial); 17 have two medial consonant matches. Let's start by seeing how many **two-consonant matches** he can expect. The probability for a single phonetic match should be .17 * .524 = 0.089. For the resemblances with a vowel match as well, we can estimate a fifth of this or .018.

What's the level of **semantic** laxness? This is hard to say. Some matches are quite close (***burp, copper, basket, son, frog, owl***); others are fairly remote (***leader/priest; snake/teach, jerky/drought, pancreas/belly; sickness/afflict; door/open; sweet tooth/glutton; small thing/suck; god/father; build/disturbed; domineering/youthful; wall/fortress; stone seat/throne room; two herb names***). I think it would be quite conservative to assume 1 Quechua word can match 20 Semitic words in the compiler's mind.

Given this semantic leeway, the probability of a match on a single Quechua word is 1 - ((1 - $\boldsymbol{p}$)$^{\boldsymbol{m}}$ = 1 - 0.911^{20} = .845. That's quite telling right there-- it means that, given his phonetic and semantic laxness, the comparer is ***ordinarily*** going to find a random match for ***almost every Quechua word***.

My own Quechua dictionary has about 2000 non-Spanish roots. Our comparer will very likely find more than 1500 chance resemblances.

With a vowel match-- which, I emphasize, the comparer bothers with only half the time-- the match probability becomes .302, for roughly 600 chance resemblances.

It's just not that hard to match just two consonants. How about the **three-consonant matches**? Given the initial and medial consonant match probabilities calculated above, the probability of a single 3-consonant exact-semantic match is .17 * .524 * .524 = 0.047. With the same semantic leeway of 20 words, the probability of a match per word becomes 1 - 0.953^{20} = .618.

Our formula gives:

p(1101 to 1200) = .051
p(1201 to 1300) = .947
p(1301 to 1400) = .014

Is it a mere coincidence that there are so many correspondences between these languages? No, it isn't; what would be really surprising would be if there weren't a thousand more of the same quality.

With a vowel match, the match probability becomes .171 and we'd expect over 300 resemblances. However, there are only eight words in the list that can be described as matching three consonants and a vowel. Three of those are ruined by including ***-na*** as part of the root, and the remaining five include such uninspiring matches as ***q/t*** and ***ll/g***.

Characteristics of the model

What we have seen offers quantified support for what many linguists would have suspected: **the number of chance resemblances soars as phonetic and semantic matches are loosened.**

The actual variation is almost **linear**. That is, allow word ***a*** to match ***n*** words phonetically and ***m*** words semantically, and you very nearly increase the expected number of matches by ***n * m***.

Thus, the **results depend almost entirely on the value of *n* and *m***, and small variations in either ***n*** or ***m*** become very **large variations** in the number of matches.

Calculating an expected number of matches, then, it is essential to estimate phonetic and semantic laxness **from the claim being evaluated**. There is no such thing as a general "number of random chances expected". It depends almost entirely on what you count as a match.

Equally, it's necessary to **carefully evaluate any probability calculations** offered by the comparer. Comparers typically present calculations for extremely narrow matches, and then give very loose matches in their word lists. And they often ignore semantic looseness entirely. Such errors are not trivial in this game; they can produce numbers that are off by several orders of magnitude.

Why are we so easy to fool?

Why do people fool themselves so easily in this area? Why is it so hard for even highly intelligent people to convince themselves that random matches will be few? I think there's several reasons.

- **We want to be fooled**. The idea that far-flung languages are related (and with them their speakers) is intriguing. Proto-World is exciting. Slogging through the comparative method, by contrast, is dull and too often gives the completely unwanted answer "I don't know."
- **The brain is a pattern matchin' machine**. We evolved to quickly find patterns in the world. It's not hard to see that this could be a matter of survival (***that red striped plant gives you a stomach ache***) or give an edge in social competition (***I know that the red striped plant is bad for you and Tumba doesn't***; or, ***Roger's eyebrows flutter when he has a good hand***). It's so useful to find patterns that the brain is very tolerant of false patterns.
- **The brain is no good at probabilities**. We simply have no great intuitive feel for probabilities. Most people's eyes glaze over when you start talking about the chances of the chances of ***r*** events among ***n*** objects over ***t*** trials with a single event probability of ***p***.

When it comes to specific calculations of probability, which so often "prove" that the chance of a random match is vanishingly small, more factors come into play.

- **We're skeptical only of numbers we don't like**. You don't have to be trying to defraud anyone to fool yourself. But it's hard not to avoid our own bias in favor of numbers that go the way we want them to. If they don't, we're displeased, and skeptically examine our calculations, and seize on new assumptions that create better numbers. If the numbers do behave, we look over our results and our assumptions more carelessly. (Feynman gives a nice example: an incorrect estimate of a physical constant, which took decades to be corrected, each correction edging toward the correct value. Since the original estimate was assumed to be correct, scientists were more than usually cautious about new estimates that deviated from it, and easily convinced themselves that they'd made a mistake somewhere. Estimates near the accepted value were given more slack.)

- **We don't double-check our results against reality**. Few who make these calculations have ***looked*** for random matches in languages they're sure are ***unrelated***. In other words, they have no control case to check their results against. It's no wonder they never notice how common random matches really are.

  (I ***have*** looked for random matches, and found plenty of them; see my lists of Chinese/Quechua and Chinese/English pseudo-cognates.)

- **We haven't worked the numbers**. Even trained linguists, though they know that random matches will occur, generally can't say how many.
- **Statistical or linguistic ignorance**. The calculations are often ruined by elementary statistical errors, or by wildly unrealistic linguistic assumptions, or by disregard of the enormous phonetic and semantic leeway comparers allow themselves.

This document can hopefully help out in this area. The calculations are here and you can use them to evaluate claims (amateur or professional), or refer others to them.

A computer program for calculating matches

Here's a very simple computer program for applying the probability calculation described above. It will work for most C compilers, but if the probability is very high, you'll need a very robust `double` data type, which I have on the Mac but not Windows.

The important parameters are the lexicon size **n**, the probability of a phonetic match **p**, and the semantic leeway **semN** (how many meanings count as a match).

The program uses some algebraic tricks to simplify the calculation of the binomial probability. For instance, we're always dividing by r!, but we always calculate the probability for each r starting at one. So we can start with rfac = 1.0, and just multiply by r each iteration to get r!.

The rest of the factorial calculation is `n! / (n-k)!`. For r = 1, this is `n! / (n - 1)!` which can be rewritten `n (n-1)! / (n-1)!`; the `(n-1)!` terms cancel, leaving us with just n. For r = 2, we have `n! / (n - 2)!`, which is the same as `n (n-1) (n-2)! / (n-2)!`, which reduces to `n (n-1)`. That in turn is just (n-1) times the value we got for r = 1. It works out that factor for each value of r is just (n - r + 1) times the value for the previous iteration, and that's how `nfac_nrfac` is calculated below.

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>


main( void )

{

    double n   = 2000;    // lexicon size

    double p   = 0.002;   // match probability

    int semN   = 10;      // semantic leeway

    int stopat = 70;      // stop calculating at this r

    int bin    = 10;      // report probs within ranges of this size


    int    r;

    double q = 1 - p;
    double rfac = 1.0;
    double nfac_nrfac = 1.0;
    double cump = 0.0;
    double pr;
    double ptor = 1.0;

    p = 1 - pow(1 - p, semN);
    printf( "Probability %7.5lf\n\n", p );
    q = 1 - p;

    for (r = 1; r <= stopat; r++)
    {
        rfac *= r;
        nfac_nrfac *= n - r + 1;
        ptor *= p;
        pr = nfac_nrfac / rfac * ptor * pow( q, n - r );

        cump += pr;

        if (bin > 1)
        {
             if (r % bin == 0)
             {
                 printf( "p( %3i to %3i ) = %le\n",
                          r - bin + 1, r, cump );
                 cump = 0.0;
             }
        }
        else
            printf( "p(%i) = %le\n", r, pr );
    }

    if (bin == 1)
        printf( "\nCumulative p is %le\n", cump );
}
```

Here's the program output generated when the parameters are set as shown. Note that stopat and bin are reporting parameters. **stopat** = 70 told the computer to calculate propabilities for r = 1 to 70; **bin** = 10 told it to report these probabilities in groups of 10. With a little trial and error you can adjust these values for the most useful reporting of results.

```
   Probability 0.01982
p(   1 to  10 ) = 1.695872e-08
```

```
p(  11 to  20 ) = 4.004021e-04

p(  21 to  30 ) = 6.627584e-02

p(  31 to  40 ) = 4.979988e-01

p(  41 to  50 ) = 3.904203e-01

p(  51 to  60 ) = 4.403604e-02

p(  61 to  70 ) = 8.649523e-04
```

The probability reported at the top is the phonetic probability, adjusted to reflect the given semantic leeway.

When the probability (after the semantic adjustment) is very low, .002 or less, you should set the bin size to 1 for best results.

Acknowledgments

Thanks to the people on sci.lang, particularly Jacques Guy, Mark Hubey, Ross Clark, Mikael Thompson, and Brian M. Scott, for discussions which led to the writing of this paper.