Language Learning Diary - Part Eight

Linas Vepštas

Sept 2022

Abstract

The language-learning effort involves research and software development to implement the ideas concerning unsupervised learning of grammar, syntax and semantics from corpora. This document contains supplementary notes and a loosely-organized semi-chronological diary of results. The notes here might not always makes sense; they are a short-hand for my own benefit, rather than aimed at you, dear reader!

Introduction

Part Eight of the diary consists of two parts. First, a very short collection of notes on hypervectors and Gaussian orthogonal ensembles. Collected here as a handy reference, because adequate articles on some of these ideas do not yet exist. The second part is an exploration of applying the idea of Gaussian orthogonal ensembles to a current dataset. Overall, a tremendous success! This appears to provide an excellent metric of word-similarity! Victory!

Summary Conclusions

A summary of what is found in this part of the diary:

- The first few sections discuss hypervectors. These are interesting, but do not appear to be directly useful in the current situation.
- The next section is titled "Gaussian Orthogonal Ensemble" (but this name is misleading and incorrect. I don't have a better name yet). In Chapters Three and Five, it was noted that the symmetric-MI of word pairs (built up out of disjuncts) is distributed as a Gaussian. Whenever one has such a situation, the points can be understood to be vectors on a high-dimensional unit sphere. In this case, associated to each word w is a unit-length word-vectors $\hat{w}$ which is uniformly distributed on the unit sphere S_{N-1} where N is the size of the vocabulary.
- Such a distribution is perfectly uniform, whenever the sampling is taken from a perfect Gaussian; this is how uniform distributions on spheres are defined.

- Given two vectors $\hat{w}$ and $\hat{u}$ drawn from a uniform distribution, we expect the angle between them: $\theta = \arccos(\hat{w} \cdot \hat{u})$ to be uniformly distributed on the interval $[0, \pi]$.
- The actual word-data is approximately uniform. The current favorite dataset is explored. We find that the word-vectors $\hat{w}$ are "almost" uniformly distributed, except that:
  - There are very few word-pairs for which $\theta \lesssim \pi/6$ and that those which do lie in this region are strongly similar, grammatically. By visual inspection, the similarity is excellent. This is the primary, most important result of this diary chapter.
  - There are few or no word-pairs for which $\theta \gtrsim 3\pi/4$. This means that there aren't any words which are "truly grammatically dissimilar from one-another". This result does not seem to be important or usable; just curious.
- Given the collection of values $\theta(w,u)$ over pairs (u,w), the trick of projecting to a sphere can be repeated, giving another set of vectors. This can again be repeated, *ad infinitum*. The first repetition is explored. It looks OK; it provides similarities that seem to be just about as good as the original sphere set, maybe every so slightly worse. This second recomputation of similarities requires additional CPU time, which is considerable, and so does not seem worth the effort. It's measuring some hard-to-comprehend second-order effect in the distribution of grammatical similarity. That is to say, it's physical interpretation is unclear.

That's it. Moving forward, it is clear that $\theta = \arccos(\hat{w} \cdot \hat{u})$ is the superior metric for measuring word grammatical similarity. And BTW, it is a true metric, satisfying the triangle inequality. It should form the foundation for future clustering work.

Bipolar Hypervectors

A bipolar hypervector is a vector in a D-dimensional space having values in the set $\mathbb{Z}_2 = \{-1, 1\}$; that is, a vector in $\{-1, 1\}^D$. Every hypervector corresponds to a vertex of a hypercube centered at the origin of a D-dimensional space. Bipolar hypervectors have the interesting property that, for D even, given any (random) vector v, if one flips half the bits to get a vector w, then v and w are orthogonal!

This interesting property can be used to map point sequences ("curves") to sequences of bipolar hypervectors such that the endpoints of the point sequence are orthogonal, and intermediate points have increasingly larger cosine distances. Geometrically, this maps the point sequence to a sequence of corners on the hypercube that are increasingly distant from the starting point. of point sequences to hypervectors

This map is a homomorphism that preserves the metric on the point sequence. This metric property can then be deployed to simplify classification problems, by mapping the space to be classified to vector arithmetic and cosine distances. These two ideas are developed below.

Metric properties

Let $[p_0, \cdots, p_N]$ be a totally ordered sequence of points. The total order is just $p_i < p_j$ for $i < j$ for integer index i, j. This order can be metricized with a metric g such that $g(p_i, p_j) = |i-j|/N$. The metric is normalized written so that the maximum distance is 1. Pick a dimension $D > 2N$ and conventionally $D \gg 2N$ and an arbitrary initial (random) vector v_0 that will correspond to p_0. Generate a sequence of bipolar hypervectors v_k as follows. Given v_i, select (randomly) $D/2N$ bits that have not been selected before, and flip them, to obtain v_{i+1}.

The above generates a sequence of (bipolar hyperdimensional) vectors with the following properties. The dot product is

$$v_k \cdot v_k = D$$

For neighboring points, the dot product is

$$v_k \cdot v_{k+1} = D\left(1 - \frac{1}{N}\right)$$

because these differ in $D/2N$ bit locations. (The Hamming distance is $D/2N$; so $D/2N$ bit positions that are $+$ are replaced by $-$, and so that total sum decreases by N.)

In general, the Hamming distance between v_k and v_{k+n} is $nD/2N$ and so the dot product is

$$v_k \cdot v_{k+n} = D\left(1 - \frac{n}{N}\right)$$

or equivalently

$$v_i \cdot v_j = D\left(1 - \frac{|i-j|}{N}\right)$$

so that

$$v_0 \cdot v_N = 0$$

are orthogonal. The Hamming distance between orthogonal vectors is necessarily $D/2$.

Normalizing by D and subtracting from 1 reproduces the original metric on the point sequence: i.e.

$$g(p_i, p_j) = \frac{|i-j|}{N} = 1 - \frac{v_i \cdot v_j}{D}$$

Of course, all this machination is pointless for one-dimensional point sequences. So ... for the more complex case.

Dimensional Oxidation

In chemistry, oxidation is the opposite of reduction. If dimensional reduction is the reduction of the number of dimensions to describe a dataset, then playing on this, dimensional oxidation is the act of increasing dimensions.

One conventional machine learning problem is the classification of regions of some M-dimensional space features. That is, there are a set of real-valued features $f_m \in \mathbb{R}$

for $1 \leq m \leq M$. These are presumed to be bounded, so that $f_m^{\min} \leq f_m \leq f_m^{\max}$ so that these can be normalized to the unit cube $[0,1]^M$ by writing

$$x_m = \frac{f_m - f_m^{\min}}{f_m^{\max} - f_m^{\min}}$$

Each unit interval may be digitized (partitioned) into $N+1$ distinct sub-intervals. This partitioning $[p_0, \cdots, p_N]$ then provides a totally ordered points sequence that can be mapped to bipolar hypervectors. A given point $x \in [0,1]^M$ is thus mapped to M vectors v_m. Summing these provides a mapping of the unit cube to $\mathbb{Z}^D$:

$$w = w(x) = \sum_m v_m$$

Since the sum is bounded between $-M$ and M in each direction, this vector lies on the hypercube lattice $(2M)^D$.

If these vectors are normalized to unit length, then they live on the surface of the hyper-sphere S_{D-1}. In general, these points are not random, evenly distributed on the hyper-sphere. In a narrow sense, we can offer a theorem: the points are randomly distributed on the hyper-sphere if and only if they are randomly distributed in the unit cube $[0,1]^M$. However, in a broader sense, when $N \ll D$, points are increasingly scattered on the unit sphere, becoming increasingly uniformly distributed. This is effectively because the start and end points of the unit interval are mapped to random hypervectors. (This follows(?) because random hypervectors have a binomial bit distribution, and for large D, the binomial distribution approaches the Gaussian).

So again, we seem to approach the case of a Gaussian Orthogonal Ensemble. Kind of...

Efficient Dimensional Oxidation

A way to encode low-dimensional classification problems into hypervectors that sharply improves on the naive uniform feature-space digitization is described in Basaklar, *et al.* "Hypervector Design for Efficient Hyperdimensional Computing on Edge Devices" https://arxiv.org/pdf/2103.06709.pdf

The presumption is that there is a preexisting training dataset, the parameter space is low dimensional, and the number of clusters is fixed.

The solution is to divide up the parameter space in a non-uniform kind of way, devoting lots of extra hypervector bit-flips to the boundary zones between clusters, so that the boundaries of the clusters can be cleanly distinguished. This is done by specifying an integer optimization problem. The trick is to (i) maximize the training accuracy and (ii) minimize the similarity between class encoders, subject to (iii) orthogonalization of parameter endpoints. Because this is an integer optimization problem, a genetic algorithm is used to perform the search. The paper provides details.

This is not directly relevant for us, because we don't have a training set, nor do we know *a priori* how many clusters there will be.

Factoids

Assorted notes:

- Almost all random vectors are orthogonal to one another (or nearly so). This follows from binomial coefficients being approximations for Gaussians. There are
$$\binom{D}{D/2} = \frac{D!}{(D/2)!^2} \approx \sqrt{\frac{2}{\pi D}}\, 2^D$$
orthogonal vectors, which follows from Stirling's law $n! \approx \sqrt{2\pi n} e^{-n} n^n$. There are
$$\binom{D}{\frac{D}{2}+1} = \frac{D!}{\left(\frac{D}{2}-1\right)!\left(\frac{D}{2}+1\right)!} \approx \sqrt{2/\pi D}\, 2^D xxx$$
almost orthogonal vectors, that differ by one bit. More generally, for $n \ll D$ there are
$$\binom{D}{\frac{D}{2}+n} = \frac{D!}{\left(\frac{D}{2}-n\right)!\left(\frac{D}{2}+n\right)!} \approx \sqrt{2/\pi D}\, 2^D \left(1 - \frac{2n^2}{D}\right)$$
vectors that differ by n bits. (Need to double check this, might be errors).
- Given two corners on a hypercube differing by d bits, there are 2^d shortest paths between them.
- The midway points on such paths can have larger Jacquard (Hamming) distances to each other, than to either endpoint. In fact, this will almost always be the case.
- If the vectors are normalized, they can be seen to live on the surface of a sphere (of the same dimension).
- Ternary hypervectors i.e. elements of $\{-1,0,1\}^D$ form a field under point-wise multiplication and sgn applied to arithmetic addition.
- The projection of lattice points in $\mathbb{Z}^D$ to the unit sphere S_{D-1} is presumably dense. No clue if its "uniformly" distributed; presumably its not, much like the rationals in the unit interval.
- No clue what the analogs of the modular group or the fundamental domain are.

Spin Glasses

Cribbed notes from Michel TALAGRAND "Mean Field Models for Spin Glasses Volume I: Basic Examples" (2010) Springer-Verlag.

Gaussian Orthogonal Ensembles

Well, not really; that's just the working title. There's no actual ensemble.

The thing to explore is this: suppose there's an index $w, u \in \{1, \cdots, N\}$ i.e. N-dimensional. Suppose there are numbers $f(w,u) = f(u,w)$ which are distributed with a normal distribution with mean μ and stddev σ. Experimentally, we compute

$$\mu = \frac{1}{N^2} \sum_{u,w} f(u,w) = \langle f \rangle$$

and

$$\begin{aligned}
\sigma^2 =& \frac{1}{N^2} \sum_{u,w} (f(u,w) - \mu)^2 \\
=& \frac{1}{N^2} \sum_{u,w} f^2(u,w) - \mu^2 \\
=& \left\langle f^2 \right\rangle - \langle f \rangle^2
\end{aligned}$$

as usual. Define $g(u,w) = (f(u,w) - \mu)/\sigma$ and then $g(u,w)$ is normally distributed about zero with unit stddev.

Fix w and define an N-dimensional vector $\vec{w} \in \mathbb{R}^N$ whose vector components are $w_u = g(w,u)$. Normalize them to unit length, so that $\hat{w} = \vec{w}/\|\vec{w}\|$ where $\|\vec{w}\| = \|\vec{w}\|_2$ is the Euclidean norm. This is the Gaussian orthogonal ensemble. There are N of these vectors, they are uniformly distributed on the $N-1$ sphere S_{N-1}.

The experimental goal is to obtain these vectors, on the actual datasets, were $f(w,u)$ is the MI for two words, the MI being given via the disjunct+shape formulas explored in earlier chapters. So lets see what we get.

Ranking

The ranked MI has the form $f(w,u) = s(w,u) - r(w) - r(u)$. How does this change the above?

$$\begin{aligned}
\mu =& \frac{1}{N^2} \sum_{u,w} f(u,w) \\
=& \frac{1}{N^2} \sum_{u,w} s(u,w) - \frac{2}{N} \sum_{w} r(w) \\
=& \langle s \rangle - 2 \langle r \rangle
\end{aligned}$$

and next

$$\begin{aligned}
\sigma^2 =& \frac{1}{N^2}\sum_{u,w} f^2(u,w) - \mu^2 \\
=& \frac{1}{N^2}\sum_{u,w} [s(u,w) - r(w) - r(u)]^2 - [\langle s\rangle - 2\langle r\rangle]^2 \\
=& \frac{1}{N^2}\sum_{u,w} \left[s^2(u,w) - 4s(w,u)\, r(w) + 4r^2(w)\right] - \left[\langle s\rangle^2 - 4\langle s\rangle\langle r\rangle + 4\langle r\rangle^2\right] \\
=& \left\langle s^2\right\rangle - 4\langle sr\rangle + 4\left\langle r^2\right\rangle - \langle s\rangle^2 + 4\langle s\rangle\langle r\rangle - 4\langle r\rangle^2 \\
=& \left[\left\langle s^2\right\rangle - \langle s\rangle^2\right] - 4[\langle sr\rangle - \langle s\rangle\langle r\rangle] + 4\left[\left\langle r^2\right\rangle - \langle r\rangle^2\right]
\end{aligned}$$

and so in general we see this is not equivalent to plain f and so we need to keep track of both, separately.

Experiment-15 (21 Sept 2022)

The data is in experiment-15. MI's are computed for the N top-ranked words. Keep in mind that the MI's here are the symmetric MI's obtained from word-disjunct vectors.

Datasets containing MI-similarities:

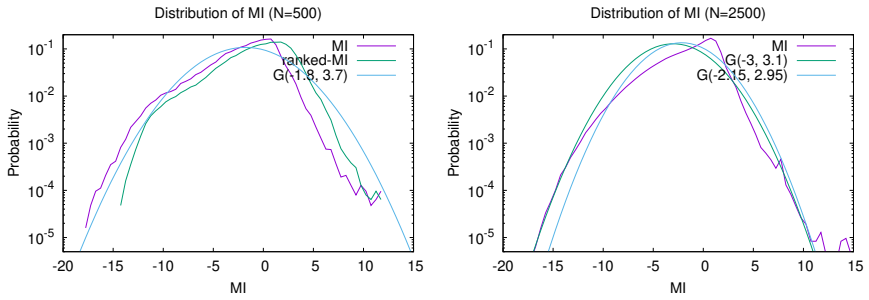
dataset	N	N_{pairs}	RAM	cpu to load
r14-sim200.rdb	200	20100	4.3GB	11 min
r15-sim500.rdb	500	125250	4.6GB	12 min
r15-sim2500.rdb	2500	3126250	6.4GB	21 min
r15-sim6000.rdb	6000	18003000	16.3GB	42 min

Each larger dataset is built from the smaller one; this allows experiments to be run in parallel; otherwise, its the same data. The N_{pairs} is just the number of SimilarityLinks in the dataset. The number of pairs is exactly $N(N+1)/2$. The similarities computed as described earlier, using the cross-sections and shapes as a part of the vectors. RAM consumption includes that for shapes/cross-sections as well as similarities. The CPU-time-to-load includes the CPU-time to load shapes and cross-sections. These are needed for computing MI, but are not needed for the GOE computations (however, they are needed to get a list of ranked words.)

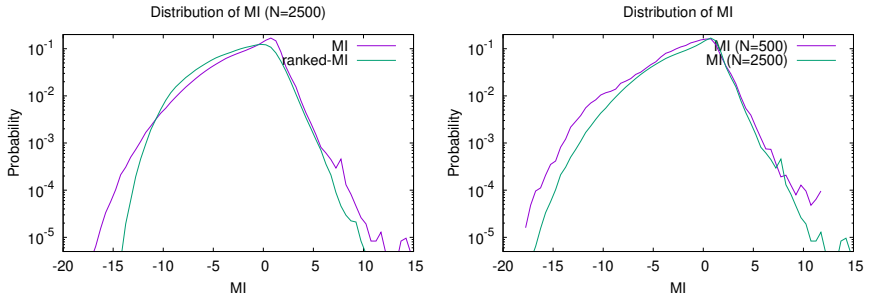
Here are the means and variations, for various different values of N:

dataset	N	μ_{MI}	σ_{MI}	$\mu_{\text{ranked-MI}}$	$\sigma_{\text{ranked-MI}}$
r14-sim200.rdb	200	-1.6966	3.2703	0.5947	3.3151
r15-sim500.rdb	500	-1.8062	3.1579	-0.4925	3.2165
r15-sim2500.rdb	2500	-1.4053	2.8985	-2.1492	2.9518
r15-sim6000.rdb	6000	-0.7240	2.6398	-2.7543	2.6019

Below are some graphs showing this dataset.[1] It's not as pretty as some of the earlier graphs. Why? See, for example, page 14 and page 25 of chapter three (ranked-MI here is called common-MI there). But it is comparable to those in the second half of the diary chapter five (pages 19, 20, 38-40), so I guess that's OK.



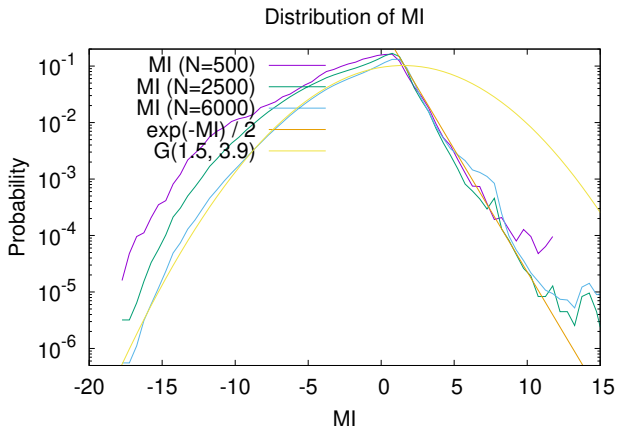
Above left: the Gaussian shown has the same parameters as the earlier ones, in chapter three. Here N=500 pairs vs N=1200 in chapter three. Also shown is ranked-MI, which shifts slightly to the right. Above right: just the MI, with two Gaussians. One is an "eyeballed fit", adjusting the Gaussian to match the tail of the distribution. This is the G(-3,3.1) Gaussian. The other shows the results of computing the mean and stddev of data: $\mu = -2.1492$ and $\sigma = 2.9518$ as reported in the table above. The mean is clearly influenced by the blip in the data just above the mean. The tails have very little influence on the calculation of the mean and stddev ... and yet... maybe they should? Maybe we should be fitting to the tails with equal weight as the center?



Above left: compare MI to ranked-MI without the visual distraction of the Gaussian. The right hand side looks a lot more linear. This linear-like right-hand-side can also be seen in graphs in chapters three, five, but not as pronounced as here. Above right: compare distributions for varying N.

Below, do it again, up to $N = 6000$.

[1] Data collected with first 50 lines of utils/orthogonal-ensemble.scm from data in experiment-15. Plotted by p8-goe/sim-mi-dist.gplot.



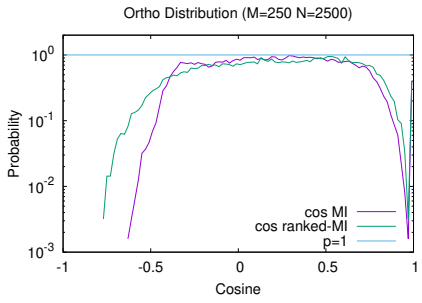
The above extends the distribution to $N(N+1)/2 = 18003000$ or 18 million pairs. The linearity on the right is maintained, and seems to have the particularly simple form of $e^{-MI}/2$ while the left side is imperfectly Gaussian (seems to drop off even faster). Recall that this dataset has some fair amount of garbage in it (due to bad escaping of quotes) and also has been trimmed; so its not clear whether the linear right is "real data" or "garbage" or "artifact of trimming".

GOE Distributions

So, using MI and RMI, define the Gaussian vectors as described at the beginning of this section. The base space has dimension $N = 2500$ – i.e. we've computed MI and RMI for the top-ranked 2500 words, for a total of $N(N+1)/2 = 3126250$ word-pairs. That is, we have $N = 2500$ vectors $\vec{w}$. Out of these, take the M top-ranked words, and compute the cosines

$$\cos(\theta_{wu}) = \hat{w} \cdot \hat{u} = \frac{\vec{w} \cdot \vec{u}}{\|\vec{w}\| \, \|\vec{u}\|}$$

The graph below shows the cosine distribution for the top $M = 250$ words, or $M(M+1)/2 = 31375$ pairs. There are two graphs, actually, one for vectors built from MI and another from RMI.[2]



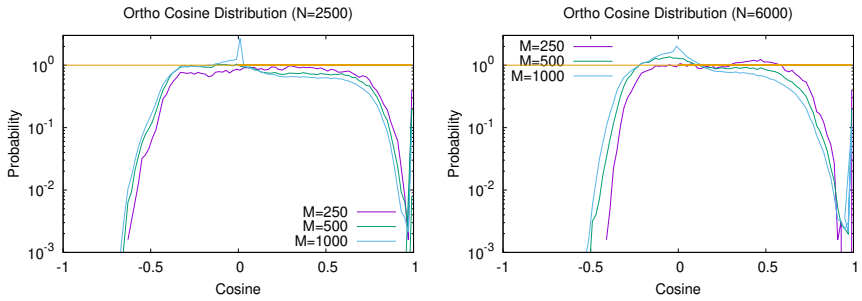
[2]Graphs built with code in util/orthogonal-ensemble.scm lines 165-200 and p8-goe/cos-mi-dist.gplot. Datasets are r15-got-2500.rdb and similar.

The distribution is flat-topped. This is exactly what we'd expect, if the initial distribution was perfectly Gaussian. But its not, so there's a fall-off. The peak at $\cos\theta = 1$ is for the self-similarity. Very curious is the notch right below $\cos\theta = 1$. This notch indicates an actual repulsion. Word are actually being repelled – there is a minimum cosine distance! That is, the word-vectors are uniformly distributed across the sphere S_{N-1} except for two things: there is a repulsive force, that prevents vectors from getting close to each-other, and depopulates near $\cos\theta = 1$. There is also an "inverted" force, depopulating a much larger region around $\cos\theta = -1$.

The depopulated region around $\cos\theta = -1$ seems like it should be "less interesting". It seems to be saying that there are no words which are truly "different" from all other words.

Perhaps the notches can be parameterized by a repulsive force... how?

Again, below-left, this time showing the distributions for the top top $M = 250$ words (as before), the top $M = 500$ words and the top $M = 1000$ words.[3] Below right are the same three curves, except this time the vector length was $N = 6000$.



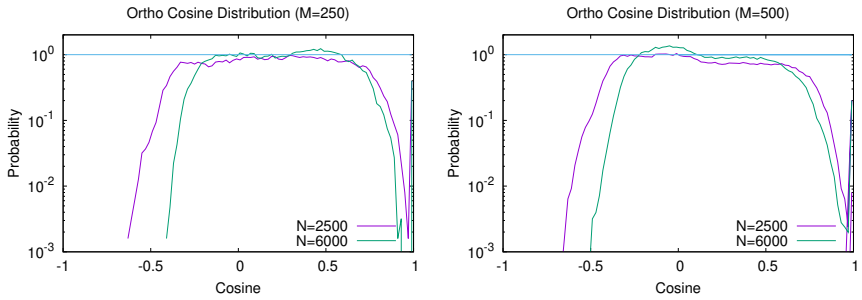
Much as before; slightly less flat. Notch on the right is slightly larger; the notch on the left is slightly smaller. The $M = 1000$ has a spike at precisely cosine=0. This is due to the appearance of word-pairs that have nothing in common: either one or the other has a zero vector component, thus the dot product is zero. A zero vector component corresponds to $MI = -\infty$ i.e. a pair of words which have no disjuncts in common. The appearance of this spike suggests we've reached the limit of the utility of the vectors: longer vectors are needed.

The mean and stddev of the $N = 2500$ and $M = 1000$ graph is $\mu_F = 0.09075$ and $\sigma_F = 0.3389$. This is used to construct F_2 described several sections below.

Again, this time showing what happens when the vector lengths are varied. One the left, the distribution of dot products for the top-ranked $M = 250$ words (as before), but compares vectors of length $N = 2500$ and $N = 6000$. The plateau is narrower. On the right, same, but the distribution of dot products for the top-ranked $M = 500$ words.[4]

[3] Graphs built with code in util/orthogonal-ensemble.scm lines 203-232 and p8-goe/cos-mi-dist-500.gplot.

[4] Made with cos-mi-dN.gplot from scripts as above.



Repulsion estimate

Consider two vectors that point at the corners of an N-cube. Nearest-neighbor corners are then a distance $\cos\theta = (N-2)/N$ apart, or $1-\cos\theta = 2/N$ For $N = 2500$ this is $2/2500 = 0.0008 = 8\times 10^{-4}$. By comparison, the right-hand gap above is at about 0.9, so this is over 100 nearest neighbors apart. For $N = 6000$ this is even farther apart.

If one selects J random corners of the N-dimensional hypercube, what is the average distance between them? Here, the hypercube is $\{-1,+1\}^N$. That is, select J random bit-strings of length N – what is the average Hamming distance between them? In the limit $N\to\infty$, this is famously a Gaussian. In the non-limit, what is it? This is a combinatoric question. There are a total of 2^N such vectors. For a fixed initial vector, there are N vectors that differ by one position, and $N(N-1)/2$ that differ by two bits, and the general case is given by the binomial coefficient $\binom{N}{k}$ for k bit differences. Note that

$$\sum_{k=0}^{N}\binom{N}{k} = 2^N$$

so we've counted them all. The average Hamming distance is then

$$\langle k\rangle = \frac{1}{2^N}\sum_{k=0}^{N} k\binom{N}{k} = \frac{N}{2}$$

and the mean-square distance is

$$\langle k^2\rangle = \frac{1}{2^N}\sum_{k=0}^{N} k^2\binom{N}{k} = \frac{N^2+N}{4}$$

and so the rms is

$$\sigma_k = \sqrt{\langle k^2\rangle - \langle k\rangle^2} = \frac{\sqrt{N}}{2}$$

Given two bit-vectors of length N differing in k bits, the cosine between then is $\cos\theta = 1-2k/N$. We thus immediately conclude that

$$\langle\cos\theta\rangle = 0$$

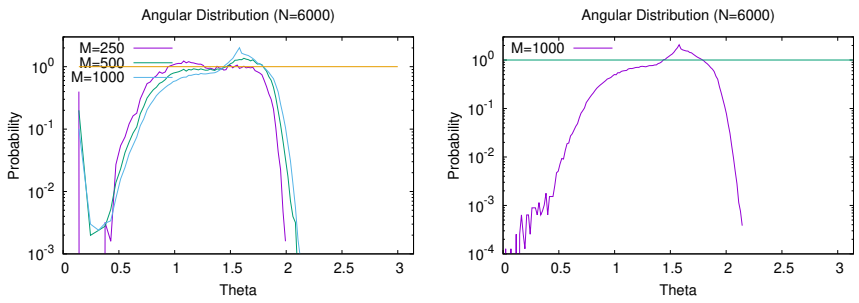
and

$$\begin{aligned}\left\langle \cos^2\theta \right\rangle &= \left\langle 1 - \frac{4k}{N} + \frac{4k^2}{N^2} \right\rangle \\ &= 1 - \frac{4\langle k\rangle}{N} + \frac{4\left\langle k^2\right\rangle}{N^2} \\ &= 1 - 2 + \frac{N^2+N}{N^2} \\ &= \frac{1}{N}\end{aligned}$$

and so the rms is $\sqrt{\langle\cos^2\theta\rangle} = 1/\sqrt{N}$. For $N = 2500$ and 6000, we have $\sqrt{\langle\cos^2\theta\rangle} =$ 0.02 and 0.013. Again, the right-hand gap widens, not narrows, as the vector-length increases. The stddev does not provide a natural scale.

Anyway, this whole approach is just-plain wrong. Selecting random hypervectors gives us the Gaussian distribution in the limit, yet we're seeing a flat-topped distribution. That's because the MI was Gaussian; the flat-top is coming from that. In other words, for x uniformly distributed in $[-1,1]$ we should be considering $\langle x^2\rangle = 2/3$. The natural scale is $\sqrt{\langle x^2\rangle} = \sqrt{2/3} = 0.8165$. Hmmm. What does this mean?

None of this is getting us any closer to "estimating the repulsion between vectors". Lets try again. Here's the same data, except this time, the x-axis is given by $\theta = \arccos(\hat{w}\cdot\hat{u})$. For a perfect uniform distribution, we should see θ uniformly distributed over $[0,\pi]$. Instead, we get the below left:

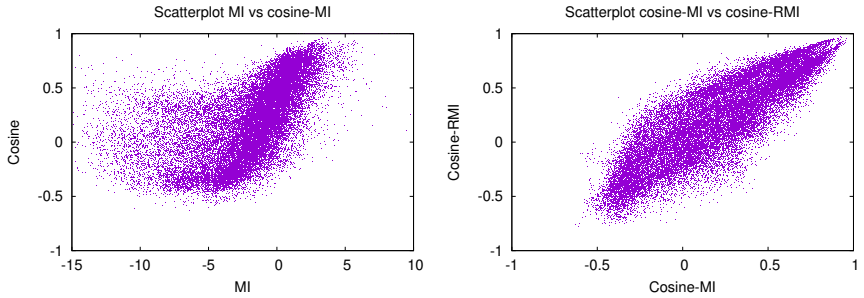


So the repulsive region is for $\theta \lesssim 0.5 \approx \pi/6$. The above right is the $M = 1000$ curve only, after removing the diagonal entries (which all have a cosine of exactly 1.) The arccosine is taken before binning, and 200 bins are used instead of 100, so that we can get a better view of what is happening near $\theta \approx 0$.

There are few or no word-pairs for which $\theta \gtrsim 3\pi/4$. This means that there aren't any words which are "truly grammatically dissimilar from one-another", but its unclear just how "deep" this insight is. Is this saying that the English grammar is not "random"? That it's impossible to be "grammatically dissimilar" in general? The meaning of this is unclear. It does not appear to be important, though; whatever insight might be offered here, it does not seem worth the effort just right now.

Correlations

The scatter plot below left shows how the cosine-MI and the MI are correlated. The graphs for the other correlations (between cosine-RMI and MI, cosine-MI and RMI, and cosine-RMI vs RMI) look similar, with offsets and slightly different shapes.



Above right shows how cosine-MI and cosine-RMI are correlated. There seem to be striations. What are they?

Top most similar

Of the 250 words that were compared, here are two tables showing the most-similar word-vectors, and the cosine. The left table shows the top-20 most similar word-vectors. The right table is the list, between 100 and 120. Things look more marginal about 200 down – there are some good matches, e.g. “head”–”face” and some mediocre ones: “people”–”life”. Note that not very many words are being compared – 250 is not that big a vocabulary (Although the word-vectors themselves have dimension $N = 2500$.)

top-20 word vectors		
$\hat{w} \cdot \hat{u}$	$\vec{w}$	$\vec{u}$
0.9604	we	they
0.9525	And	But
0.9522	she	he
0.9466	they	he
0.9418	she	they
0.9412	we	he
0.9384	may	should
0.9351	when	how
0.9342	1	3
0.9322	she	we
0.9313	You	We
0.9308	people	men
0.9307	when	if
0.9263	is	was
0.9251	shall	should
0.9250	with	by
0.9245	where	how
0.9233	him	them
0.9215	had	would
0.9214	when	where

word vectors 100-120		
$\hat{w} \cdot \hat{u}$	$\vec{w}$	$\vec{u}$
0.8834	by	after
0.8834	she	when
0.8833	and	but
0.8832	after	who
0.8826	where	who
0.8823	what	which
0.8823	night	day
0.8815	And	Then
0.8807	had	could
0.8806	would	could
0.8805	had	has
0.8804	What	A
0.8790	eyes	hand
0.8790	in	to
0.8790	himself	away
0.8789	with	from
0.8788	eyes	head
0.8785	though	if
0.8782	This	What
0.8777	eyes	face

Below, as a reminder, is the top-20 list, on the same dataset, the same pairs, of the top-20, ranked by ranked-MI. Clearly a very different beast. There's lots of punctuation, because punctuation occurs very frequently, and thus gets ranked highly (recall, the ranking factor is the log of the frequency). Some of these are OK, such as "is"–"was" which has both high RMI and high cosine. But the abysmal cosine for the —, + and [, + pairs just says that the ranking factor is a better, more accurate measure of similarity.

top-20 by ranked-MI			
$\hat{w} \cdot \hat{u}$	RMI	$\vec{w}$	$\vec{u}$
0.3329	9.5244	....	–
0.1105	9.1864	—	+
0.7866	9.1428	;	,
0.9263	9.0413	is	was
0.8833	9.0177	and	but
0.7530	8.9218	.	?
0.8053	8.9079	!	?
0.6956	8.8475	It	He
0.2569	8.8309	[	+
0.6577	8.2733	”	"
0.7111	8.1875	No	A
0.7734	8.1764	in	of
0.9522	8.1732	she	he
0.6513	8.1305	It	There
0.8076	8.0921	and	as
0.7171	8.0558	!	.
0.4163	8.0537	\\\\\\\\\\\\\\"I	\\\\\\\"I
0.8941	8.0221	‘	“
0.4701	8.0113	–	’
0.8035	7.9720	the	his

Comparing these two, it suggests that the GOE similarity provides a much healthier concept of similarity.

However, the results are strongly dependent on word frequency. The tables above considered only the top-ranked 250 words. Here's the similarity for the top-ranked 400 and 700 words.

top-20 out of M=400		
$\hat{w} \cdot \hat{u}$	$\vec{w}$	$\vec{u}$
0.9773	\\\\\\\\\\\\\\"I	\"I
0.9662	7	6
0.9634	4	3
0.9604	we	they
0.9525	And	But
0.9522	she	he
0.9498	1	5
0.9483	7	10
0.9466	they	he
0.9464	father	mother
0.9418	she	they
0.9412	we	he
0.9411	10	6
0.9410	3	5
0.9384	may	should
0.9367	What	Why
0.9351	when	how
0.9342	1	3
0.9322	she	we
0.9313	You	We

top-20 out of M=700		
$\hat{w} \cdot \hat{u}$	$\vec{w}$	$\vec{u}$
0.9954	Ag	Ap
0.9866	26	13
0.9773	\\\\\\\\\\\\\\"I	\"I
0.9756	\\\\\\\"No	\\\\\\\"Well
0.9749	\\\\\\\"Well	\\\\\\\\\\\\\\"Yes
0.9734	11	29
0.9732	\\\\\\\"Well	\\\\\\\"Oh
0.9719	9	29
0.9708	10	13
0.9699	\\\\\\\"Well	\\\\\\\"Yes
0.9690	9	11
0.9662	7	6
0.9658	\\\\\\\\\\\\\\"Yes	\\\\\\\"Oh
0.9640	\\\\\\\"No	\\\\\\\"Oh
0.9634	4	3
0.9627	\\\\\\\"No	\\\\\\\\\\\\\\"Yes
0.9604	we	they
0.9577	\\\\\\\"Yes	\\\\\\\"Oh
0.9575	Ag	Jl
0.9563	10	26

It continues like this, the deeper we go. Conclude: there are lots of infrequent words that are even more similar than the frequent ones. All these matchup appear to be quite healthy, overall. No complaints.

Word analogy via vector additivity

The litmus test would be doing the word-vector sum "King-man+woman" to see what happens. I suspect the vocabulary is too small to do this, and that much larger datasets are needed. But lets try it anyway.

So "king-man+woman" fails, since apparently, "queen" is not well-represented e.g. king dot queen = 0.

Here are some that work:

husband-man+woman		brother-man+woman		boy-man+woman	
word	dot	word	dot	word	dot
husband	1.0892	brother	1.0565	boy	1.0854
wife	1.0051	wife	0.9705	woman	1.0126
woman	0.9511	husband	0.9625	lady	0.9229
brother	0.9298	sister	0.9519	girl	0.9126
mother	0.9237	mother	0.9205	husband	0.8872
king	0.9220	friend	0.9091	mother	0.8805
sister	0.9192	woman	0.9088	wife	0.8727
heart	0.9070	father	0.8781	boys	0.8724

So that's not too bad. This is a medium-small language model, and it gives plausible results. The fact that base vectors (husband, man, woman,...) show at the top suggests that the language model is just not large enough to defnititvely knock these out.

Vector sums between Paris, France, Spain, Germany, England, Berlin etc all fail completely; get similarities to "O", "E", "W", "Mr", "Mrs", "6d" – so garbage. Presumable cause the sample texts are not talking about countries or capitals. Wikipedia could solve this!?

Here are some more:

black-white+up		short-light+long	
word	dot	word	dot
up	1.0496	long	1.0339
off	0.9487	short	0.9931
down	0.9468	strong	0.8728
away	0.9384	hard	0.8702
himself	0.9256	quick	0.8595
around	0.9240	large	0.8322
once	0.9089	bright	0.8172
herself	0.8950	small	0.8122

After some effort, I couldn't find any more. The sample size really is too small. Using RMI instead of MI does not substantially change results.

Vector Layers and Hypervectors

In the prior section, we took an $N \times N$ matrix $F(u,w)$ whose matrix elements were distributed approximately as a Gaussian, and rescaled it to obtain a new matrix $G(u,w)$, having the same distribution, but now having a mean of zero and an stddev of 1. Such Gaussians can be interpreted as vectors uniformly randomly distributed on an $N-1$ sphere.

The cosine products of these vectors give a new matrix $F_2(u,w)$. The matrix entries are given by the cosines:

$$F_2(u,w) = \frac{\sum_v G(u,v)\, G(w,v)}{\sqrt{G^2(u,*)\, G^2(v,*)}}$$

where

$$G^2(u,*) = \sum_v G^2(u,v) = \|G(u)\|_2^2$$

The matrix elements $F_2(u,w)$ are now uniformly distributed on the unit cube $[-1,1]^N$ centered at the origin. They are no longer Gaussian. For high-dimensional N, this implies that the (row or column) vectors of F_2 are clustered along the diagonals.

Vector Layers

We can iterate on the construction process, and obtain G_2 and then F_3 and G_3 and then F_4 and so on. So, "layers". What are these layers? I asked this question at MathOverflow.

Some algebra might shed light. Let B be the matrix all of whose entries are one: i.e. $B(u,v) = 1$. Note that $B^2 = NB$ so its idempotent up to a constant. Alternately, $C(u,v) = 1/N$ so that $C^2 = C$ is idempotent. Then, given F, we had

$$G = \frac{1}{\sigma_F}(F - \mu_F B)$$

where μ_F and σ_F are scalars. Define a unitized matrix H by

$$H(u,v) = \frac{G(u,v)}{\|G(u)\|_2}$$

so that all of the row-vectors of H are unit-length. Then

$$F_2 = HH^T$$

where H^T is the transpose of H. Hmm. Plugging through provides no insight or simplification.

Note that BF is a matrix whose rows are all identical, and the ℓ_1 norm of each row is $N\mu_F$.

Hypervectors

For high-dimensional N, this implies that the (row or column) vectors of F_2 are naturally clustered along the diagonals. Because they already have this clustering, we can project onto a hypervector J in $\{-1,1\}^N$. Define J as

$$J(u,v) = \Theta(F_2(u,v))$$

where

$$\Theta(x) = \begin{cases} +1 & \text{for } 0 < x \\ -1 & \text{otherwise} \end{cases}$$

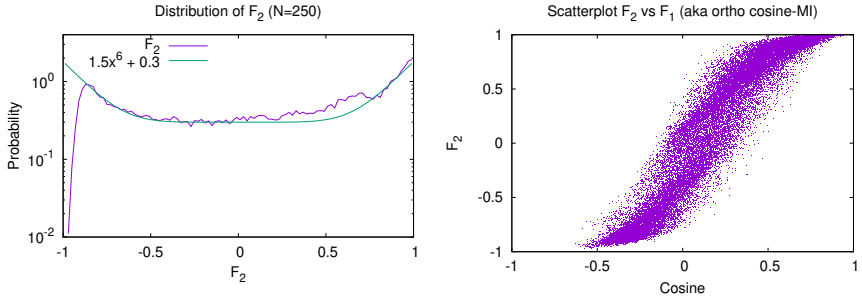
is the Heaviside step function. This provides an encoding of words as hypervectors. Is it useful for anything? How good is it?

F_2 Data

The mean and stddev for F_1 are $\mu_F = 0.09075$ and $\sigma_F = 0.3389$ – this is the mean and stddev for the top 1000 pairs (w,u) in the graphs called "ortho cosine", up above, for $N = 2500$ and $M = 1000$.

Notation: Here, $F(w,u) = F_1(w,u) = \hat{w} \cdot \hat{u}$ where $\hat{w} \cdot \hat{u}$ was called the "ortho cosine" above, and $\hat{w}$ is the normalized unit vector constructed from MI.

Below left is a graph of the actual distribution of $F_2(w,u)$ for $N(N+1)/2 = 250 \times 251/2 = 31375$ word pairs (w,u). It appears to support the claim that vectors cluster along the diagonals, as witnessed by the two peaks at +1 and -1. The green line is an eyeballed fit, given by $1.5x^6 + 0.3$.



Above right is the correlation between F_1 and F_2. It looks sinusoidal; the sine (shifted and compressed) seems to be a decent separatrix between the two lobes. I have no idea why it's correlated by that. It could have been anything, but it wasn't.

How about the top pairs? Two tables: the present F_2 on the left, and the older F_1 on the right:

Top-20		
$F_2(u,v)$	u	v
0.9997	we	they
0.9993	she	he
0.9991	And	But
0.9990	may	has
0.9985	when	where
0.9984	she	they
0.9982	when	how
0.9982	they	if
0.9981	how	who
0.9981	where	how
0.9980	shall	may
0.9980	shall	must
0.9980	when	who
0.9980	we	if
0.9979	they	he
0.9978	were	are
0.9978	night	day
0.9978	would	must
0.9977	where	who
0.9977	shall	has

Top-20 word vectors		
$\hat{w} \cdot \hat{u}$	$\vec{w}$	$\vec{u}$
0.9604	we	they
0.9525	And	But
0.9522	she	he
0.9466	they	he
0.9418	she	they
0.9412	we	he
0.9384	may	should
0.9351	when	how
0.9342	1	3
0.9322	she	we
0.9313	You	We
0.9308	people	men
0.9307	when	if
0.9263	is	was
0.9251	shall	should
0.9250	with	by
0.9245	where	how
0.9233	him	them
0.9215	had	would
0.9214	when	where

Several things to observe. First, the pairings are quite similar. Most are quite reasonable, except F_2 has the bizzaro (they, if) and (we, if) whereas F_1 has the quite reasonable (when, if). Things still look reasonable 100 or 200 words down, but mediocre at 1000 and bad at 5000, even thought $F_2 = 0.883$ down at the 5000 level. So good F_2 is very compressed at the corners of the hypercube. At the other end, near $F_2 \approx -1$, the pairings are "more than bad", they look random, randomly unrelated.

Conclusion: F_2 takes considerably more CPU resources to compute, and does not appear to offer any benefits. It's interesting, it's curious how it correlates with F_1 and has the potential of being turned into a hypervector ... but ... it does not seem to be useful in practice.

The End

This is the end of Part Eight of the diary.