Model comparisons for ENA

David Williamson Shaffer and Zachari Swiecki

October 5, 2021

1 Introduction

This document summarizes a series of conversations we have had with Yeyu, Jaeyoon, and Carl regarding methods for comparing ENA models.

2 Motivation

This problem is motivated by work in QE, and more specifically work comparing ENA models.

ENA models produce a vector for each unit in the data. Comparing whether 2 models are similar thus means comparing the distributions of two sets of vectors.

Working with high-dimensional data of this sort poses three questions:

1. Should we normalize dimensions in the high-dimensional space, and if so, how?
2. How should we measure the difference between vectors?
3. What does it mean to compare two models?

I try to address these questions in the sections that follow.

3 Normalization in ENA

3.1 Normalization within units

In ENA, we typically normalize dimensions to account for differences in the amount of coded talk.

I think of of it this way:

Consider two units u_1 and u_2, with respective ENA vectors $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$.

If the relative values of the connections are the same (ie, $\boldsymbol{\nu}_1 = \gamma\boldsymbol{\nu}_2$ for some scalar γ), then it is more or less as if u_i was just repeating the same thing: they were using the same pattern of talk, just using it more.

Therefore, if we care about u_i and u_j's patterns of discourse, we don't care about the fact that u_i uses the pattern more.

Thus, we normalize *within* each unit by computing $\mathbf{v}_i = \frac{\boldsymbol{\nu}_i}{\|\boldsymbol{\nu}_i\|}$, with the result that $\boldsymbol{\nu}_i = \gamma\boldsymbol{\nu}_j \implies \mathbf{v}_i = \mathbf{v}_j$. That is, two units with the same pattern of talk have the same normalized vector regardless of the amount of coded talk for each unit.

3.2 Normalization between units

In ENA, we *don't* normalize *between* units.

What would it mean to normalize *between* units?

Consider a case where, across all units, there is a lot of variance in the connection strengths between codes c_1 and c_2, but very little variance in the connection strengths between c_2 and c_3. That is, $var(\mathbf{v}_{*,12}) \gg var(\mathbf{v}_{*,23})$ (where $\mathbf{v}_{*,jk}$ indicates the connections between c_j and c_k in the for all u_i.

In ENA we don't standardize the dimensions so that $var(\mathbf{v}_{*,12}) = var(\mathbf{v}_{*,23})$ — that is, we don't rescale the dimensions so that they have equal variance. (This is what PCA does, eg.)

That is because:

1. The dimensions are on the same scale, so we don't need to rescale them (eg, it isn't like height and weight)
2. We don't want the model to treat the variances as equal because that will overweight differences in $\mathbf{v}_{*,23}$ — which are very small, and thus may just be noise — and thus underweight differences in $\mathbf{v}_{*,12}$, which are large.

That's all *within* a given model, though.

4 Comparing models with correlation

Here, we are are considering how to compare ENA models.[1]

4.1 Correlation of means

Given two models, $m(D)$ and $m'(D)$, applied to data set D, we could correlate (or take the cosine of the angle between) the mean vectors for each model, $\text{cor}\,(\bar{m}(D), \bar{m}'(D))$.

Correlating the mean vector of two models is appealing, because it produces a summary statistic (correlation or cosine) that reflects information about differences between the mean of the models on every dimension (that is, every code pair).

Of course, that just gives us one correlation, with no way to determine whether that difference is significant. Put another way, $m(D)$ and $m'(D)$ are different (or similar) *for this specific data set* D.

Thus, we wind up having to choose an arbitrary threshold for what it means to be "similar" or "different", and perhaps that is OK because correlations have generally understood meanings as to what is a high or low correlation.

But more important, we don't know anything about whether the models are different *in general*.

[1] Much of the argument here will follow for any models that produce a vector for each unit, but those larger implications are is beyond the scope of this memo.

4.2 Randomized models

Why might we care about whether models are different in general?

One general use case is that we want to determine if some model $m(D)$ is different from a model (perhaps the same model) applied random noise: $m'(R)$, or model m' applied to some form of randomized data, where it is possible that $m = m'$.

It is possible that $m(D)$ is similar to (or different from) $m'(R)$ — that is to one *particular* randomized data set, but that's just a chance result. Any one random model could exactly the same or completely different from $m(D)$ — or anything in between.

To account for that, we can generate a large number of the random data sets, R_i, and now we want to see whether $m(D) \in \mathbf{X}_i\left(m'(R_i)\right)$ — that is, whether $m(D)$ is in the distribution of $m'(R_i)$.

One approach is to correlate the mean of $m(D)$ with the mean of each $m'(R_i)$. That is, compute $\mathbf{X}_i\left(\text{cor}\left(\bar{m}(D), \bar{m}'(R_i)\right)\right)$.

This gives us the distribution of correlations between the mean of M(D) and the means of the random models. But we then have to determine whether the CI for the correlations is over some threshold (since the CI will never contain 1).

So this still leaves us with the threshold problem, but I think it does solve the problem of CIs for correlation/cosine. Basically, it is unlikely that the mean vector from a model will be entirely zero, and also unlikely that the correlation of two mean vectors will be 1. Possible, but probably not enough to throw off the CI too much.

4.3 Comparing units rather than means

So far so good, since we're (a) not averaging correlations and (b) using the means of a model means few zero vectors (ie, $\mathbf{v}_i = (0, 0, 0, \dots)$) or perfect correlations. Both of those things caused trouble in the past because zero vectors and perfect correlations blew up average correlations in the past[2].

But there is a bigger problem, I think. Namely, that the *units* in all of these models are the same.

This means that we're conducting under-powered and potentially flawed tests. The *mean vector* for two models may be highly correlated — eg,

$$\text{cor}\left(\bar{m}(D), \bar{m}'(D)\right) \longrightarrow 1$$

[2] To calculate meaningful averages of the correlation coefficient, the typical approach is to apply Fischer's z transformation to the coefficients, mean the values (or more often, calculate confidence intervals), and then back-convert them to the original correlation scale. When perfect correlations exist in the dataset, this can be problematic because the transformation takes 1 to infinity. This is typically fine because the average correlation value is not normally used, the transformation is used to allow the calculation of confidence intervals using the standard normal distribution and then the values are converted back–so, infinity becomes 1 again. However, when the average correlation values are of interest, we must impose yet another arbitrary threshold for correlations–e.g, transforming 1 to 0.999, 0.99, 0.9, etc. These seemingly small changes can have large effects on the mean values.

But that doesn't guarantee that the vectors for the *individual units* are correlated.

A thought experiment. As a thought experiment, consider that for any model and any data set, $\text{cor}\left(\bar{m}(D), \bar{m}(D)\right) = 1$.

Now reshuffle the vectors for the units in one model — that is, take the vectors for all units and randomly assign them *without replacement* back to the units.

The means will stay the same, but the models will clearly be different.

Problem re-statement This is actually where all this trouble started: because instead of computing the correlation of *mean vectors*, we wanted to compute the mean correlations for *the vectors of each individual unit*.

This blew up the correlation method (see footnote 2).

If we are looking at a lot of unit vectors, we can easily have a bunch that are zero. The problem is that cosine (and correlation) is *undefined* for vectors of zero length. So we are introducing an undefined quantity into the computation of our CI, and whatever value we decide to assign, it causes trouble.

So, if we compare the mean vectors, we can use correlation as described above. But if we want to retain the information that any two models represent a paired sample, then the correlation method becomes problematic.

5 Comparing models with distances

This brought us to thinking about comparing distances.

Given two models, m and m', we do not compute $\mathbf{X}_i \left(\text{cor}\left(\mathbf{v}_i^m, \mathbf{v}_i^{m'}\right)\right)$ — that is, we do not compute, for each u_i, the correlation between between its vectors in the two models. Instead, we find $\mathbf{X}_i \left(\left|\mathbf{v}_i^m - \mathbf{v}_i^{m'}\right|^*\right)$ — that is, for each u_i, the distance between its vectors in the two models (ie, between their points in the HD space). The notation $|\ldots|^*$ indicates an undetermined distance function.

Now we can find an average distance. (This is similar to finding the mean difference between points in a paired sample t test.)

However, as Zach pointed out, there will be no negative distances, so the CI can never contain zero — just as the CI for correlations can never contain 1. This means we have to set a threshold.

But we do not have an obvious way to set a meaningful threshold for distance.

5.1 Proposed solution

Here, we propose an alternative model that (a) uses mean difference between vectors for each unit in a model rather than correlation of mean vectors for each model; and (b) does not require setting an arbitrary threshold.

This solution is described for the case of determining whether some $m(D) \in \mathbf{X}_i\,(m'(R_i)$ — that is, whether one model on one data set is in the distribution of a model (perhaps the same model) on a group of randomized data sets.

Construct a mean model If $\mathbf{v}_h^i$ is the vector for unit u_h under model $m'(R_i)$, construct a mean model $M\,(\mathbf{X}_i\,(m'(R_i)) = \mathbf{V}$ such that:

$$\mathbf{V}_h = \bar{\mathbf{v}}_h^*$$

That is, $\mathbf{V}_h$ represents the mean vector for unit u_h across all of the models in $\mathbf{X}_i\,(m'(R_i)$.

Find the average distances. For each model $m'(R_i)$, find the mean difference of its units to $\mathbf{V}$. That is, compute:

$$a_i = \frac{\sum_h \left|\mathbf{v}_h^i - \mathbf{V}_h\right|^*}{max(h)}$$

This gives a distribution $\mathbf{X}_i\,(a_i)$, which is the distribution, across all models, of the average distance of the units of each model to the mean model.

Test whether $\boldsymbol{m(D) \in \mathbf{X}_i\,(m'(R_i))}$. Compute the average distance of units from $m(D)$ to $\mathbf{V}$:

$$a_t = \frac{\sum_h \left|\mathbf{v}_h^m - \mathbf{V}_h\right|^*}{max(h)}$$

This lets us compare (a) the mean difference across all units from our model to a mean model to (b) the distribution of mean differences between a set of randomized models and the mean model. We get an empirical p-value by counting the number of values in the distribution that are more extreme than a_t. If this p-value is less than our threshold, we conclude that the original model and the randomized (or simulated or different in some other way) models are significantly different.

5.2 On re-scaling

So now the question is: Which distance measure should we use?

The two logical possibilities for measuring distance in a high-dimensional space are:

1. Euclidean distance
2. Mahalanobis distance

Of course Jeff Linderoth argues that there are many such metrics we could use.

The main difference between these measures is that the Mahalanobis accounts for the correlation of dimensions by re-scaling the dimensions based on their

variance. (Technically, it multiplies distances by the inverse of the covariance matrix. But the effect is the same.) Euclidean distance does not rescale the dimensions.

So, it seems to me that we probably *don't* want to rescale the dimensions, for the same reason as above: we're measuring the distance between units, and thus we don't want to over-emphasize dimensions where there is little variance between the units.

As a result, I suggest that $|\mathbf{x}|^* = \|\mathbf{x}\|$. That is, we want to use Euclidean distance rather than Mahalanobis to measure the distance between units in the algorithm above.

6 Thoughts welcome here...

I rather suspect this is more or less precisely what Carl has been arguing for all along, so I welcome his (and others') comments and corrections.