Introduction to Large Language Models (LLMs)

Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti

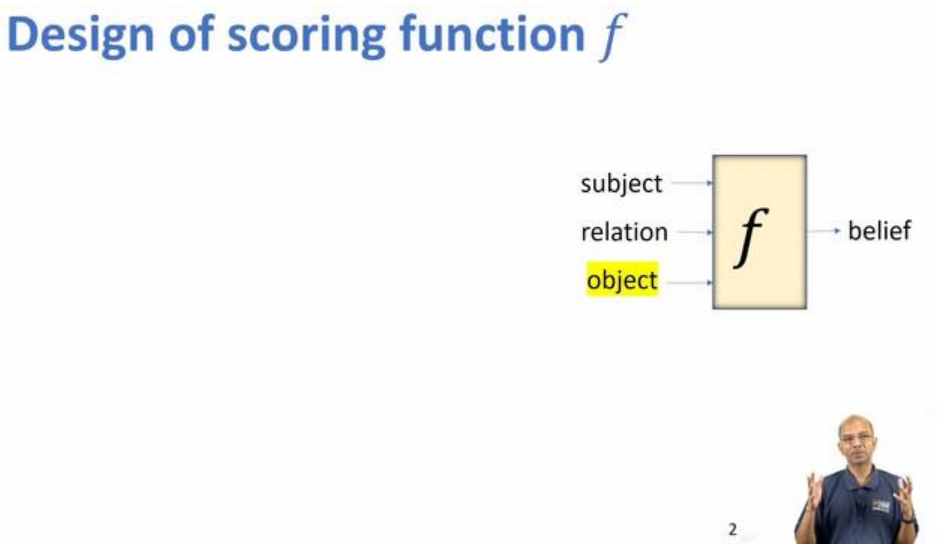
Department of Computer Science & Engineering

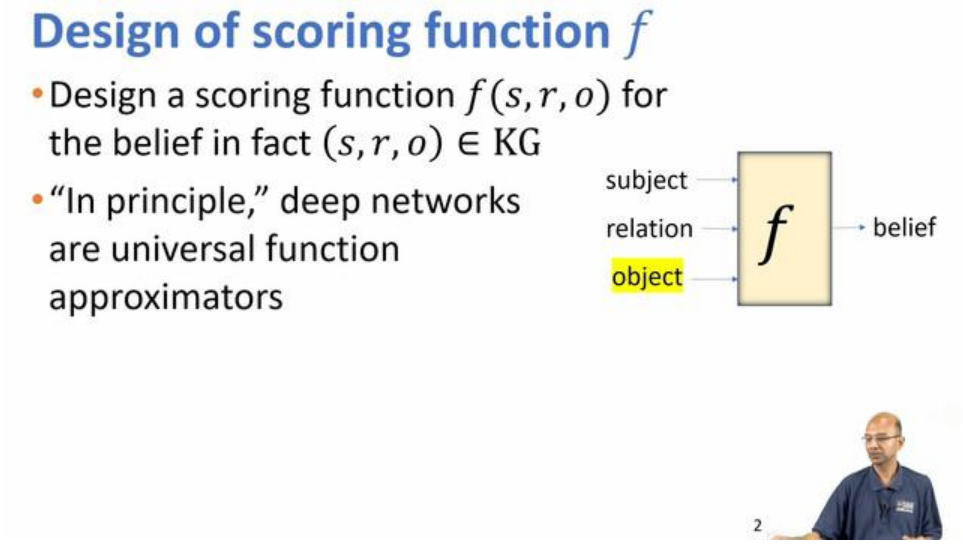
Indian Institute of Technology, Delhi

Lecture 28

Knowledge and Retrieval: Translation and Rotation Models

Next, we will focus on translation and rotation models to plug into that box that we called f recently. This is about designing the scoring function f and plugging in various forms of the subject, relation and object and then decide the shape of the function itself to output the belief. Now, in principle, deep networks are universal function approximators. If F has sufficient depth and width, it should be able to fit any function between the subject relation object and the output belief. In practice, several years and hundreds of papers have proposed different forms of F associated loss functions and sampling protocols to train the model parameters, not only inside F, but also in the representation of entities and relations.





Before we dive into it, we will quickly review word embeddings that have already been covered in this course. As you know, Word2Vec and GloVe are popular word embedding schemes which scan through an ordinary text corpus and map every token in the corpus to a dense vector embedding. It is also known that both of these are related to low rank factorizations of the document by term matrix. But what really catapulted word embeddings into scientific limelight was this property of translations as relations. It was found that the vector difference from man to king was approximately the same vector translation as between woman and queen.

Completing this parallelogram basically tells us that man to king involves a translation that means royal. And that same royalty transition takes a woman point to the queen point. Similarly, translating from a man to a woman involves a transformation which adds femaleness. That same approximate translation turns king or takes the point king to the point queen in the embedding space. And this is not only true of this specific relation, during the embedding, it was found that there are very often approximately parallel translations between two entities which are related by the same relation.

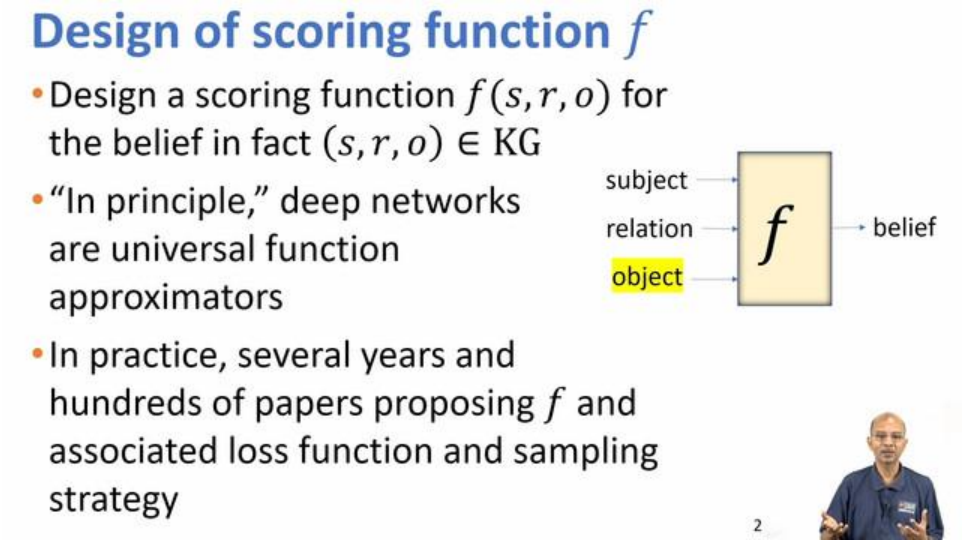
For example, here we see brown colored nodes which represent countries such as Russia and Canada and Spain, and the green colored nodes which represent cities. And the very specific relation between these countries and cities is that these cities are the capital of those countries. And surprisingly, it turns out that the translation from Spain to Madrid is approximately the same as the translation from Italy to Rome. So this suggests a very

simple way to get started on designing that function f. Suppose we start with Italy embedded in space.

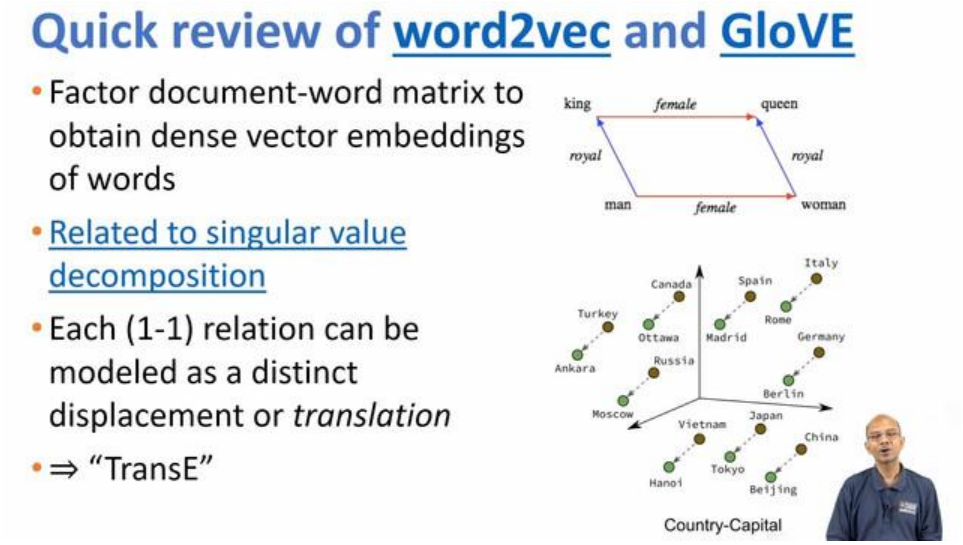
And then suppose we also have a representation of the relation has capital as a vector. So we could think of this as the subject vector. There's an origin somewhere. And then Italy is the point here in the some multidimensional space has capital is a translation, which is denoted by the vector r. And now we look for other points in the knowledge graph for our candidate objects.

So let's say here is the vector o1. We measure the distance between this tip and o1, right? And here is another vector o2. We measure the distance from the tip of that vector to o2. By the way, this tip actually has a vector name. We go from the origin to s, then we take the path r, therefore this vector is s plus r.

And here is a candidate object. So what we have to do is to measure that distance, and that distance is s. plus r minus o i for the ith candidate and we measure this distance and that is the score. So we might say f of s r o is equal to that distance. So if f is large then our belief is small and so it's a bad fact or an unpromising fact In this case, it's the opposite of what we are discussing.



If the distance is small, that means the object is very close to the point where we want it to be, and my belief is high, and it's a good triple. So with this formalism in mind, we will say that FSRO is the norm of the vector S plus R minus O. where in fact each entity and the relation will be embedded as d-dimensional real vectors. In words, the subject translated by the relation should be near the most promising object. Otherwise, we will have low confidence in the proposed fact SRO being actually in the kg.



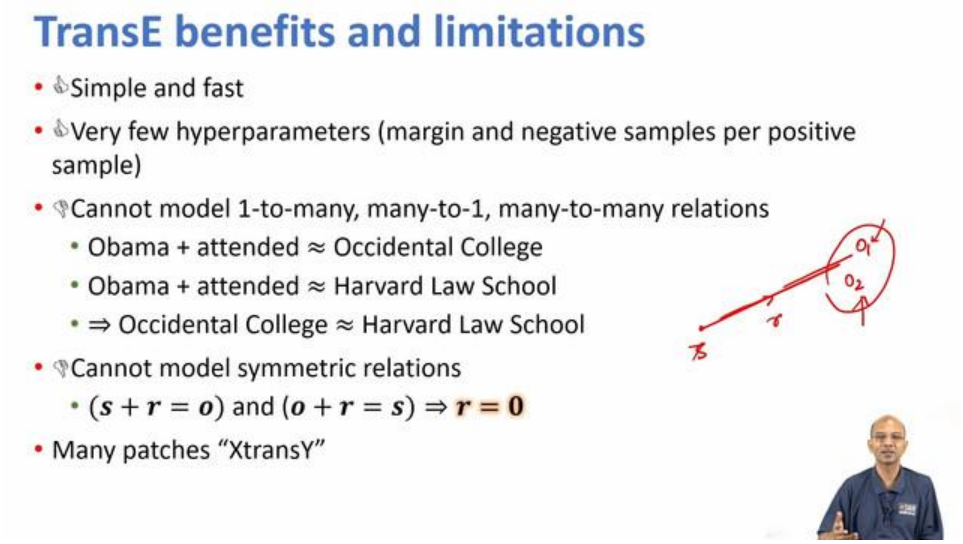
So low score is high confidence and vice versa. The loss polarity has to be then adjusted to the same hinge form. So we will take k negative samples and have the earlier hinge loss.

We want the bad scores to be now large, so we have to subtract it. And we want the good scores to be small, the good distances to be small.

So this would be the form of the hinge loss. Now observe that the function f itself has no parameters of its own. The only variables here are coming from entity embeddings, which could turn into subject or object embedding depending on the current fact context and the relation embeddings. Those are the only variables in the optimization. And that's what we will try to optimize while minimizing this kind of hinge loss over all the triples in the knowledge graph.

What are the benefits and limitations of this model, which is called TransE for translated entities? It's very simple and fast. The network is basically non-existent. It's just two linear functions with one sum and one subtraction and taking a norm. So computation is extremely fast. There are very few hyperparameters.

They're just the margin that we use and the number of negative samples per positive sample. So this is very small number of hyperparameters. The big difficulty is that Transy cannot model one-to-many, many-to-one, or many-to-many relationships. If Obama attended Occidental College and he also attended Harvard Law School, because Obama is the single subject and attended is the single relationship, what will happen is that if you attended both O1 and O2, the optimization would try to push O1 and O2 together toward the same point in space. And so we'll naturally come to believe that Occidental College and Harvard Law School are the same when they're not.

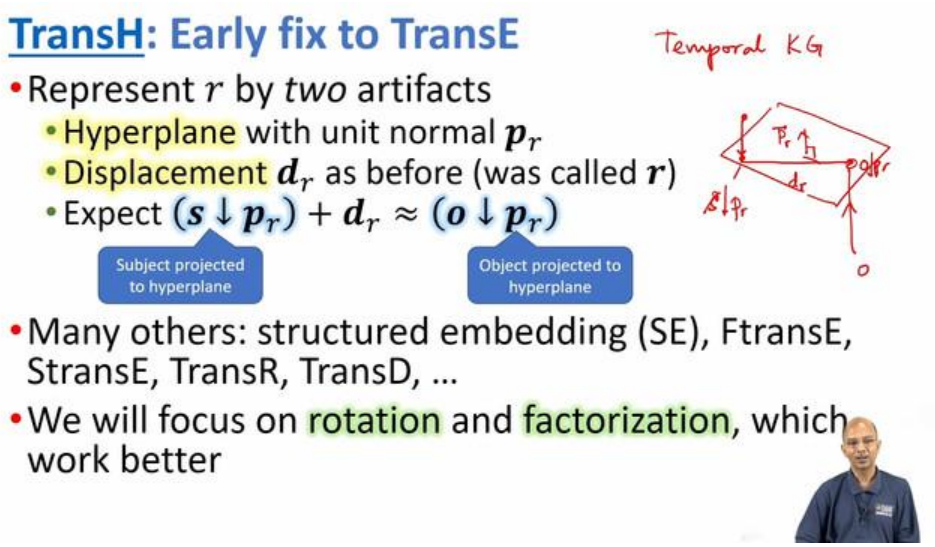


So we don't want their relationships to collapse to the same point. It also cannot model symmetric relations. If we want s plus r to be equal to o, and o plus r to be equal to s, by solving this equation simultaneously, you will see that this will hold only when r is equal to zero. So the norm of r has to come down to zero, which means all symmetric relationships, like sibling or friend, has to be represented by the same vector, which is also a limitation. Over time, many patches have been found generally named in the pattern X trans Y instead of trans E.

We might review one or two of them. TransE is one of those early fixes. And this is useful because we will come across this kind of fix later again when we discuss temporal knowledge graphs, where time will take a very interesting role because the president of US changes depending on time. So this is a natural thing to want to study. In trans H, we will represent a relation not with a single vector artifact, but with two artifacts.

First, a hyperplane in space which will be represented by its unit normal PR. So generally as a broad picture, if you're thinking of a hyperplane, it has a normal vector, which we will call PR. And the second is a displacement as before, earlier we called it just R, but now we'll have to call it DR to distinguish it from PR. So R will now be represented as both PR and DR together. What happens to our earlier law of subject plus relation equal to object? It will remain almost the same, but modified in one important way.

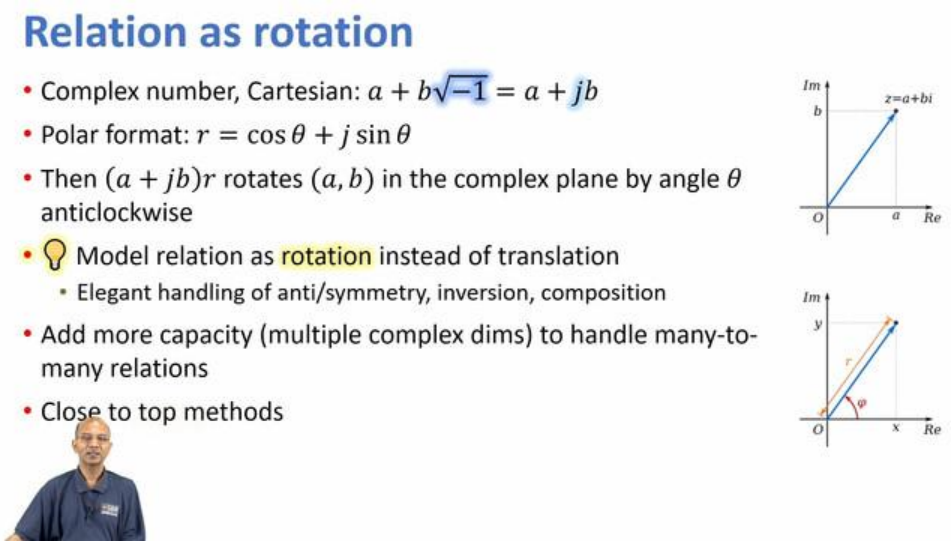
The subject, which might be here, will first be projected or dropped to the hyperplane. So the subject projected to the hyperplane plus the displacement dr within the hyperplane should roughly be close to the object projected to the hyperplane. So this is object projected to the hyperplane. That point is the subject projected to the hyperplane. Now observe that this geometry makes it possible to model many-to-many relations.



Now Barack Obama would project down to this point on the hyperplane meant for attended educational institution, and our Occidental College and Harvard Law School may be two different points which project down to a very similar place on the attended hyperplane. So this is what implements the many-to-one mapping. There have been many other attempts which are similar in all kinds of ways, but also different in some significant ways, such as structured embedding, f-transy, s-transy, and so on and so forth. Now, we won't dwell too much on this because translation models have soon been overtaken by knowledge graph techniques, which depend on rotation and factorization. So let's move on to rotation now.

As many of you may remember, in the Cartesian plane, a complex number may be represented by two real numbers, A and B. A will be on the real axis, as a component of the blue vector and b will be multiplied by square root of minus one which we call j over here and those will point to the imaginary axis and the overall vector z which will be a plus bi will represent this point. We also know that complex numbers can be represented as polar coordinates. So if the length of the vector is r and its angle from the real axis is theta, we can write it as cos theta on the real axis and sine theta multiplied by j on the imaginary

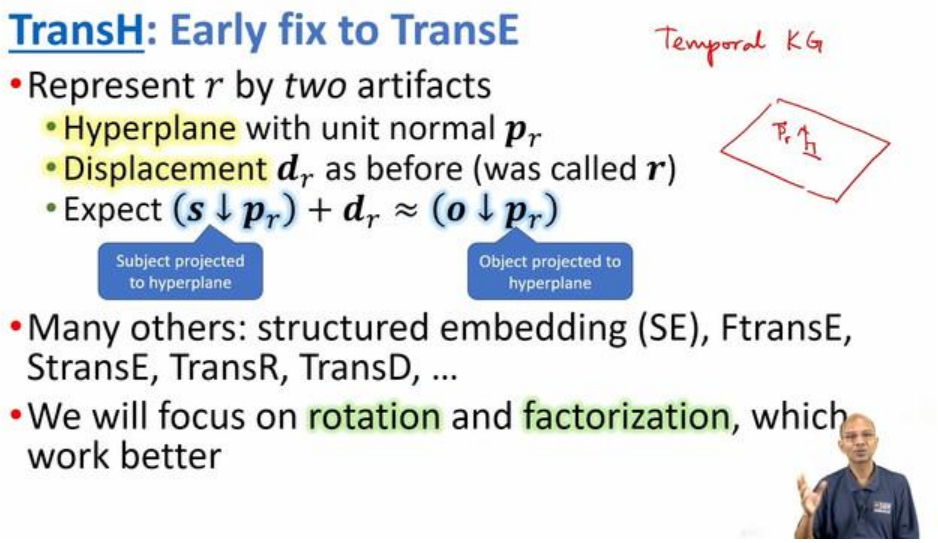
axis. We know that if we take one complex number and multiply with another complex number r that actually results in rotating the first complex number through an angle theta anticlockwise and that gives the idea of modeling a relation not as a translation but as a rotation.



So take the subject which is a complex vector and then rotate it through an angle determined by r to end up at a place which is close to an object vector, which is also a complex number. To be able to handle many-to-many relations and a large knowledge graph, we will add more capacity by not turning one complex number, but turning a whole bunch of complex numbers. So our entity now will map to an entity vector, which is not in real number to the d, but it will be in complex number to the d. So if I say that say D is equal to five, so there will be five complex numbers, each with its real part and imaginary part. So there will actually be 10 real numbers representing that entity.

Now, complex embeddings and certain interesting functions combining them give close to the best possible performance in contemporary knowledge graph completion tasks. The norm of a complex number a plus jb is simply by Pythagoras rule the square root of a squared plus b squared. If s and o are complex vectors like described recently, then the norm of the complex vector will be written as the square root of the sum of the norm of the individual complex numbers. So going back to the previous slide, we will take this as a and b and from this part calculate square root of a1 squared plus b1 squared for a1 and b1, if this is a2 and b2 we'll compute the same and then finally we will keep squaring those add

them up and take a final square root on top of them. It's easy to see that basically what we are doing is taking all the real numbers, real parts, squaring them, taking all the imaginary parts, also squaring them, adding these up and taking the square root over the whole thing.



So that is defined as the norm of a complex vector. We will insist that the complex vector corresponding to a relationship has to have norm one in all its dimensions. So when we optimize these models through gradient descent, we must ensure that every complex element of the vector r has to have unit modulus or amplitude. Just like earlier in trans E, our loss became subject plus relation, sorry, subject plus relation and minus object. And we wanted to minimize this.

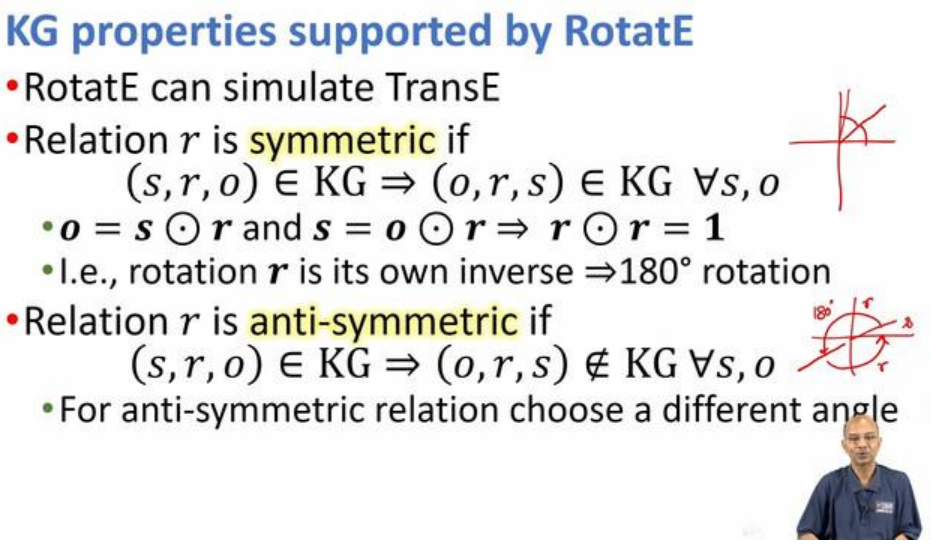
This time we'll start with the subject. We will multiply by the relation which amounts to a rotation. And now we want to get close to the object. So that would be a standard complex subtraction. And then we'll take the norm squared of that quantity.

It is easy to derive that what's written compactly in this line can be expanded out to be the sum over all dimensions of these complex vectors, the real parts of these numbers. So remember SD itself is a complex number, RD itself is a complex number and OD itself is a complex number. And so we can take the real part of that and square it, and we can take the imaginary part of that and square it, add those up, and then sum it over all the dimensions of the complex embedding vector. Now Rotatee, which is the system using

complex numbers and rotation as relation, can simulate transE in all respects. So it's a superset.

It's at least as powerful as transE. Now what happens if a relation is symmetric? When a relation is symmetric, that means if S R O is in the kg, then O R S is also in the kg. If we write this out in complex notation, it means O is equal to R complex multiplied by S and S is equal to O complex multiplied by R. If we combine these two facts, we will see that R rotated by R should be equal to one. That means the rotation R has to be its own inverse.

What does that mean? That means if I rotate something by some angle and then I rotate by another time, that should come back to the same position. The only way that can happen is if the initial rotation of the element s took it to 180 degree plus s. And so another 180 degree rotation would take it back to s. That's why it's symmetric. So r is its own inverse.



For anti-symmetric relations, which are not symmetric, we just have to choose a different angle. We can't choose 180 degree as the rotation vector. Now, note that in this case, we don't collapse all symmetric relations to the same complex vector because they may be rotating different components of the vectors. So we will still have distinct symmetric relations. If relations r and r prime are inverses of each other, it is easy to see that one corresponds to a clockwise rotation through theta, the other corresponds to an anticlockwise rotation by the same angle theta by standard complex arithmetic.

Also, compositions of two relations, like where was Barack Obama's father born, amounts to starting from Barack Obama, taking a rotation to represent the father relationship, and then taking another rotation in high dimensional space corresponding to the birthplace relationship. So there's a very simple algebraic structure of adding up angles of rotations as we compound relationships. So traversing a graph of relationships through a path of individual relations amounts to just starting with an S vector and then turning it in various ways to roam around this high dimensional space. Thank you