Introduction to Large Language Models (LLMs)

Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti

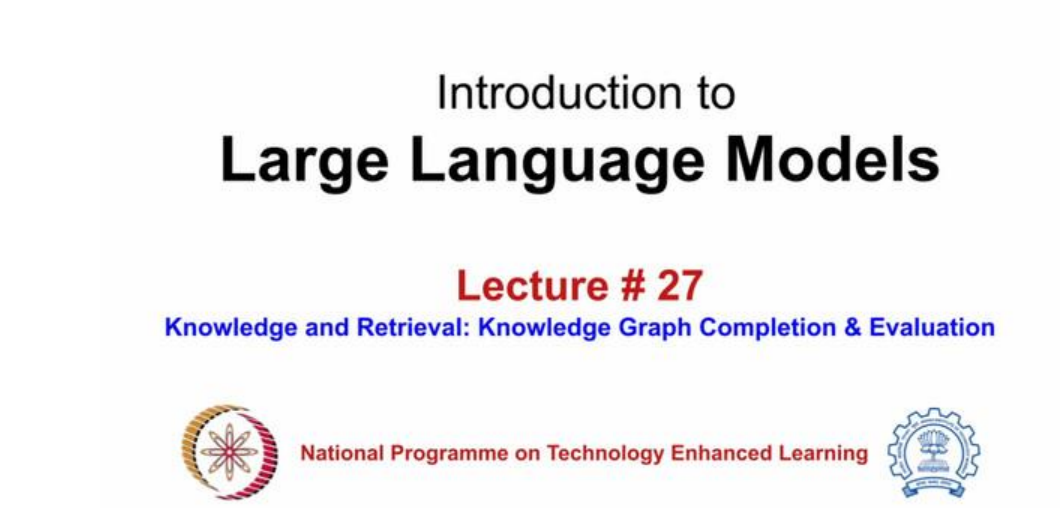
Department of Computer Science & Engineering

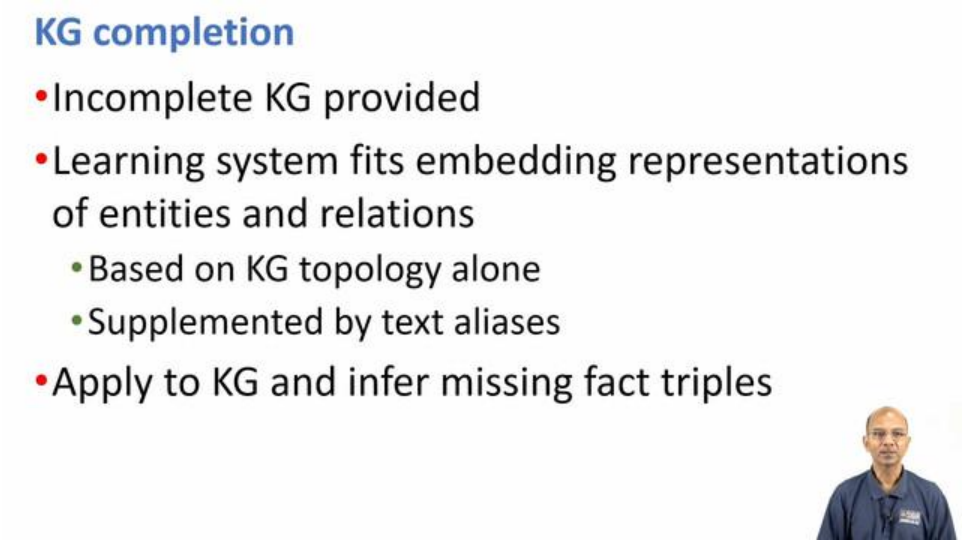
Indian Institute of Technology, Delhi

Lecture 27

Knowledge and Retrieval: Knowledge Graph Completion and Evaluation

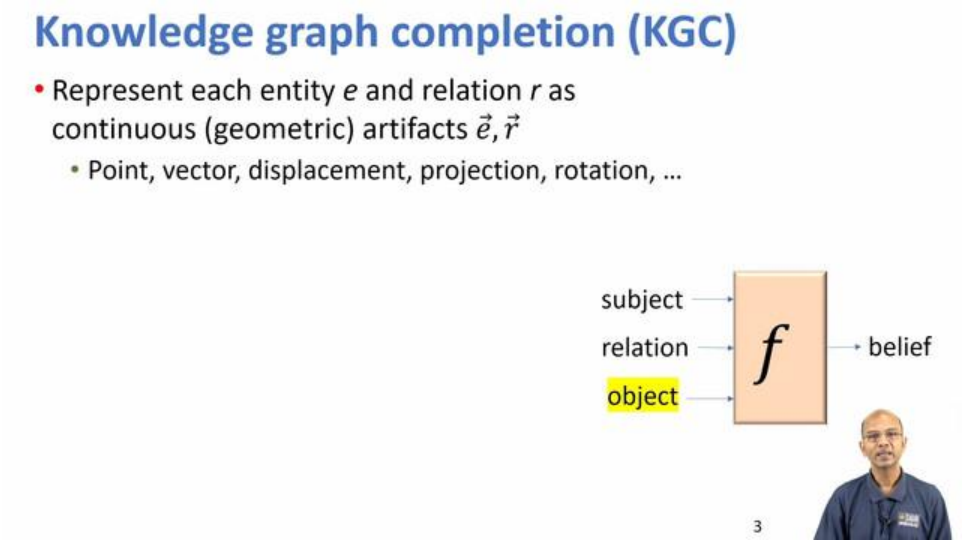
Hello, today we will continue with knowledge graph completion and its evaluation. As we mentioned recently, knowledge graphs are very incomplete and we need to complete them either as hard decisions or as soft probabilistic judgments. In this setting, an incomplete knowledge graph is provided and a modeling or machine learning system first fits embedding representations of the entities and relations in the knowledge graph. Within this paradigm, there are two possibilities. One is that these representations are based on the topology or the shape of the knowledge graph alone.





And then there is the more realistic and beneficial setting where we supplement the knowledge of the graph topology with the text aliases of the entities and relations that we briefly viewed in the last part. After these embedding representations are fitted, these will be applied back to the knowledge graph to infer missing fact triples in the knowledge graph with a certain level of confidence. So in short, knowledge graph completion consists of proposing a function f. The inputs to this function are the subject, the relation, and the object. While training, we would like to provide all the three inputs, but while testing one of these inputs may be withheld or unknown.

The output of F is the belief in the fact, which connects the subject relation and object. Now, most knowledge graph representation and completion algorithms will represent each entity, which may be a subject or an object, and each relation as some kind of continuous geometric artifacts. Such an artifact may be a point in space, some kind of high dimensional space where the KG artifacts are living, or it could be a vector or a displacement. It can be a projection or a hyperplane onto which these points are being projected. It can be a rotation.



It can be all kinds of real or complex artifacts. Without getting into the detail of that yet, the purpose of the scoring function f is to take these representations as input and output some measure of belief in subject relation object as a triple. For example, if I say Barack Obama, president of US, as the three inputs to f, f should output a high level of belief. If instead I say something like Barack Obama, discovered theory of relativity, then the output of F should be very low. We shouldn't believe in that fact based on what else we know about these entities and relationships.

During training, we will provide the identity of these subjects, relations, and objects, and we will tell the algorithm whether that fact is true or false. By the way, when I say fact in this lecture, it basically means some kind of a claim. So a claim can be either true or false. During training, I will give triples like this and their ideal belief value, be it one or be it zero. and the algorithm will fit the representations like those points or vectors or displacements or rotations.

After that, once the model is fitted, we can then infer unknown triples, which would be similar to predicting links in a social network. Some of you may be aware that in social networks, we might have a person, P1, who is already friends with some people, say P1, P2, P3, and P4, and now there may be another person, P20, and based on their behavior patterns and linkage patterns, we might ask, does the link P1 to P20 exist or should it exist? It's a bit similar to that, except that now our links are very richly endowed with features like links represent relation types and even the entities have rich features. But that's true

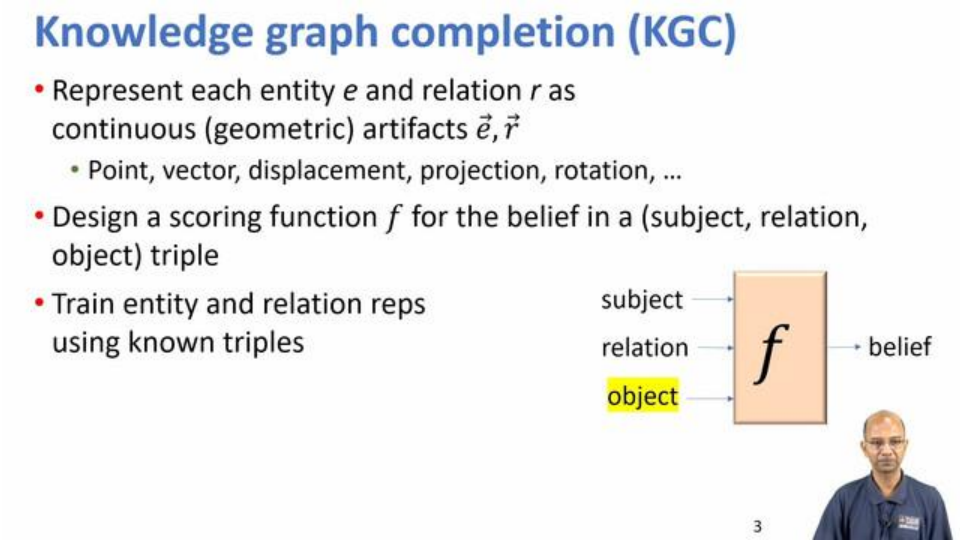
even in social network analysis. Now, in principle, Neural networks are universal function approximators. So if we plug in a sufficiently strong or high capacity neural network into the box F, it should be able to learn a function which outputs the correct belief in every proposed fact or claim.

In practice, though, the design of F is something of an art. And hundreds of papers through the last five or six years have invented and proposed mechanisms for the design of F and to fit it to observed data. Before we proceed further, let us set up some notation. We will represent entities as E in general. And when an entity takes the role of a subject, it will be written as S.

When it has the role of an object, it will be written as O. Now, why do we distinguish between subject and object in relations? That's because not all relations are symmetric. If the relationship is sibling of, then obviously what is subject and what is object doesn't matter. But if the relation is asymmetric, like parent of or CEO of a company, then we cannot switch subject and object. And so E will have to be specialized into either S or O.

When the learning system has fitted a representation for the entity E, we will write it as vector E or we'll write it as a boldface E. like this as a vector with an arrow on top or a boldface E. So subjects and objects will have corresponding embeddings bold S or bold O. And similarly, relations will have representations which are written as boldface R. Now, depending on the exact nature of the model, the exact shapes of S, O, and R might change, in some cases, there may be vectors.

In other cases, there may be matrices, and so on and so forth. Now, this function f will apply to s, r and o given as inputs and output a real number. Generally speaking, this real number may be positive or maybe even not positive. It's a raw confidence score. It's not already a probability.



To turn it into a probability, we may have to apply certain functions to it. If a particular SRO triple is known to be true in our knowledge graph, we'll call it a positive triple or a positive fact. Whereas if S prime, R prime, O prime is not in the knowledge graph, there are a couple of ways to think about it. Remember that our knowledge graph is incomplete. So if S prime, R prime, O prime does not appear in the knowledge graph, that doesn't necessarily prove that it's false.

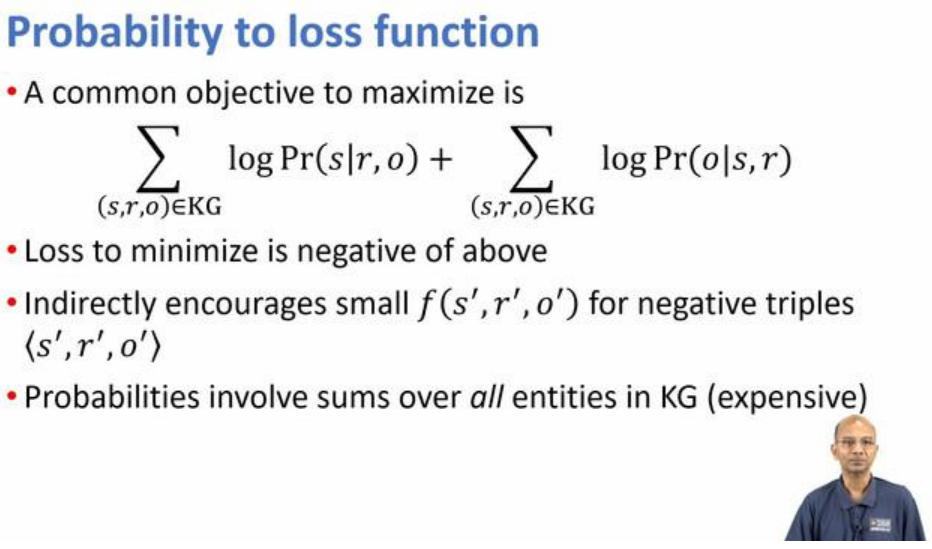
But there are ways to sample S prime, R prime, O prime, which makes it very unlikely that S prime, R prime, O prime is true. We shall visit one or two ways to do that in the upcoming slides. If S prime, R prime, O prime is known to be false, we'll call it a negative triple or a negative fact. Now, earlier I was saying that this F return value is not really a probability. If we wanted to turn it into a probability, we need some kind of a probability space.

One example of this is to fix two of the three items in the triple, like the subject and the relation, and then say what's the probability that the object is O. For example, I might say probability of Hawaii given Barack Obama and born in, and that probability should be high. Whereas if I say probability of Tokyo, given Barack Obama and birthplace, that probability should be low. And it is defined in a straightforward softmax-like fashion. We take the raw F scores, which could be positive or negative reals, and we exponentiate them.

In the numerator, we will keep e to the power fsro. And in the denominator, we will add that value over all possible values of o, which we call o prime. This normalizes the numerator and makes it add up to 1 over all candidates o. So think of a little edge r between s and an unknown o. We are trying to find out which particular o it might be.

Similarly and conversely, given the relation and the object, we might want to find out what the subject is. Now, already you see the effect of asymmetry, and that will not be tackled for the first couple of models, but we'll eventually get to it. In the form of birthplace. So if I say, what's the probability of Barack Obama given birthplace and Hawaii? Because so many people were born in Hawaii, the probability of one Barack Obama will be quite small. And so the interesting thing to observe here is that Barack Obama is now competing with everyone born in Hawaii for the fixed chunk of probability one.

And these are interesting implications for inference and this use in language models. Somehow in the literature, fixing subject and object and then trying to sample or find the probability of R has been relatively rare. I don't exactly know why. Now, if SRO is in the knowledge graph, we want the probability of O given SR to be large, and we want the probability of S given RO to be also large. How do we turn the probability to a loss function? That's fairly standard now in machine learning and deep learning.



Given a knowledge graph with known triples i would like over all the triples in the knowledge graph the log probability of s given r o we add all of these up and then we also

add the probability of o given s r take the log of that and then take the sum over that we want this entire sum to be maximized To turn that into a loss, we can negate that. So it's minus log probability of S given R O minus log probability O given S R. And then we sum over all triples in the kg and we want to minimize this negative log. So this only indirectly encourages small f of S prime R prime O prime for the negative triples. Remember that this probability has denominators.

And to make the whole thing large, I need to make the denominator for all the wrong triples small. So that's how training would work. Observe also that the probabilities in the denominator involve summing over all elements in the knowledge graph. And so that could be quite expensive. To save computational cost, we might want to sample the denominator.

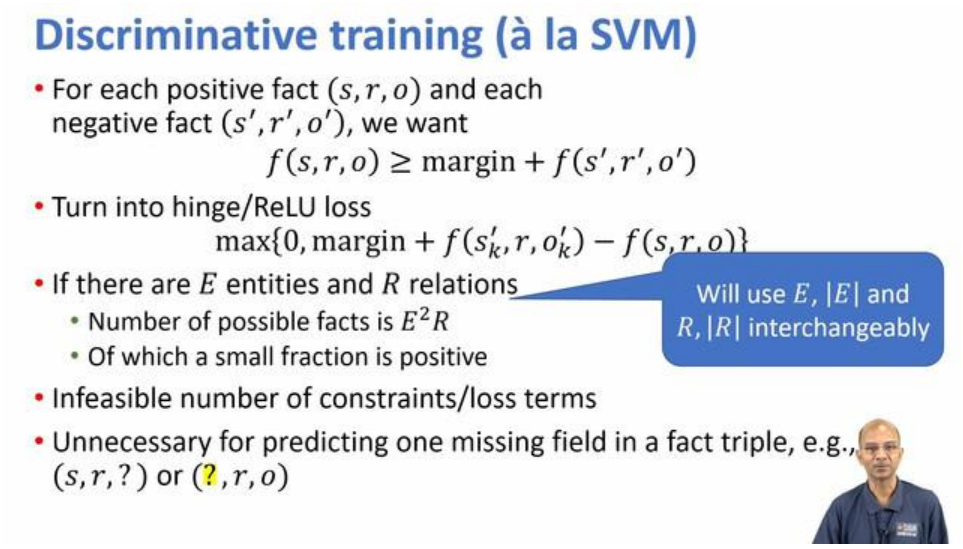
So we take a positive fact SRO, and then we replace S with a randomly sampled entity S prime. So in our running example, we say Barack Obama born in Hawaii. And this is my positive triple. Now I cancel out Barack and replace by random samples like Gandhi. So this is assumed to be positive, to be negative.

Why is that? Because there are so many people in the world that by random sampling, we would get another person born in Hawaii would be fairly small. This is called the local closed world assumption. It's not really valid given that knowledge graph is very incomplete. If we are sampling for people born in Hawaii versus elsewhere, this might succeed, but for classes which are much broader, we have to be really careful in not mistakenly sampling positive things and thinking they're negative. Even so, this helps replace the full sum in the denominator with more efficient to compute sampled estimates.

We could do uniform sampling, but that would be too expensive. So we could sample K out of the E entities uniformly at random. Suppose there are total of E entities and we take K negative samples, then our expectation of that denominator has to be adjusted for that K, right? So we have to invert that by multiplying it by, E over K or K over E, depending on which way the sampling is going. And that restores the denominator to the expected sum over all entities in the knowledge graph. Now here, one problem is that if I'm doing

the sampling of k things or entities uniformly at random, I might actually make the mistake of leaving out the gold or the true O.

So we may have to introduce it back by force into this denominator sum, and this may further bias my estimates. So those are certain standard problems that we live with in defining these efficient sampled estimates. The other approach is to not bother with probabilities at all. So instead of trying to shoehorn an expression for a probability using a softmax, we might say that, look, the positive fact has a score F of SRO and any negative fact has a score of F of S prime, R prime, O prime. And the understanding is that at least one of them is wrong.



and therefore the whole fact is wrong. So now we might say we want the good score, which is the score of the correct triple, to beat the bad score, which is the score of a triple which is wrong in at least one place, by an additive margin. So this is similar to support vector machines for those of you that are familiar with SVMs. This can be turned into what's called a hinge or a relu loss. So we say max of zero and then margin plus bad score minus good score.

If our system, if our function f and the learned weights are already perfect or close to perfect, then the good score will exceed the bad score, not by itself, but even when assisted by the margin. And therefore, this quantity over here will go negative. Because it goes negative, my overall loss taken with a maximum of zero will turn out to be zero. And

that's when we can stop the training because the system is already trained. Otherwise, if this loss turns out to be positive, we can differentiate this loss with regard to the shape of the function f and the embeddings of the entities and relations that participate in this expression.

And that starts off back propagation into the model weights. The problem again with this formulation is very similar to the summing over a large number of entities. Because there are E entities and R relations, the number of possible facts in the universe are E squared R, each of which could be true or false. Only a small fraction is positive. So that means the number of constraints between good and bad is the number of good times the number of bad.

So that's an astronomical and infeasible number. Now, generally the strategy that's followed is that I'm not going to take SRO and then perturb all three of them in all possible ways. I'm only going to perturb one thing at a time, and that limits the search space between the good bad pairs. So we would take SRO and only knock out the object, or we'll take SRO and only knock out the subject. So this leads to the formation of a batch for training a stochastic gradient based optimizer.

For each positive fact SRO in the batch, we would sample K presumed negative facts by perturbing either the subject or the object, or maybe both. And then we would accumulate the losses over these K samples. in each we'll apply the hinge loss as mentioned before where we do margin plus bad minus good. So this is the good that is the bad and we want the good to beat the bad and this whole sum to come down to zero. So this is how many knowledge graph modeling and completion algorithms are trained.

So far, we have assumed that a large score F makes the triple SRO more likely, and a small score makes it less likely. Some models may work with the opposite polarity. Some models may use a notion of distance, as we shall see. In that case, a large distance F will make SRO less likely, and a small distance will make it more likely. Usually, the loss and the optimization can be adjusted without much trouble to this flipped polarity.

How are knowledge graph completion algorithms tested or evaluated? For this, people have defined a whole bunch of knowledge graph completion tasks or challenges. The

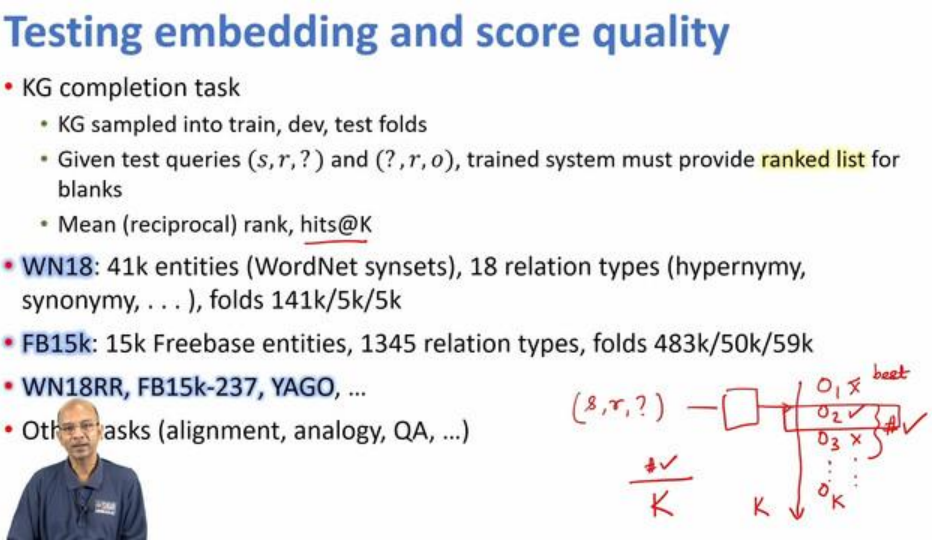
knowledge graph is first sampled into three folds, which are usually disjoint, the training fold, the dev fold, and the test fold. The default is also called the validation for it sometimes. Given a test query, eventually when the system is deployed, which may be of the form SR question mark or question mark RO, the trained system must provide a ranked list for the blanks. For example, if the test query is SR question mark, the trained system should output candidates O1, O2, O3 and so on.

And in the judgment of the system, O1 should be the best or most likely answer. On this list, we will define standard ranking measures like mean reciprocal rank or the number of hits at K. Let's take the second one first because that's easier to explain. So we go down to K elements and among the K elements, we see whether O1 is correct, O2 is correct and so on. And we count up the number of correct options and we divide it by K.

So that's the hits at K. It's like you go to Google and do a search and the top 10 or top K results are given to you. How many of them make you happy? Now, this of course makes sense if, in general, if there are multiple objects for which the triple is true. And this can happen for relations that are not one is to one. A person can be only born in one place.

So that's many to one, actually. Many people are born in a particular place. But there could be also the opposite, which is Barack Obama attended which academic institutions? And he might have attended two schools and three colleges. So it's one to many, one Barack Obama attending multiple educational institutions. So every time I get a hit, I will increment the counter and then eventually I'll divide it by K.

Now mean reciprocal rank is a bit different. So think of going down this list again until I hit the first relevant entity, which is in this example at rank two. If I do that, my reciprocal rank will be half because two was the rank at which I hit my first success. So MRR is good for situations where there is exactly one correct answer, whereas HITSAT-K is useful if there are multiple correct answers. Now, think of what happens if I hit my correct object at rank 1.



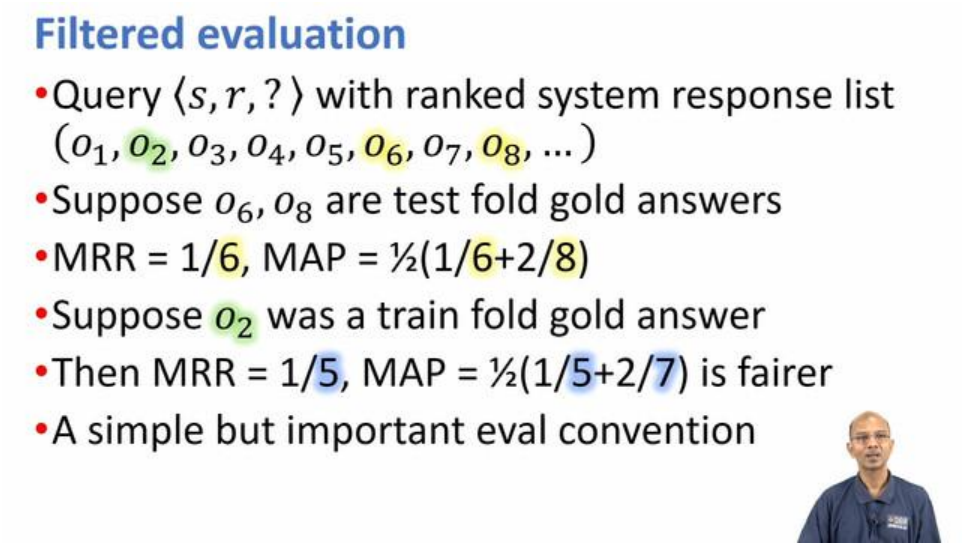
In that case, reciprocal rank will be equal to 1 over 1, which is 1, which is the maximum possible value. As the rank of the first correct answer goes down, reciprocal rank will dwindle to 0. Just like...hits at k ranges between 0 and 1, reciprocal rank will thus also range between 0 and 1. Now observe that the penalty for dropping something from rank 1 to rank 2 is to go down from 1 to half. Whereas the penalty for dropping something from 2 to infinity is to go down from half to 0. So that's sometimes found a bit extreme. For that reason, mean reciprocal rank is sometimes called the mean reciprocal rank.

It's very mean. It penalizes you as much for dropping something from 1 to 2 as dropping from 2 to infinity, where you wouldn't even see it. There are other measures of knowledge graph completion efficacy, which we won't have time to get into, but you can read about it from the references. How about datasets? There are a few very prominent datasets that are used in this area of work. There is WordNet 18, which comes from the semantic or lexical network called WordNet.

WordNet is a network where the nodes are words or synsets. So words which are synonyms of each other are collected into the same node and made into a sense set or synset, synonym set. And the edges represent Or toward relation types like hypernomy, a camel is a mammal is a living being, or synonymy that envy and jealousy are sometimes considered to be synonyms. The training and dev and test folds have sizes 141000, 5000 and 5000. And there are 18 relation types.

Another popular data set in use is FB15K, which has 15,000 freebase entities connected with 1,345 relation types, and the training dev and test folds have about 480,000 and 50 to 60,000 fact triples in them. There have been refinements on these datasets. These datasets have been found to be somewhat simple in the sense that certain algorithms can cheat. And to prevent them from doing that, modified enhanced datasets, WN18RR and FB15K-237 have been evolved over time. There are other datasets like YAGO, yet another great ontology, and several other tasks.

One important last note about evaluation is the element of filtering. So as I said, your query might be subject relation question mark, and now the ranking system has to provide a response list, O1, O2, et cetera, and so on. Suppose O6 and O8 are the designated gold correct answers. In that case, we might say that MRR is one over six, because that's the rank at which I found the first correct response. And mean average precision is the third measure might as well goal.



Mean average precision says to get my first relevant response, I have to go to rank six. When I went to rank six, I found only one relevant response up to that time. For the moment, ignore the green color on O2. If I walk down two more steps to rank 8, I see 2, namely 6 and 8, correct responses. I will now average them up.

1 sixth plus 2 eighth divided by half So that's called mean average precision. And the mean reciprocal rank is 1 in 6, as we already saw. But now, suppose that in the training set,

O2 was already known to be a correct answer. then you might say, look, it's unfair to penalize position 6 and 8 as 6 and 8 because training already revealed to us that 2 was relevant. So maybe we should knock O2 out of the list and regard the MRR as 1 out of 5 and the MAP as the average of 1 5th and 2 7th.

This is a simple modification, but it's an important evaluation convention that is not followed sufficiently uniformly across papers. So when you compare the numbers reported in papers, you have to be careful about whether they did a filtered evaluation or an unfiltered evaluation. Next, we will start looking at what kind of functions can go into that box marked F and what form the input subject relation and object vectors or embeddings can take.