DAVID SONTAG: OK, so then today's lecture is going to be about data set shift, specifically how one can be robust to data set shift. Now this is a topic that we've been alluding to throughout the semester, and the setting that I want you to be thinking about is as follows.

You're a data scientist working at, let's say, Mass General Hospital. And you've been very careful in setting up your machine learning task to make sure that the data is well specified. The labels that you're trying to predict are well specified. You train on your training data. You test it on a held out set. You see that the model generalizes well. You do chart review to make sure what you're predicting is actually what you think you're predicting. And you even do prospective deployment, where you then let your machine learning algorithm drive some clinical decision support, and you see things are working great.

Now what? What happens after this stage, when you go to deployment? What happens when your same model is going to be used not just tomorrow but also next week, the following week, the next year? What happens if your model, which is working well at this one hospital-- then there's another institution, say maybe Brigham Women's Hospital or maybe UCSF or some rural hospital in the United States wants to use the same model. Will it keep working in this short term to the future time period or in a new institution?

That's the question which we're going to be talking about in today's lecture. And we'll be talking about how one could deal with data set shift of two different varieties. The first variety is adversarial perturbations to data. The second variety is the data that changes for natural reasons. Now the reason why it's not at all obvious that your machine learning algorithm should still work in this setting is because the number one assumption we make when we do machine learning is that you're training distribution. You're training data is drawn from the same distribution as your test data. So if you now go to a setting where your data distribution has changed, even if you've computed your accuracy using your holdout data and it looks good, there's no reason that should continue to look good in this new setting where the data distribution has changed.

A simple example of what it means for data distribution to change might be as follows. Suppose that we have as input data and we're trying to predict some label which maybe meant something like y is if a patient has or will be newly diagnosed with type 2 diabetes. And this is an example which we talked about when we introduced risk stratification. You learn a

model to predict y from x.

And now, suppose you go to a new institution where their definition of what type 2 diabetes means has changed. For example, maybe they don't actually have type 2 diabetes coded in their data. Maybe they only have diabetes coded in their data, which is lumping together both type 1 and type 2 diabetes-- type 1 being what's usually a juvenile diabetes and is actually a very distinct disease from type 2 diabetes. So now the notion of what diabetes is different. Maybe the use case is also slightly different. And there's no reason, obviously, that your model which was used to predict type 2 diabetes would work for that new label.

Now this is an example of a type of data set shift which is, perhaps for you, kind of obvious. Nothing should work in this setting, right? Because here, the distribution of p of y given x changes. Meaning even if you have the same individual, your distribution py given x-- and let's say the distribution p of 0 and the distribution p of y given x in p 1, where this is one institution, this is another institution. These now are two different distributions if the meaning of the label has changed. So for the same person, there might be different distribution over what y is. So this is one type of data set shift.

And a very different type of data set shift is where we assume that these two are equal. And so that would, for example, rule out this type of data set shift. But rather what changes is p of x from location 1 to location 2, OK? And this is the type of data set shift which will be focused on in today's lecture. It goes by the name of covariant shift. And let's look at two different examples of that.

The first example would be of an adversarial perturbation. And so you've all seen the use of convolutional networks for image classification problems. This is just one illustration of such an architecture. And with such an architecture, one could then attempt to do all sorts of different object classification or image classification tasks. You could take as input this picture of a dog which is clearly a dog, right? And you could modify it just a little bit. Just add in a very small amount of noise.

What I'm going to do is I'm now going to create a new image which is that original image. Now with every single pixel, I'm going to add a very small epsilon in the direction of that noise. And what you get out is this new image, which you could stare at it however long you want. You're not going to be able to tell a difference. Basically, to the human eye, these two look exactly identical. Except when you take your machine learning classifier, which is trained on original

unperturbed data, and now apply it to this new image it's classified as an ostrich.

And this observation was published in a paper in 2014 called *Intriguing Properties of Neural Networks.* And it really kickstarted a huge surge of interest in the machine learning community on adversarial perturbations to machine learning. So asking questions, if you were to perturb inputs just a little bit, how does that change your classifiers output? And could that be used to attack machine learning algorithms? And how can one defend against it? By the way, as an aside, this is actually a very old area of research and even back in the land of linear classifiers, these questions had been studied. Although I won't get into it in this course.

So this is a type of data set shift in the sense that what we want is that this should still be classified as as a dog, right? So the actual label hasn't changed. We would like this distribution over the labels given the perturbed into it to be slightly different. Except that now, that sort of distribution of inputs is a little bit different because we're allowing for some noise to be added to each of the inputs. And in this case, the noise actually isn't random, it's adversarial. And towards the end of today's lecture, I'll give an example of how one can actually generate the adversarial image, which can change the crossfire.

Now the reason why we should care about these types of things in this course are because I expect that this type of data set shift-- which is not at all natural, it's adversarial-- is also going to start showing up in both computer vision and non computer vision problems in the medical domain. There was a nice paper by Sam Philipson, Andy Beam, and Zack Cohen recently which presented several different case studies of where these problems could really arise in healthcare.

So for example, here what we're looking at is an image classification problem arising from dermatology. You're given as input an image. For example, you would like that this image be classified as an individual having a particular type of skin disorder nevus, and this other image melanoma. And what one can see is that with a small perturbation of the input, one can completely swap the label that would be assigned to it from one to the other.

And in this paper which we're going to post as optional readings for today's course, they talk about how one could maliciously use these algorithms for benefit. So for example, imagine that a health insurance company now decides in order to reimburse for a expensive biopsy of a patient's skin, a clinician or nurse must first take a picture of the disorder and submit that picture together with the bill for the procedure. And imagine now that the insurance company

were to have a machine learning algorithm be an automatic sort of check. Was this procedure actually reasonable for this condition? And if it isn't, it might be flagged.

Now a malicious user could perturb the input such that it would, despite the patient having perhaps even completely normal looking skin, nonetheless be classified by a machine learning algorithm as being abnormal in some way, and thus perhaps could get reimbursed by that procedure. Now obviously this is an example of a nefarious setting, where we would then hope that such a individual would be caught by the police, sent to jail. But nonetheless, what we would like to be able to do is build checks and balances into the system such that that couldn't even happen, right? Because to a human, it's kind of obvious that you shouldn't be able to trick anyone with such a very minor perturbation.

So how do you build algorithms that could also be not tricked as easily as humans wouldn't be tricked?

AUDIENCE: For any of these examples, did the attacker need access to the network? And is there a way to do it if they didn't?

DAVID SONTAG: So the question is whether the attacker needs to know something about the function that's being used for classifying. There are examples of both what are called white box and black box attacks where in one setting, you have access to the function, and other settings you don't. So both have been studied in the literature, and there are results showing that one can attack in either setting.

Sometimes you might need to know a little bit more. Like for example, sometimes you need to have the ability to query the function a certain number of times. So even if you don't know exactly what the function is, if you don't know the weights of the neural network, as long as you can query it sufficiently many times you'll be able to construct adversarial examples. That would be one approach. Another approach would be maybe we don't know the function, but we know something about the training data. So there are ways to go about doing this even if you don't perfectly know the function. Does that answer your question?

So what about a natural perturbation? So this figure is just pulled from lecture 5 when we talked about non stationarity in the context of risk stratification. Just to remind you, here the x-axis is time. The y-axis is different types of laboratory test results that might be ordered. And the color denotes how many of those laboratory tests were ordered in a certain population at a point in time. So what we would expect to see if the data is was stationary is that every row

would be a homogeneous color. But instead what we see is that there are points in time-- for example, a few months interval over here-- when suddenly it looks like, for some of the laboratory tests, they were never performed. That's most likely due to a data problem or perhaps the feed of data from that laboratory test provider got lost. There were some systems problems.

But there are also going to be settings where, for example, a laboratory test is never used until it's suddenly used. And that may be because it's a new test that was just invented or approved for reimbursement at that point in time. So this is an example of non stationarity, and of course this could also result in changes in your data distribution, such as what I described over there, over time. And the third example is when you then go across institutions, when of course both the language that might be used-- think of a hospital in the United States versus a hospital in China. The clinical notes will be written in completely different languages. That would be an extreme case. And a less extreme case might be two different hospitals in Boston where the acronyms, or the shorthand that they use for some clinical terms might actually be different because of local practices.

So what do we do? This is all a setup. And for the rest of the lecture, what I'll talk about is first, very briefly, how one can build in population level checks for has something changed. And then the bulk of today's lecture we'll be talking about how to develop transfer learning algorithms and how one could think about defenses to adversarial attacks.

So before I show you that first slide for bullet one, I want to have a bit of discussion. You've suddenly done that thing of learning a machine learning algorithm in your institution, and you want to know will this algorithm work at some other institution. You pick up the phone. You call up your collaborating data scientist at that other institution. What are the questions that you should ask them in order to try and understand will your algorithm work there as well? Yeah?

AUDIENCE: What kind of lab test information they collect regularly.

DAVID SONTAG: So what type of data do they have on their patients, and do they have similar data types or features available for their patient population? Other ideas, someone who hasn't spoken in the last two lectures. Maybe someone in the far back there, people who have their computer out. Maybe you with your hand in your mouth right there. Yeah, you with the glasses on. Ideas?

AUDIENCE: Can you repeat the question?

DAVID SONTAG: You want me to repeat the question? The question was as follows. You learn your machine learning algorithm at some institution, and you want to apply it now in a new institution. What questions should you ask of that new institution to try to assess whether your algorithm will generalize to that new institution?

AUDIENCE: I guess it depends on your problem you're looking at. Are there possible differences in your population? If you're acquiring data with particular tools, what are the differences in the tools that are being used? Are their machines calibrated differently? Do they use different techniques to acquire the data?

DAVID SONTAG: All right, so let's break down each of the answers that you gave. The first answer that you gave was are there differences in the population. Someone else now, what would an example of a difference in a population? Yep.

AUDIENCE: Age distribution, where they have younger people in Boston versus central Massachusetts, for example.

DAVID SONTAG: So you may have younger people in Boston versus older people over in central Massachusetts. How might a change in age distribution affect your ability of your algorithm to generalize?

AUDIENCE: It's possible that health patterns for younger people are very different than that for older people. Perhaps some of the diseases there are more prevalent in populations that are older.

DAVID SONTAG: Thank you. So sometimes we might expect a different set of diseases to occur for a younger population versus older population, right? So type 2 diabetes, hypertension-- these are diseases that are often diagnosed when individuals are 40s, 50s, and older. If you have people who are in their 20s, you don't typically see those diseases in a younger population. And so what that means is if your model, for example, was trained on a population of very young individuals, then it might not be able to-- suppose you're doing something like predicting future cost, or so something which is not directly tied to the disease itself. The features that are predictive of future cost in a very young population might be very different from features for predictors of cost in a much older population, because of the differences in conditions that those individuals have.

Now the second answer that was given had to do with calibration of instruments. Can you elaborate a bit about that?

AUDIENCE: Yes. So I was thinking related in the colonoscopy space. So if, in that space, you're collecting videos of colons, and so you could have machines that are calibrated very differently. Let's say different light exposure, different camera settings. But you also have the GIs, and the physicians have different techniques as to how they explore the colon. So the video data itself is going to be very different.

DAVID SONTAG: So the example that was given was of colonoscopies and data that might be collected as part of that. And the data that could be collected could be different for two different reasons. One, because the actual instruments that are collecting the data, for example imaging data, might be calibrated a little bit differently. And second reason might be because the procedures that are used to perform that diagnostic test might be different in each institution. Each one will result in slightly different biases to the data, and it's not clear that an algorithm trained on one type of procedure or one type of instrument will generalize to another.

So these are all great examples. So when one reads a paper from the clinical community on developing a new risk stratification tool, what you will always see in this paper is what's known as table one. Table one looks a little bit like this. Here, I've pulled one of my own papers that was published in *JAMA Cardiology* for 2016, where we looked at how to try to find patients with heart failure who are hospitalized.

And I'm just going to walk through what this table is. So this table is describing the population that was used in the study. At the very top, it says these are characteristics of 47,000 hospitalized patients. Then what we've done is, using our domain knowledge, we know that this is a heart failure population. And we know that there are a number of different axes that differentiate patients who are hospitalized that have heart failure. And so we enumerate over many of the features that we think are critical to characterizing the population, and we give descriptive statistics on each one of those features.

You always start with things like age, gender, and race. So here, for example, the average age was 61 years old. This was, by the way, NYU Medical School. 50.8% female, 11.2% black African-American. 17.6% of individuals were on Medicaid, which was the state provided health insurance for either disabled or lower income individuals. And then we looked at quantities like what types of medications were patients on. 42% of inpatient patients were on something called beta blockers. 31.6% of outpatients were on beta blockers.

We then looked at things like laboratory test results. So one could look at the average

creatinine values, the average sodium values of the patient population, and this way describe what is the population that's being studied. Then when you go to the new institution, that new institution receives not just the algorithm, but they also receive this table one that describes the population which the algorithm was learned on. And they could use that together with some domain knowledge to think through questions like what I elicited from you in our discussion. So we could think, does it make sense that this model will generalize to this new institution? Are the reasons why it might not? And you could do that even before doing any prospective evaluation on the new population.

So almost all of you should have something like table one in your project write ups, because that's an important part of any study in this field, describing what is the population that you're doing your study on. You agree with me, Pete?

AUDIENCE: I would just add that table one, if you're doing a case control study, will have two columns that show the distributions in both populations, and then a p value of how likely those differences are to be significant. And if you leave that out, you can't get your paper published.

DAVID SONTAG: I'll just repeat Pete's answer for the recording. This table is for a predictive problem. But if you're thinking about a causal inference type problem where there is a notion of different intervention groups, then you'd be expected to report the same sorts of things but for both the case population, the people who received treatment one, and the control population, people who received treatment zero. And then you would be looking at differences between those populations as well, at the individual feature level, as part of the descriptive statistics for that study.

AUDIENCE: Just to identify [INAUDIBLE] between hospitals, is it sufficient to do t test on those tables?

DAVID SONTAG: To see if they're different. So they're always going to be different, right? You go to a new institution, it's always going to look different. And so just looking to see has something changed-- the answer is always going to be yes. But it enables a conversation, to think through. And then you might use some of the techniques that Pete's going to talk about next week on interpretability to try and understand. What is the model actually using? Then you might ask, oh, OK, the model is using this thing which makes sense in this population but might not make sense in another population. And it's these two things together that make the conversation.

Now this question has really come to the forefront in recent years in close connection to the

topic that Pete discussed last week on fairness in machine learning. You might ask if a classifier is built in some population, is it going to generalize to another population if that population that it was learned on was very biased. For example, it might have been all white people. You might ask, is that classifier going to work well in another population that might perhaps include people of different ethnicities.

And so that has led to a concept which was recently published. This working draft that I'm showing the abstract from was just a few weeks ago called *Datasheets for Datasets.* And the goal here is to standardize the process of eliciting the information, what is it about the data set that really played into your model. And so I'm going to walk you through, very briefly, just a couple of elements of what a example data set for a data sheet might look like.

This is too small for you to read, but I'll blow up one section in just a second. So this is a data sheet for a data set called studying face recognition in an unconstrained environment. So it's for a computer vision problem. There are going to be number of questionnaires which this paper that I pointed you to outlines. And you, as the model developer, go through that questionnaire and fill out the answers to it. So including things about motivation for the data set creation, composition, and so on. So in this particular instance, this data set labeled faces in the wild was created to provide images that study face recognition in unconstrained settings, where image character vision characteristics such as pose, illumination, resolution, and focus cannot be controlled. So it's intended to be real world settings.

Now one of the most interesting sections of this report that one should release with the data set has to do with how was the data pre processed or cleaned. So for example for this data set, it walks through the following process. First, raw images were obtained from the data set, and it consisted of images and captions that were found together with that image in news articles or around the web. Then there was a face detector that was run on the data set. Here were the parameters of the face detector that were used. And then remember, the goal here is to study face detection. And so one has to know how were the labels determined. And how would one, for example, eliminate if there was no face in this image. And so there, they describe how a face was detected and how a region was determined to not be a face in the case that it wasn't. And finally, it describes how duplicates were removed.

And if you think back to the examples we had earlier in the semester from medical imaging, for example, and pathology and radiology, similar data set constructions had to be done there. For example, one would go to the PAC system where radiology images are stored. One would

decide which images are going to be pulled out. One would go to radiography reports to figure out how do we extract the relevant findings from that image, which would give the labels for that learning task. And each step there will incur some bias, which one needs to describe carefully in order to understand what might the bias be of the learned classifier.

So I won't go into more detail on this now, but this will also be one of the suggested readings for today's course. It's a fast read. I encourage you to go through it to get some intuition for what are questions we might want to be asking about data sets that we create. And for the rest of the lecture today, I'm now going to move on to some more technical issues.

We're doing machine learning now. The populations might be different. What do we do about it? Can we change the learning algorithm in order to hope that your algorithm might transfer better to a new institution? Or if we get a little bit of data from that new institution, could we use that small amount of data from the new institution or a point in the future to retrain our model to do well in that slightly different distribution?

So that's the whole field of transfer learning. So you have data drawn from one distribution, on p of x and y, and maybe we have a little bit of data drawn from a different distribution, q of x, y. And under the covariance shift assumption, I'm assuming that q x comma y is equal to q of x times p of y given x, namely that the conditional distribution of y given x hasn't changed. The only thing that might have changed is your distribution over x. So that's what the covariant shift assumption would assume.

So suppose that we have some small amount of data drawn from the new distribution q. How could we then use that in order to perhaps retrain our classifier to do well for that new institution? So I'll walk through four different approaches to do so. I'll start with linear models, which are the simplest to understand. And then I'll move on to deep models.

The first approach is something that you've seen already several times in this course. We're going to think about transfer as a multi task learning problem, where one of the tasks has much less data than the other task. So if you remember when we talked about disease progression modeling, I introduced this notion of regularizing the weight vectors so that they could be close to one another. At that time, we were talking about weight vectors predicting disease progression at different time points in the future.

We can use exactly the same idea here where you take your linear classifier that was trained

on a really large corpus. I'm going to call the weights of that classifier W old. And then I'm going to solve a new optimization problem, which is minimizing over the weight w that minimizes some loss. So this is where your new training data come in. So I'm going to assume that the new training data D is drawn from the q distribution. And now I'm going to add on a regularization that asks that W should stay close to W old.

Now if D, the data from that new institution, was very large then you wouldn't need this at all. You would be able to ignore the cross fire that you learned previously and just refit everything to that new institution's data. Where something like this is particularly valuable is if there's a small amount of data set shift, and you only have a very small amount of labeled data from that new institution. Then this would allow you to change your weight vector just a little bit, right? So if this coefficient was very large, it would say that the new W can't be too far from the old W. So it would allow you to shift things a little bit in order to do well on the small amount of data that you have.

If there is a feature which was previously predictive but that feature is no longer present in the new data set, so for example it's all identically 0, then of course the new weight for that feature is going to be set to 0. And that weight you can think about as being redistributed to some of the other features. This make sense? Any questions? So this is the simplest approach to transfer learning. And before you ever try anything more complicated, I always try this.

So the second approach is also with a linear model. But here, we're no longer going to assume that the features are still useful. So when you go from your first institution, let's say MGH on the left, you learn your model and you want to apply it to some new institution, let's say UCSF on the right, it could be that there are some really big change in the feature set, such that the original features are not at all useful for the new feature set. And a really extreme example of that might be the setting that I gave earlier. You know, your model's trained on English, and you're testing it out on Chinese, right? If you used a bag of words model, that would be an example where your model obviously wouldn't generalize at all because your features are completely different.

So what would you do in that setting? What's the simplest thing that you might do? So you're taking a text classifier learned in English, and you want to apply it in a setting where the language is Chinese. What would you do? Translate, you said. There was another answer?

AUDIENCE: Train RNN.

DAVID SONTAG: Train an RNN to do what? Oh, so assume that you have some ability to do machine translation. You translate from Chinese to English. It has to be in that direction because your original classifier was trained in English. And then your new function is the composition of the translation and the original function, right? And then you can imagine doing some fine tuning if you had a small amount of data.

Now the simplest translation function might be just to use a dictionary, right? So you look up a word, and if that word has an analogy in another language, you say OK, this is the translation. But there are always going to be some words in your language which don't have a very good translation. And so you might imagine that the simplest approach would be to translate, but then to just drop out words that don't have a good analog and force your classifier to work with, let's say, just the shared vocabulary. Everything we're talking about here is an example of a manually chosen decision. So we're going to manually choose a new representation for the data, such that we have some amount of shared features between the source and target data sets.

So let's talk about electronic health record. By the way, the slides that I'll be presenting here are from a paper published in *KDD* by Jan, Tristan, your instructor Pete, and John Guttag. So you have two electronic health records, electronic health record 1, electronic health record 2. How can things change? Well, it could be that the same concept in electronic health record 1 might be mapped to a different encoding, so that's like an English to Spanish type translation, in electronic health record 2. Another example of a change might be to say that some concepts are removed. Like maybe you have laboratory test results in electronic health record 1 but not in electronic health record 2. So that's why you see an edge to nowhere. There might be new concepts. So the new institution might have new types of data that the old institution didn't have.

So what do you do in that setting? Well, one approach, we could say OK, we have some small amount of data from electronic health record 2. We could just train using that and throw away your original data from electronic health record 1. Of course, if you only have a small amount of data from the target to distribution, then that's going to be a very poor approach because you might not have enough data to actually learn a reasonable enough model.

A second obvious approach would be OK, we're going to just train on electronic health record 1 and apply it. And for those concepts that aren't present anymore, so be it. Maybe things will work very well. A third approach, which we were alluding to before when we talked about

translation, would be to learn a model just on the intersection of the two features. And what this work does is they say we're going to manually redefine the feature set in order to try to find as much common ground as possible. This is something which really involves a lot of domain knowledge, and I'm going to be using this as a point of contrast from what I'll be talking about in 10 or 15 minutes, where I talk about how one could do this without that domain knowledge that we're going to use here. OK?

So the setting that we looked at is one of predicting outcomes such as in hospital mortality or length of stay. The model which is going to be used is a bag of events model. So we will take a patient's longitudinal history up until the time of prediction. We'll look at different events that occurred. And this study was done using PhysioNet. So in Mimic, for example, events are encoded with some number. 5814 might correspond to a CVP alarm. 1046 might correspond to pain being present. 25 might correspond to the drug heparin being given, and so on. So we're going to create one feature for every event, which is encoded with some number. And we'll just say 1 if that event has occurred, 0 otherwise. So that's the representation for a patient.

Now when one goes to this new institution, EHR2, the way that events are encoded might be completely different. One won't be able to just use the original feature representation, and that's the English to Spanish example that I gave. But instead, what one could try to do is come up with a new feature set where that feature set could be derived from each of the different data sets. So for example since each one of the events in Mimic has some text description that goes with it-- event 1 corresponds to ischemic stroke. Event 2, hemorrhagic stroke, and so on. One could use that English description of the feature to come up with a way to map it into a common language. In this case, the common language is the UMLS, the United Medical Language System that Pete talked about a few lectures ago.

So we're going to now say, OK, we have a much larger feature set where we've now encoded ischemic stroke as this concept, which is actually the same ischemic stroke, but also as this concept and that concept which are more general versions of that original one, right? So this is just general stroke. And it could be multiple different types of strokes. And the hope is that even if some of these more specific ones don't show up in the new institutions' data, perhaps some of the more general concepts do show up there. And then what you're going to do is learn your model now on this expanded translated vocabulary, and then translate it. And at the new institution, you'll also be using that same common data model. And that way, one hopes

to have much more overlap in your feature set.

And so to evaluate this, the author has looked at two different time points within Mimic. One time points was when the Beth Israel Deaconess Medical Center was using electronic health record called CareVue, and the second time point was when that hospital was using a different electronic health record called MetaVision. This is an example actually of non stationarity. Now because of them using two different electronic health records, then codings were different and that's why this problem arose.

And so we're going to use this approach, and we're going to then learn a linear model on top of this new encoding that I just described. And we're going to compare the results by looking at how much performance was lost due to using this new encoding, and how well we generalize from the source task to the target task.

So here's the first question, which is how much do we lose by using this new encoding? So as a comparison point for looking at predicting in hospital mortality, we'll look at what is the predictive performance if you're to just use an existing, very simple risk score called the SAP score. And that's this red line where the y-axis here is the area under the ROC curve, and the x-axis is how much time in advance you're predicting, so the prediction gap. So using this very simple score SAP, get somewhere between 0.75 and 0.80 area under the ROC curve.

But if you were to use all of the events data, which is much, much richer than what went into that simple SAP score, you would get the purple curve, which is SAPS plus the events data, or the blue curve which is just the events data. And you can see you can get substantially better predictive performance by using that much richer feature set. The SAP score has the advantage that it's easier to generalize, because it's so simple. Those feature elements one can trivially translate to any new EHR, either manually or automatically, and thus it will always be a viable route. Whereas this blue curve, although it gets better predictive performance, you have to really worry about these generalization questions.

The same story happens in both the source task and the target task. Now the second question to ask is, well, how much do you lose when you use the new representation of the data? And so here, looking at again both EHRs, what we see first in red is the same as the blue curve I showed in the previous slide. It's using SAPS plus the item IDs, so using all of the data. And then the blue curve here, which is a bit hard to see but it's right there, is substantially lower. So that's what happens if you now use this new representation. And you see that you do lose

something by trying to find a common vocabulary. The performance does get hit a bit.

But what's particularly interesting is when you attempt to generalize, you start to see a swap. So now the colors are going to be quite similar. Red here was at the very top before, so red is using the original representation of the data. Before, it was at the very top. Shown here is the training error on this institution, CareVue. You see there's so much rich information in the original feature set that it's able to do very good predictive performance. But once you attempt to translate it-- so you train on CareVue, but you test on MetaVision-- then the test performance shown here by this solid red line is actually the worst of all of the systems. There's a substantial drop in performance because not all of these features are present in the new EHR.

On the other hand, the translated version-- despite the fact that it's a little bit worse when evaluated on the source-- it generalizes much better. And so you see a significantly better performance that's shown by this blue curve here when you use this translated vocabulary. There's a question.

AUDIENCE: So when you train with more features, how do you apply the model if not all the features are there?

DAVID SONTAG: So you assume that you have come up with a mapping from the features in both of the EHRs to this common feature vocabulary of QEs. And the way that this mapping is going to be done in this paper is based on the text of the events. So you take the text based description of the event, and you come up with a deterministic mapping to this new UMLS-based representation. And then that's what's being used. There's no fine tuning being done in this particular example.

So I consider this to be a very naive application of transfer. The results are exactly what you would expect the results to be. And obviously, a lot of work had to go into doing this. And there's a bit of creativity in thinking that you should use the English based description of the features to come up with the automatic mapping, but sort of the story ends there.

And so a question which all of you might have is how could you try to do such an approach automatically? How could we automatically find new representations of the data that are likely to generalize from, let's say, a source distribution to a target distribution? And so to talk about that, we're going to now start thinking through representation learning based approaches, of which deep models are particularly capable. So the simplest approach to try to transfer

learning in the context of, let's say, deep neural networks would be to just chop off part of the network and reuse some internal representation of the data in this new location.

So the picture looks a little bit like this. So the data might feed in the bottom. There might be a number of convolutional layers, some fully connected layers. And what you decide to do is you're going to take this model that's trained in one institution, you chop it at some layer. It might be, for example, prior to the last fully connected layer. And then you're going to take the new representation of your data.

Now the representation of the data is what you would get out after doing some convolutions followed by a single fully connected layer. And then you're going to take your target distribution data, which you might only have a small amount of, and you learn a simple model on top of that new representation. So for example, you might learn a shallow classifier using a support vector machine on top of that new representation. Or you might add in a couple more layers of a deep neural network and then fine tune the whole thing end to end.

So all of these have been tried. And in some cases, one works better than another. And we saw already one example of this notion in this course, and that was when Adam Yalla spoke in lecture 13 about breast cancer and mammography. Where in his approach, he said that he had tried both taking a randomly initialized classifier and comparing that to what would happen if you initialized with a well known image net based deep neural network for the problem.

And he had a really interesting story that he gave. In his case, he had enough data that he actually didn't need to initialize using this pre trained model from ImageNet. If he had just done a random initialization, eventually-- and this x-axis, I can't remember. Maybe it might be hours of training or epochs. I don't remember. It's time. Eventually, the random initialization gets to a very similar performance. But for his particular case, if you were to do a initialization with ImageNet and then fine tune, you get there much, much quicker. And so it was for the computational reason that he found it to be useful.

But in many other applications in medical imaging, these same tricks become essential because you just don't have enough data in the new test case. And so one makes use of, for example, the filters which one learns from an ImageNet task which is dramatically different from the medical imaging problems. And then using those same filters together with sort of a new set of top layers in order to fine tune it for the problem that you care about.

So this would be the simplest way to try to hope for a common representation for transfer in a

deep architecture. But you might ask, how would you do the same sort of thing with temporal data, not image data? Maybe data that's from a language or data from timed series of health insurance claims. And for that, you really want to be thinking about recurrent neural networks.

So just remind you, recurrent neural network is a recurrent architecture where you take as input some vector. For example, if you're doing language modeling, that vector might be encoding just a one hot encoding of what is the word of that location. So for example, this vector might be all zeros except for the fourth dimension, which is 1 denoting that this word is the word "class." And then it's fed into a recurrent unit which takes previous hidden states, combines it with the current input, and gives you a new hidden state. And in this way, you read in, you encode the full input. And then you might predict, make a classification based on the hidden state of the last timestamp. That would be a common approach.

And here would be a very simple example of a recurrent unit. Here I'm using s to denote the hidden state. Often we see h used to denote the hidden state. This is a particularly simple example where there's just a single non linearity. So you take your previous hidden state, you hit it with some matrix Wss, and you add that to the input being hit by a different matrix. You now have a combination of the input plus the previous hidden state. You apply non linearity to that, and you get your new hidden state out. So that would be an example of a typical recurrent unit, a very simple recurrent unit.

Now the reason why I'm going through these details is to point out that the dimension of that Wsx matrix is the dimension of the hidden state, so the dimension of s, by the vocabulary size if you're using a one hot encoding of the input. So if you have a huge vocabulary, that matrix Wss is also going to be equally large. And the challenge that presents is that it would lead to over fitting on rare words very quickly. And so that's a problem that could be addressed by instead using a low rank representation of that Wsx matrix.

In particular, you could think about introducing a lower dimensional bottleneck which, in this picture, I'm noting as xt prime-- which is your original xt input, which is the one hot encoding, multiplied by a new matrix We. And then your recurrent unit only takes inputs of that xt prime's dimension, which is k, which might be dramatically smaller than v. And you can even think about each column of that intermediate representation We as a word embedding. And this is something that Pete talked to quite a bit about when we were talking my natural language processing. And many of you would have heard about in the context of things like word to vec.

So if one wanted to take a setting, for example, one institution's data where you had a huge amount of data, learn a recurrent neural network on that institution's data, and then generalize it to a new institution, one way of trying to do that-- if you think about it, what is the thing that you chop? One answer might be all you do is keep the word embedding. So you might say, OK, I'm going to keep the We's. I'm going to translate that to my new institution. But I'm going to let the recurrent parameters-- for example, that Wss, you might allow to be relearned for each new institution. And so that might be one approach of how to use the same idea that we had from feed forward neural networks within a recurrent setting.

Now all of this is very general. And what I want to do next is to instantiate it a bit in the context of health care. So since the time that Pete presented the extensions of where to vec, such as Bert and Elmo-- and I'm not going to go into them now, but if you go back to Pete's lecture from a few weeks ago to remind yourselves what those were-- since the time you presented that lecture, there are actually three new papers that tried to apply this in the health care context, one of which was from MIT.

And so these papers all have the same sort of idea. They're going to take some data set. And these papers all use Mimic. They're going to take that text data. They're going to learn some word embeddings or some low dimensional representations of all words in the vocabulary. In this case, they're not learning a static representation for each word. Instead, these Burt and Elmo approaches are going to be learning what you can think of as dynamic representations. They're going to be a function of the word and their context on the left and right hand sides.

What they'll do is then take those representations and attempt to use them for a completely new task. Those new tasks might be on Mimic data. So for example, these two tasks are classification problems on Mimic. But they might also be on non Mimic data. So these two tasks are from classification problems on clinical tasks that didn't even come from Mimic at all. So it's really an example of translating what you learn from one institution to another institution.

These two data sets were super small. Actually, all of these data sets were really, really small compared to the original size of Mimic. So there might be some hope that one could learn something that really improves generalization. And indeed, that's what plays out. So all these tasks are looking at a concept detection task. Given a clinical note, identify the segments of text within a note that refer to, for example, a disorder or a treatment or something else, which you then, in a second stage, might normalize to the ULMS.

So what's really striking about these results is what happens when you go from the left to the right column, which I'll explain in a second, and what happens when you go top to bottom across each one of these different tasks. So the left column are the results. And these results are an f score. The results, if you were to use embeddings trained on a non-clinical data set-- or definitely not on Mimic, but on some other more general data set. The second column is what would happen if you trained those embeddings on a clinical data set, in this case Mimic. And you see pretty big improvements from the general embeddings to the Mimic based embeddings.

What's even more striking is the improvements that happen as you get better and better embeddings. So the first row are the results if you were to use just word to vec embeddings. And so for example, for the ITB2 challenge in 2010, you get 82.65 f score using word to vec embeddings. And if you use a very large Burt embedding, you get 90.25 f score, f measure, which is substantially higher. And the same findings were found time and time again across different tasks.

Now what I find really striking about these results is that I had tried many of these things a couple of years ago, not using Bert or Elmo but using word to vec, glove, and fast text. And what I found is that using word embedding approaches for these problems-- even if you threw that in as additional features on top of other state of the art approaches to this concept extraction problem-- did not improve predictive performance above the existing state of the art.

However in this paper, here they used the simplest possible algorithm. They used a recurrent neural network fed into a conditional random field for the purpose of classifying each word into each of these categories. And the features that they used are just these embedding features. So with just the word to vec embedding features, the performance is crap. You don't get anywhere close to the state of art. But with the better embeddings, they actually improved on the state of the art for every single one of these tasks. And that is without any of the manual feature engineering.