Addressing data mismatch

(SPEECH)
If your training set comes from a different distribution, than your dev and test set, and if error analysis shows you that you have a data mismatch problem, what can you do?

There are completely systematic solutions to this, but let's look at some things you could try.

If I find that I have a large data mismatch problem, what I usually do is carry out manual error analysis and try to understand the differences between the training set and the dev/test sets.

To avoid overfitting the test set, technically for error analysis, you should manually only look at a dev set and not at the test set.

But as a concrete example, if you're building the speech-activated rear-view mirror application, you might look or, I guess if it's speech, listen to examples in your dev set to try to figure out how your dev set is different than your training set.

So, for example, you might find that a lot of dev set examples are very noisy and there's a lot of car noise.

And this is one way that your dev set differs from your training set.

And

(DESCRIPTION)
The phrase big, noisy, dash, car noise are hand written on the slide.

(SPEECH)
maybe you find other categories of errors.

For example, in the speech-activated rear-view mirror in your car, you might find that it's often mis-recognizing street numbers because there are a lot more navigational queries which will have street address.

So, getting street numbers right is really important.

When

(DESCRIPTION)
The phrase street numbers is hand written on the slide.

(SPEECH)
you have insight into the nature of the dev set errors, or you have insight into how the dev set may be different or harder than your training set, what you can do is then try to find ways to make the training data more similar.

Or, alternatively, try to collect more data similar to your dev and test sets.

So, for example, if you find that car noise in the background is a major source of error, one thing you could do is simulate noisy in-car data.

(DESCRIPTION)
The earlier note for noisy and car noise is underlined.

(SPEECH)
So a little bit more about how to do this on the next slide.

Or you find that you're having a hard time recognizing street numbers, maybe you can go and deliberately try to get more data of people speaking out numbers and add that to your training set.

Now, I realize that this slide is giving a rough guideline for things you could try.

This isn't a systematic process and, I guess, it's no guarantee that you get the insights you need to make progress.

But I have found that this manual insight, together we're trying to make the data more similar on the dimensions that matter that this often helps on a lot of the problems.

So, if your goal is to make the training data more similar to your dev set, what are some things you can do?

One of the techniques you can use is artificial data synthesis and let's discuss that in the context of addressing the car noise

(DESCRIPTION)
New slide, Artificial data synthesis.

(SPEECH)
problem.

So, to build a speech recognition system, maybe you don't have a lot of audio that was actually recorded inside the car with the background noise of a car, background noise of a highway, and so on.

But, it turns out, there's a way to synthesize it.

So, let's say that you've recorded a large amount of clean audio without this car background noise.

So, here's an example of a clip you might have

(DESCRIPTION)
Audio with the phrase, The quick brown fox jumps over the lazy dog, forms the first part of the equation. It is followed by a plus, for car noise. Finally, equals, synthesized in-car audio.

(SPEECH)
in your training set.

By the way, this sentence is used a lot in AI for testing because this is a short sentence that contains every alphabet from A to Z, so you see this sentence a lot.

But, given that recording of "the quick brown fox jumps over the lazy dog," you can then also get a recording of car noise like this.

So, that's what the inside of a car sounds like, if you're driving in silence.

And if you take these two audio clips and add them together, you can then synthesize what saying "the quick brown fox jumps over the lazy dog" would sound like, if you were saying that in a noisy car. So, it sounds like this.

So, this is a relatively simple audio synthesis example.

In practice, you might synthesize other audio effects like reverberation which is the sound of your voice bouncing off the walls of the car and so on.

But through artificial data synthesis, you might be able to quickly create more data that sounds like it was recorded inside the car without needing to go out there and collect tons of data, maybe thousands or tens of

thousands of hours of data in a car that's actually driving along.

So, if your error analysis shows you that you should try to make your data sound more like it was recorded inside the car, then this could be a reasonable process for synthesizing that type of data to give you a learning algorithm.

Now, there is one note of caution I want to sound on artificial data synthesis which is that, let's say, you have 10,000 hours of data that was recorded against a quiet background.

And, let's say, that you have just one hour of car noise.

So, one thing you could try is take this one hour of car noise and repeat it 10,000 times in order to add to this 10,000 hours of data recorded against a quiet background.

If you do that, the audio will sound perfectly fine to the human ear, but there is a chance, there is a risk that your learning algorithm will over fit to the one hour of car

(DESCRIPTION)
This phrase is hand-written under car noise. A circle is drawn under synthesized in-car audio, and he begins to write nearby.

(SPEECH)
noise.

And, in particular, if this is the set of all audio that you could record in the car or, maybe the sets of all car noise backgrounds you can imagine, if you have just one hour of car noise background, you might be simulating just a very small subset of this space.

You might be just synthesizing from a very small subset of this space.

(DESCRIPTION)
A small circle is drawn within the larger circle, and labeled as synthesis.

(SPEECH)
And to the human ear, all these audio sounds just fine because one hour of car noise sounds just like any other hour of car noise to the human ear.

But, it's possible that you're synthesizing data from a very small subset of this space, and the neural network might be overfitting to the one hour of car noise that you may have.

I don't know if it will be practically feasible to inexpensively collect 10,000 hours of car noise so that you don't need to repeat the same one hour of car noise over and over but you have 10,000 unique hours of car noise to add to 10,000 hours of unique audio recording against a clean background.

But it's possible, no guarantees.

But it is possible that using 10,000 hours of unique car noise rather than just one hour, that could result in better performance through learning algorithm.

And the challenge with artificial data synthesis is to the human ear, as far as your ears can tell, these 10,000 hours all sound the same as this one hour, so you might end up creating this very impoverished synthesized data set from a much smaller subset of the space without actually realizing it.

Here's

(DESCRIPTION)
The number 10,000 hours is written under the car noise portion, and an arrow drawn to the quick brown fox phrase on the left.

(SPEECH)
another example of artificial data synthesis.

(DESCRIPTION)
New slide, Artificial data synthesis. Car recognition is the title above a photo of a blue Acura coupe.

(SPEECH)
Let's say you're building a self driving car and so you want to really detect vehicles like this and put a bounding box around it let's say.

So, one

(DESCRIPTION)
A red box is drawn around the blue car.

(SPEECH)
idea that a lot of people have discussed is, well, why should you use computer graphics to simulate tons of images of cars?

And, in fact, here are a couple of pictures of cars that were generated using computer graphics.

And I think these graphics effects are actually pretty good and I can imagine that by synthesizing pictures like these, you could train a pretty good computer vision system

(DESCRIPTION)
Two more photos of cars appear. The middle one depicts a racing stock car with colorful camoflauge. The photo on the left depicts a white , two door luxury sedan.

(SPEECH)
for detecting cars.

Unfortunately, the picture that I drew on the previous slide again applies in this setting.

Maybe this is the set of all cars and, if you synthesize just a very small subset of these cars, then to the human eye, maybe the synthesized images look fine.

But you might overfit to this small subset you're synthesizing.

In particular, one idea that a lot of people have independently raised is, once you find a video game with good computer graphics of cars and just grab images from them and get a huge data set of pictures of cars, it turns out that if you look at a video game, if the video game has just 20 unique cars in the video game, then the video game looks fine because you're driving around in the video game and you see these 20 other cars and it looks like a pretty realistic simulation.

But the world has a lot more than 20 unique designs of cars, and if your entire synthesized training set has only 20 distinct cars, then your neural network will probably overfit to these 20 cars.

And it's difficult for a person to easily tell that, even though these images look realistic, you're really covering such a tiny subset of the sets of all possible cars.

So, to summarize, if you think you have a data mismatch problem, I recommend you do error analysis, or look at the training set, or look at the dev set to try this figure out, to try to gain insight into how these two distributions

of data might differ.

And then see if you can find some ways to get more training data that looks a bit more like your dev set.

One of the ways we talked about is artificial data synthesis.

And artificial data synthesis does work.

In speech recognition, I've seen artificial data synthesis significantly boost the performance of what were already very good speech recognition system.

So, it can work very well.

But, if you're using artificial data synthesis, just be cautious and bear in mind whether or not you might be accidentally simulating data only from a tiny subset of the space of all possible examples.

So, that's it for how to deal with data mismatch.

Next, I like to share with you some thoughts on how to learn from multiple types of data at the same time.