Training and testing on different distributions

(DESCRIPTION)
Text, Mismatched training and dev, test data. Training and testing on different distributions.

(SPEECH)
Deep learning algorithms have a huge hunger for training data.

They just often work best when you can find enough label training data to put into the training set.

This has resulted in many teams sometimes taking whatever data you can find and just shoving it into the training set just to get it more training data.

Even if some of this data, or even maybe a lot of this data, doesn't come from the same distribution as your dev and test data.

So in a deep learning era, more and more teams are now training on data that comes from a different distribution than your dev and test sets.

And there's some subtleties and some best practices for dealing with when you're training and test distributions differ from each other.

Let's take a look.

(DESCRIPTION)
New slide, Cat app example. Two columns hold three pictures a piece of cats. The first set of three pictures is marked, Data from webpages. The second set of three is marked, Data from mobile app.

(SPEECH)
Let's say that you're building a mobile app where users will upload pictures taken from their cell phones, and you want to recognize whether the pictures that your users upload from the mobile app is a cat or not.

So you can now get two sources of data.

One which is the distribution of data you really care about, this data from a mobile app like that on the right, which tends to be less professionally shot, less well framed, maybe even blurrier because it's shot by amateur users.

The other source of data you can get is you can crawl the web and just download a lot of, for the sake of this example, let's say you can download a lot of very professionally framed, high resolution, professionally taken images of cats.

And let's say you don't have a lot of users yet for your mobile app.

So maybe you've gotten 10,000 pictures uploaded from the mobile app.

But by crawling the web you can download huge numbers of cat pictures, and maybe you have 200,000 pictures of cats downloaded off the Internet.

So what you really care about is that your final system does well on the mobile app distribution of images, right?

(DESCRIPTION)
The phrase, care about this, is marked above the Data from mobile app group.

(SPEECH)
Because in the end, your users will be uploading pictures like those on the right and you need your classifier to do well on that.

But you now have a bit of a dilemma because you have a relatively small dataset, just 10,000 examples drawn from that distribution.

And you have a much bigger dataset that's drawn from a different distribution.

There's a different appearance of image than the one you actually want.

So you don't want to use just those 10,000 images because it ends up giving you a relatively small training set.

And using those 200,000 images seems helpful, but the dilemma is this 200,000 images isn't from exactly the distribution you want.

So what can you do?

Well, here's one option.

(DESCRIPTION)
A line is drawn for option one.

(SPEECH)
One thing you can do is put both of these data sets together so you now have 210,000 images.

And you can then take the 210,000 images and randomly shuffle them into a train, dev, and test set.

And let's say for the sake of argument that you've decided that your dev and test sets will be 2,500 examples each.

So your training set will be 205,000 examples.

Now so set up your data this way has some advantages but also disadvantages.

The advantage is that now you're training, dev and test sets will all come from the same distribution, so that makes it easier to manage.

But the disadvantage, and this is a huge disadvantage, is that if you look at your dev set, of these 2,500 examples, a lot of it will come from the web page distribution of images, rather than what you actually care about, which is the mobile app distribution of images.

So it turns out that of your total amount of data, 200,000, so I'll just abbreviate that 200k, out of 210,000, we'll write that as 210k, that comes from web pages.

So all of these 2,500 examples on expectation, I think 2,381 of them will come from web pages.

This is on expectation, the exact number will vary around depending on how the random shuttle operation went.

But on average, only 119 will come from mobile app uploads.

So remember that setting up your dev set is telling your team where to aim the target.

And the way you're aiming your target, you're saying spend most of the time optimizing for the web page distribution of images, which is really not what you want.

(DESCRIPTION)
A red X is drawn next to option one.

(SPEECH)
So I would recommend against option one, because this is setting up the dev set to tell your team to optimize for a different distribution of data than what you actually care about.

(DESCRIPTION)
A line is drawn for option two.

(SPEECH)
So instead of doing this, I would recommend that you instead take another option, which is the following.

The training set, let's say it's still 205,000 images, I would have the training set have all 200,000 images from the web.

And then you can, if you want, add in 5,000 images from the mobile app.

And then for your dev and test sets, I guess my data sets size aren't drawn to scale.

Your dev and test sets would be all mobile app images.

So the training set will include 200,000 images from the web and 5,000 from the mobile app.

The dev set will be 2,500 images from the mobile app, and the test set will be 2,500 images also from the mobile app.

The advantage of this way of splitting up your data into train, dev, and test, is that you're now aiming the target where you want it to be.

You're telling your team, my dev set has data uploaded from the mobile app and that's the distribution of images you really care about, so let's try to build a machine learning system that does really well on the mobile app distribution of images.

The disadvantage, of course, is that now your training distribution is different from your dev and test set distributions.

But it turns out that this split of your data into train, dev and test will get you better performance over the long term.

And we'll discuss later some specific techniques for dealing with your training sets coming from different distribution than your dev and

(DESCRIPTION)
New slide, Speech recognition example. A photo insert shows the cockpit of a car. The rearview mirror is circled for emphasis.

(SPEECH)
test sets.

Let's look at another example.

Let's say you're building a brand new product, a speech activated rearview mirror for a car.

So this is a real product in China.

It's making its way into other countries but you can build a rearview mirror to replace this little thing there, so that you can now talk to the rearview mirror and basically say, dear rearview mirror, please help me find navigational directions to the nearest gas station and it'll deal with it.

So this is actually a real product, and let's say you're trying to build this for your own country.

So how can you get data to train up a speech recognition system for this product?

Well, maybe you've worked on speech recognition for a long time so you have a lot of data from other speech recognition applications, just not from a speech activated rearview mirror.

Here's how you could split up your training and your dev and test sets.

So for your training, you can take all the speech data you have that you've accumulated from working on other speech problems, such as data you purchased over the years from various speech recognition data vendors.

And today you can actually buy data from vendors of x, y pairs, where x is an audio clip and y is a transcript.

Or maybe you've worked on smart speakers, smart voice activated speakers, so you have some data from that.

Maybe you've worked on voice activated keyboards and so on.

And for the sake of argument, maybe you have 500,000 utterences from all of these sources.

And for your dev and test set, maybe you have a much smaller data set that actually came from a speech activated rearview mirror.

Because users are asking for navigational queries or trying to find directions to various places.

This data set will maybe have a lot more street addresses, right?

Please help me navigate to this street address, or please help me navigate to this gas station.

So this distribution of data will be very different than these on the left.

But this is really the data you care about, because this is what you need your product to do well on, so this is what you set your dev and test set to be.

So what you do in this example is set your training set to be the 500,000 utterances on the left, and then your dev and test sets which I'll abbreviate D and T, these could be maybe 10,000 utterances each.

That's drawn from actual the speech activated rearview mirror.

Or alternatively, if you think you don't need to put all 20,000 examples from your speech activated rearview mirror into the dev and test sets, maybe you can take half of that and put that in the training set.

So then the training set could be 510,000 utterances, including all 500 from there and 10,000 from the rearview mirror.

And then the dev and test sets could maybe be 5,000 utterances each.

So of the 20,000 utterances, maybe 10k goes into the training set and 5k into the dev set and 5,000 into the test set.

So this would be another reasonable way of splitting your data into train, dev, and test.

And this gives you a much bigger training set, over 500,000 utterances, than if you were to only use speech activated rearview mirror data for your training set.

So in this video, you've seen a couple examples of when allowing your training set data to come from a different distribution than your dev and test set allows you to have much more training data.

And in these examples, it will cause your learning algorithm to perform better.

Now one question you might ask is, should you always use all the data you have?

The answer is subtle, it is not always yes.

Let's look at a counter-example in the next video.