## Using fast.ai to sort text documents (part 1)

For this next project, I want you to think about your spam filter, right? A spam filter looks at an incoming email message and says, does this go in the spam filter or does it go into your inbox? And it's looking at the language being used in that message to make that decision. We're going to do the same kind of thing, sorting a pile of documents into two piles.

This is a two step process. The first process is that we take a model that is able to recognize the text of the corpus that we're using. That corpus could be a set of news articles or medical documents or in our case, it's going to be a special set of tweets. So first, we teach the language model to recognize the language of our corpus. The second step is making the classification model, and the classification model is actually the model that takes that knowledge of the language and sorts it into those two piles. So let's start out with building our language model for this project.

Okay, for this exercise, we're going to go back to Google colab and go to GitHub, "quartz". And we'll be looking at the last notebook in the set here, gg-sort-tweets-with-fastai, so we want that one. And here's our plan, we're going to try to look through a set of tweets to see if we can teach the computer to detect tweets that are checkable statements of fact. Let's take a look at the kinds of tweets that were in question here. These are #TXLege. So these are Texas legislature tweets or people in the legislature or tweeting at the Texas legislature. So there is a bunch of different information that's in this hashtag TXLedge, lots and lots of tweets. And sometimes there's statements of fact. Sometimes there are not. And folks at the Austin American Statesman in Texas wanted to see if they could detect when somebody tweeted something that was fact checkable.

Recognizing fact check ability seems like something that only a human could do. But we're actually able to teach a computer to do it. And it's an example of searching documents into two piles. In this case, checkable or not checkable. And again, we're going to use transfer learning to do this. It's kind of cool, there are two steps to this. First, we make a language model and that is just a model that understands the language we're working with, that's English. And additionally understands the corpus that we're working with. And that's the particular kind of tweet that we're going to be looking at. So that's what we'll do first, we'll make a language model and then in the next video we'll make the classification model. That's the model that actually sorts the tweets into two separate piles. So first, we'll get all the tweets into our notebook. Then we'll make the language model, then we'll make a classification model and then we'll try to protect predict the checkability of a tweet that has not been seen by this model.

And this was done as a project with the Austin American Statesman newspaper. And the folks there did a lot of work to hand code these tweets, so thanks to them. We are going to need to use the GPU, so let's go turn that on. And it's on, good, I'm all set. So then I need to load fast.ai with this cell. And we're gonna run this cell to bring all of that into the notebook and get the libraries ready. Next, we're going to get our data. This is actually just two CSV those are comma separated values files. If we look in here, you can see that they're just two of them in there. One is called "hand_coded_austin_tweets.csv" and one is called "tweet_corpus.txt". So the first one, this hand coded Austin tweets CSV, let's take a quick look at that.

You can see this is just the first five rows of that file. There are 3,700, almost 3,800 tweets in here that they went through, and they did they said, is it checkable, true or false? So here is a tweet or part of it. And they said that that was false. That means not checkable. And then this one is true, it is checkable. And so this is how we're going to train the model. But we also remember we need to build that language model, the thing that understands both English and this particular language of Twitter. So I also pulled together just a bunch more tweets. These have not been hand coded, they're just tweets, they're tweets with that same hashtag, the #TXLege hashtag and I pulled them in. So we're going to use all of that data to build the language model.

The first thing I'm going to do is here combine just the tweet column of those two files. So this will create a new thing called LM for a language model tweets. I'm going to concatenate those two files. And if we play this, you'll see that there were hand coded tweets where 3,797, the corpus of tweets, that second file with only tweets had a little bit under 3,700. And so together we have about 7,500. All right, so the next thing we're going to do is we're going to save that into our file system here real quickly, because that makes it easier to pull it out later. And now we're going to make a special kind of data bunch. Remember before we were making data bunches of images. We're going to make a language model data bunch. And we're going to we're basically saying, hey, we're going to pull together all of these tweets and we're going to use them for a language model. That's what's going on here. So we're going to do that.

So load all of those tweets into data.ln. And next, we're going to make our learner. This is the learned variable. And remember, we're gonna go and combine what are all our tweets now because data.ln is our data bunch with all that-- all those tweets. And we're gonna go get that pre-trained model. So this is called the WikiText-103. Here it's represented as a AWD_LSTM, but it is the WikiText-103 model and together fast.ai will combine that and that into the learner and that's how we're going to make our language model. So this model is actually trained to try to predict the next English word in Wikipedia articles, that's how it was trained. In the process it learned in a way the patterns of English as represented in long Wikipedia articles. Then we're bringing in those extra tweets to combine and do transfer learning. So you I just quickly downloaded that WikiText model and now we're all combined.

So the next step we're going to do is something that we haven't explored yet. It's called finding the learning rate. The learning rate is a concept in machine learning where it's sort of I like to think of it like looking for marble in a football field, right? So that's what the computer is trying to do, it's trying to find the best solution to our problem. And if you were to take very, very tiny steps to find that marble, you wouldn't-- it would take you forever. And if you took two big steps or leaps, you might leap over the answer. It's kind of like that. You have to figure out the happy medium. And doing that has always been really hard, except that fast.ai actually makes it much, much easier. It's one of the things that really helps. And by finding the right learning rate, we can actually make the model work even better. So what we're going to do is run this little, little operation here, which allows us to find the learning rate. This could take a minute. So I'm going to skip ahead.

Ok, so it's done and then to see the graph, we just hit this next play button and this is a very useful graph to try to determine what's the best learning rate. And it turns out that the best learning rate is when the learning rate really starts to dive down. So 1e-2 is probably pretty good. It's, it's definitely on its way down at this point. And that's that's going to be the learning rate we use. So these next two cells are actually modifying what we know is fit one cycle to train our models. And we're gonna use this value for the learning rate, which is 1e-2, which is right there. And then this has another value we're not going to go into, but it's about it's called momentum and it's another what's called hyper parameter. So learning rate and momentum and some other things are hyper parameters which allow you to sort of fine tune how your model learns and how well it does. But right now, we're just going to go ahead and fit these two together. It'll fit. We'll do this one. And then when this cell is done, it will do this one next. So this is just waiting for that one to end.

Ok, and then we're going to-- we're done. We've got an accuracy up to about 37 percent. We're gonna do another step, which is to unfreeze the original model. The original WikiText-103 model, and then let some of what we've learned here in this process with the tweets sort of get infused back into the model. It's a little bit technical, but basically we're unfreezing the previously established layers of the WikiText model.

Ok, so our accuracy is about 40 percent. Now, being 40 percent accurate might not seem great, but really what this is being trained to do is to guess the next word in what would be a normal tweet or a Texas legislature tweet. And if you're going to guess a word and you're right one out of three times, or better than that. So better than a third of the time, that's pretty good, actually. So our accuracy is pretty high and we can actually see this in action with this next cell. This is a little trick to show how I would guess the rest of a sentence if you started with, "I wonder if this," and have it guess the next 40 words. So I'm going to play this and see. So one sentence here at the beginning, "I wonder if this is the Texas primary application judge, Texas deserves safe deserved by the CPS." Ok, it doesn't make a whole lot of sense, but it's pretty close to English and it definitely looks like a tweet. So it's doing pretty well. At least we know that the language model has a sense of what tweets sort of look like and would be able to maybe recognize patterns in there. Remember, these aren't real. They were generated by a machine. But that's not our goal, really. We want to use these smarts to help predict whether or not a tweet is checkable or not, and that's what we'll do next time. In order to do that, we need to save the values inside the model that the that are being used to understand the tweet language, and that's called the encoder. So we'll save that value, that encoder value into the file system and we'll use it in the next video.