Thank you for the introduction. My name is Abigail and I'm a data scientist. I'm going to talk about a few things today. I'll talk about Large language models (LLMs), like ChatGPT. I'll talk about classification problems (where we categorize items into specific groups or classes). And I'm going to talk about a particular Python library that if you're trying to solve certain types of problems with LLMs, I think you should use, even if you otherwise code exclusively in R.

But why I'm really here today and why I picked this topic to talk to you all is that I've been on both sides of federal government hiring. Like, as an applicant and on the agency side, where we're trying to hire. And there are a lot of problems in federal hiring which are not things you can fix with better technology. There are genuinely difficult matters of law, or issues where authority to change something is incredibly dispersed throughout not just agencies but individual hiring managers. But my hope is that there's a small piece of the hiring problem that actually is a data science problem, and that we can help solve with the help of LLMs. And that part is just the part where we assign a title to a job based on what the agency says they're looking for in the job description.

USAJobs is the primary platform where the U.S. federal government posts civilian job vacancies.

So if you're a data scientist - or anything else! - this is where you're going to go.

But when I look for data science jobs, I get a lot of jobs that just aren't what I'm looking for. They're analyst jobs where you're using primarily drag and drop tools instead of code, or they're software engineer jobs where they're looking for coding skills I don't have, or maybe they're program manager jobs that are about leadership and coordination. Or it's really hard to tell what they're looking for, because there's either not that many details, or there are so many different tasks being listed that it looks like they're trying to hire a whole department in a single person. And I feel like, guys -- what's the line, dammit Jim, I'm a data scientist, not a whatever else! [That's a Star Trek reference, guys]

There are a lot of reasons why this happens. And we can argue about how broadly we should define data science – "what does it even mean to be a data scientist?" but it doesn't really matter. The important piece is, we want potential applicants to be able to better find the kinds of jobs they're interested in. And I know I'm not the one with this problem.

I attacked this in a few different ways, but where I ended up is: this is a classification problem. We have some text, and we want to sort these job listings into different buckets - different job titles – based on that text.

And this is useful for a few different reasons: we want to help job seekers, but also, making job announcements more legible can promote better internal research and analysis. Like, for

instance, at one point at the Army, I was trying to model which listings led to offers actually being made - that is, was a search successful? There are also efforts looking at time to hire by different job type - like, how long does it take to hire data scientists? I've also done analysis of actual position descriptions, which have some similarities with job listings, and we were trying to count the number of data science position descriptions (or the number of people on those position descriptions).

In other words, we want to turn the text of a job posting or description into something short and accurate. We want a model where you can put in some text about a job and get out a description of that text, or a job title.

___

Here you can see me asking GPT to find me data science jobs. And this is basically what I'm doing! Except instead of using the web interface, I use the OpenAI API, an interface that allows developers to access and utilize the capabilities of OpenAI's models like GPT-3.5. It lets me iterate through job postings, send each one to GPT and ask it to write me a job title that describes the duties.

_____

We're going to take a quick detour from USAJobs to talk more generally about classification problems.

HAM vs. SPAM -- HAM being the kinds of emails you want going to your inbox -- is a classic example of a classification problem. When you get emails, your email client wants to sort every email you get into these two buckets. And it does that using a variety of different features of the email. And one of those sets of features is the words in that email. Based on previous emails, your email client has a model running in the background which does this sorting, and it improves on the basis of the feedback that humans give it, like when you click "report spam" in gmail.

_____

And this is an example of a confusion matrix, or a way we can evaluate how good our classification models are. What we're looking at is a grid of how our imaginary MODEL evaluated a set of emails, vs. what the actual ground truth was (let's say, how the recipients of the email would evaluate the emails). Those numbers at the top left and bottom right - those are the ones it got correct. The other ones it got wrong. This would be a really bad classification model, in the context of SPAM -- we should be getting more than 94% correct, unless we're talking about just exceptionally good SPAM! But in another context - a harder classification problem - these might be great numbers.

Now, you may notice – in this problem, I have two buckets, and I know what they are ahead of time. But with titles, I'm not actually picking ahead of time what the possible list of titles are, I'm

going to let GPT pick those. (I tried a standard classification problem – called a multi-class classification problem, meaning there were more than two buckets but they were pre-defined. I didn't think it worked very well. But if you wanted to quibble, you could call what we're doing here something more akin to "generative labeling" and not classification because there aren't pre-defined buckets I'm using. I'm going to call it classification.)

___

Back to USAJobs listings. This is what a sample listing looks like. This is for a job in the occupational category "data science", and the job title is "data scientist". There are different sections to this posting. You're probably familiar with this general format. I focused on the 'duties' section, or the list of specific things the job involves - that's the text I'm using in my model. I thought about using additional sections. Like, maybe the "qualifications" section should tell us something about what the job involves? But I'm not actually sure it consistently does! I think a lot of times those are fairly unrelated! So I just used the duties section.

_____

This is closer-up on the duties section for this particular job. I highlighted some of the duties. This position is creating data architecture, pipelines, doing ETL, and automating tasks. Raise your hand or just shout out what this job is - someone. It's not a trick question.

_____

Right - so, this is a data engineer job! And that's how my model classified it. It's also not just misnamed, as in the job title is incorrect, I think it's actually misclassified in terms of the four-digit occupational code. That is, for those who are not familiar in obsessive detail with the Office of Personnel Management - each of the jobs listed on USAJobs has what's called an occupational code, or a four-digit number that corresponds to sort of a broad position type.

There's a big OPM book saying what belongs in each occupational code. Data Science, or 1560, is a very new code. Previously we were getting stuck in all other kinds of codes. But data engineer is probably a 2210, for Information Technology management. Maybe it's a 1550 for Computer Science. I don't know! But I don't think it's a 1560.

That kind of analysis – determining potentially misclassified jobs - is also something you could use a text classification model for. But the bigger priority for me is -- let's say you're a data engineer looking for jobs! You should just be able to search "data engineer" and you should see that in this job title! And currently, you can't.

However, this method doesn't solve everything by any means. For instance, if there's limited information in the duties section, we shouldn't expect to get an accurate title.

_____

For instance, what is this job posting, exactly? Is this a data science job? A data analyst job? An economist job? Unclear! And we shouldn't expect good model results for this vague of a duties section.

___

This is my full workflow for this project.
- I pulled 2022 to present-ish jobs classified as 1560, or data scientist. Some of these are also interdisciplinary, meaning they have multiple occupational code designations, and you can pick the ones to apply under based on which qualifications you meet. I pulled these via the USAJobs past jobs API, which is fairly new, it's great.
I dropped those job listings with no duties, or very short duties, including those which basically said "go to this other website to see the duties"
I then used a package called Marvin, which we'll get to next, to get structured output from GPT-3.5 to get various kinds of output about each listing, including a job title. I used GPT 3.5 because it's cheaper than GPT-4.

And then I did various analysis and iteration. I'll show you some of the analysis. The iteration included – you'll see on the next slide, I changed my prompt because I thought sometimes that when the duties section references the actual official job title, then the model was just returning that job title, and I didn't want it to do that.

_____

I wrote a lot of code for this project, including for parts that are beyond the scope of this presentation, as well as for making the figures I'll show you. But for the actual prompting piece and for communicating with the OpenAI API, THIS IS THE ONLY CODE I wrote, and then there's one line where I call this code in a function.

Literally, this is all I'm doing. I'm not managing error handling for the API - like, what to do if and when it fails. I'm not writing a prompt that's like PLEASE JUST GIVE ME THE JOB TITLE. I'm not doing any of that. This is 9 lines of Python code!! And what Marvin does behind the scenes it's just managing not just the communication with the OpenAI API, but also, it's reshaping the docstring I wrote here into a prompt that will ONLY GET ME the job title. Basically, it's getting this LLM to act like I need it to act. So I HIGHLY RECOMMEND THIS if you are you doing classification, or any other kind of Natural Language Processing research type projects you want to do with LLMs.

If you're coding in R, there are libraries for calling Python code from R. And this isn't even really Python code? The major thing we're doing here is this two-line docstring. That's what's doing the work for generating the actual prompt that gets sent to the LLM. And you can use this for a variety of different NLP problems.

_____

So, what was the point of this again? Why am I doing this?

We can get a view of the kinds of jobs getting classified as 1560s. we can get a prototype job labeling system for applicants. we can a way to highlight jobs that might be mislabeled (as in, a disconnect between official job title and job duties), or misclassified (a disconnect between occupational category and job duties).

So let's go over a little of what we got:

We're looking here at the top job titles from my LLM vs. from the official job titles from USAJobs. The biggest thing you'll see is that "Data Analyst" does not make the top 10 official job titles, but it's the second-most-popular job title that my model generated. And those are the jobs where maybe it doesn't seem like there's coding, where it mentions dashboards, maybe Excel. And those actually might be data scientist jobs! Maybe the duties aren't accurate! But if they're not accurate - if the model can't tell - neither can potential applicants!

Honestly, I don't know how useful "data science analyst" or "data science specialist" are as titles. But relative to "interdisciplinary" or even "manager" or "engineer/scientist", I think these are all more descriptive titles.

Another thing to notice -- some of these positions are describing significant leadership responsibilities! Like, Chief Data Scientist or Chief Data Officer. Those might or might not how the agency is envisioning these jobs. But if that's the degree of responsibility being described, that'd be a good thing to convey in the title. Also, if you're listing a job that reads like Chief Data Scientist but it's, say, a GS-13, that's a disconnect between level of pay and level of responsibility. That should be flagged internally, and this kind of model can help do that.

This is another way of looking at the results. These are the job titles that the LLM generated but that were **never** found in the actual, official job titles. You can see "Data engineer". I don't love all of these job titles. But some of them are useful. Like, data science manager, data engineer, geospatial analyst -- these all refer to job titles where someone could really be looking for that, and where we make it harder for them to find those jobs than we need to.

Next, here are some examples of job titles with big discrepancies from the official ones to my LLM-generated ones. This is a job that my model labeled "data analyst". You can see -- PIVOT TABLES. MACROS! No mention of programming or coding.

Here's another one: technical program manager. This is a leadership job. It looks really cool! It's being labeled as a data scientist job, which is not that helpful to applicants.

But what's better than ad hoc assessment would be systematic assessment. What I actually want to be doing is having someone who is an expert on this look at some lists of job duties and label the jobs. Then we can compare my model-generated titles to the official job titles, evaluate both, and potentially see where each of them does well.

We might need something more complicated than the confusion matrix I showed you earlier. For instance, we might want to analyze the actual meaning - the semantic content - of each expert-generated label, and then compute the distance between that label and my job titles.

But the basic principle is the same - it'd be great to have labeled data and to do actual assessment.

Quickly, I tried a few other ideas as well using GPT. Named Entity Recognition worked quite well. I was able to pull out names of tools and software in the duties. I compared the response from GPT 3.5 to the results I got from keyword searches. Keyword searches worked well, too, but it was higher-effort because I had to spend a lot of time looking at actual results and iterating -- not just because I was missing some keywords, but because what keywords you search for and the specifics of what matches you count depend strongly on the actual dataset. You can think of like, the difficulty of finding "R" in a dataset via keywords or regular expressions - but there's actually a lot of things like that.

I also tried requesting a broad job category in addition to a job title, where I provided the buckets to use. I thought it worked ok but not great. And then I tried a classification model looking for mismatch between my LLM job title label and the official one, and I couldn't get it to work well at all.

What's next?

I'd love to actually have someone in the federal government using this kind of thing! If you're interested in building on this, my code is available on GitHub, it's ... reasonably clean, I can clean it up more or talk you through it if you're interested.

And if you have a text classification problem you want to attack with LLMs, I highly recommend that, and specifically Marvin, and you can also look at my code for that.

Also! There's a happy hour tonight - Data Science DC is organizing it - it's at 5 PM, please come! We can talk more about this.