6/30/19

On Sunday 6/23, I suddenly received a message from my friend from Brandeis. Having some few chats with him, he suddenly introduced me to a job opportunity.

A team in Korea was apparently trying to find a software engineer to make their idea to real life.

Though I was interested at the beginning, I initially had to say no. This was because I had no knowledge on what they wanted me to make and more importantly, they had a strict deadline of finishing it in 3 days.

However, being myself (with sometimes a bit too much courage), I ended up telling my friend that I would need to look into what is available and I could decide if I can do it or not the next day.

Basically, their product was to transcribe meeting logs digitally. As I headed towards Yokohama to go to Karaoke with my friends, I quickly searched on Google for APIs since it was probably impossible to engineer some NLP speech recognition package in a day. I quickly resolved to Google Cloud Speech API. Reading the documentation, I figured that Google has our lives so much easier and immediately told my friend that this was going to be possible for me to complete. With his quick response, I was added to their Slack team and was explained about their product in further detail.

And this how my 2 day journey of creating the prototype of Dolly began!

First, since their UI seemed to look like it was transcribing speech to text as the meeting went on, I decided to go with the streaming option of the Google Cloud Speech-to-Text API. Finding out that it was easy to configure with multiple languages, I decided to support four languages in this prototype: English, Korean, Japanese, and Chinese. After doing some rewrites, I was able to get this function to work and it was now writing from speech to text. The other function the team wanted me to implement was the ability to recognize different speakers. However, looking through the documentation, there was no such functionality implemented in the API yet. Luckily, I found out that they had a beta version that would detect different voices although honestly, the accuracy isn’t that high yet. With this functionality also implemented, my final step was to output some interesting results including words that were repeated over 30 times and 10 times over the meeting. Another functionality that the Korean team wanted was “Dolly’s random suggestions”. Rather than completely “random”, I thought I could use some feature/keyword extraction technique from NLP and decided to output 5 of the most relevant words that may not have been repeated overly during the meeting. This finally concluded the development of the prototype of Dolly! However, it wasn’t the end of my job because I had to make sure that the Korean team can run this program locally. I made a whole instruction list from the very beginning of how to install Python so I will be making several posts in the future about this too.

In the end, I am glad that my friend introduced me to this opportunity and I was able to meet new people through work. I am also thankful that I was able to expand my knowledge in this new field of speech recognition and NLP.

Keep reading to see how I actually implemented Dolly!

# Hello, World!

This is my first blog post made with flask, peewee, and postgresql!

<br>

<br>

Shout out to this blog post I referenced to make this: https://charlesleifer.com/blog/how-to-make-a-flask-blog-in-one-hour-or-less/

<br>

<br>

Since the blog post was written to work with sqlite, I had to modify a little to work with postgresql. The reason for this is because sqlite runs in memory and stores data in a file. Since Heroku has an ephemeral file system, the data stored will be periodically cleared (!) and therefore, it would be necessary for us to implement the databse with postgres.

<br>

<br>

I thought of writing briefly about the switch from sqlite to postgresql since some of you may want to deploy to Heroku and have an actual working blog that doesn't erase your data.

<br>

## Move to PostgreSQL

1. Add the following import statements

<br>```from playhouse.postgres\_ext import \*```

<br>```from dotenv import load\_dotenv```

<br>- postgres\_ext is for the searching capability, which I decided not to include so I will not really touch on it but you can read the documentation (http://docs.peewee-orm.com/en/latest/peewee/playhouse.html#postgres-ext) and make the necessary changes.

<br>- The dotenv package is for loading in the DATABASE\_URL (as well as all the other credentials including ADMIN\_PASSWORD and the flask SECRET\_KEY).

2. Next include the line

<br>```DATABASE = os.environ['DATABASE\_URL']```

<br>- This will get the DATABASE\_URL from Heroku

<br>- That's it for the Python part!

3. We'll get our PostgreSQL addon on Heroku. On your command line run

<br>```$ heroku addons:create heroku-postgresql:<PLAN\_NAME>```

<br>- For the plan name I chose ```hobby-dev```

4. Create the database on Heroku. On your command line run

<br>```$ heroku run python```

<br>```from your\_app import db```

<br>```python your\_app.py```

5. Now we are ready to deploy to Heroku

<br>```$ git add .```

<br>```$ git commit -m "PostgreSQL "```

<br>```$ git push heroku master```

<br> and \*\*that should be it!\*\*

6. Notes: Accessing database on Heroku from command line

<br>```$ heroku pg:psql```

<br> When I ran this, I got the message:

<br> ``` ▸ The local psql command could not be located. For help installing psql, see

▸ https://devcenter.heroku.com/articles/heroku-postgresql#local-setup```

<br> This happens when the PosgreSQL versions do not match. You can check the version by

<br> ```$ heroku pg:info``` and

<br> ```$ psql --version```

<br> To install you can run the following command:

<br> ```$ brew install postgresql@12```

### References

- About SQLite on Heroku: https://devcenter.heroku.com/articles/sqlite3

- PostgreSQL on Heroku: https://devcenter.heroku.com/articles/heroku-postgresql

- Flask + Heroku + Postgres: https://dev.to/paultopia/the-easiest-possible-way-to-throw-a-webapp-online-flask--heroku--postgres-185o

- Flask + Heroku + Postgres (in Japanese): https://qiita.com/croquette0212/items/9b4dc5377e7d6f292671

- Building a RESTful Blog APIs using python and flask: https://www.codementor.io/@olawalealadeusi896/restful-api-with-python-flask-framework-and-postgres-db-part-1-kbrwbygx5

- Installing PostgreSQL 12 with brew: https://installvirtual.com/how-to-install-postgresql-12-on-mac-os-with-brew/

- Installing PostgreSQL 12 with brew (in Japanese): https://note.com/haraaa/n/ndf9d96d20ed0

- Markdown reference: https://about.gitlab.com/handbook/markdown-guide/

### Other references

- Setting up Postgres, SQLAlchemy, and Alembic: https://realpython.com/flask-by-example-part-2-postgres-sqlalchemy-and-alembic/

- Deploying a Flask App with Peewee to Heroku: https://swifthorseman.com/2015/06/18/deploying-a-flask-app-with-peewee-to-heroku/

- Migrating Homebrew Postgres to a New Version: https://olivierlacan.com/posts/migrating-homebrew-postgres-to-a-new-version/