Clerkie Coding Challenge

Congratulations on making it this far in our interview process. We're excited about the opportunity to have you join the team and are very appreciative of you taking the time to complete this coding challenge.

The purpose of this coding exercise is to evaluate and get to know you along 4 dimensions:

1. AI / ML Coding Experience (e.g., how good and familiar you are with AI / ML frameworks and relevant libraries)
2. Affinity for finance (e.g., your ability and passion for understanding financial models and problems)
3. Self-proficiency (e.g., how much you can get done on your own)
4. Collaboration (e.g., your willingness and courage to ask for help)

But, please note that you do not have to excel in all 4 areas to get the job. These dimensions do, however, give us some good data points as to how well you enjoy and excel at AI / ML development.

Deadline

You will have until **Sunday (8/26)** before midnight (e.g., by 11:59 PM PST) to complete this challenge. Please take your time and don't hesitate to reach out (via email or text message) if you have any questions or concerns.

Submission Requirements

Please email your final work to talent@clerkie.io :

- If you decide to use AWS, please include a **copy of your AWS spend during the 10-day challenge period (which can be found in AWS Cost Explorer)**, **we will reimburse up to $100** of your server costs
- Please **include a short 2-10mn demo video of your system in action** (i.e. enter in various requests to show that your system classified them accurately and responded appropriately). The video can be shared in a variety of ways (e.g., Dropbox link, or raw video file via email, or included in your git repo, etc – whatever is easiest).
- Please include a **link to your github repo**. (NOTE: please ensure that all **dependencies/libraries** are included in the repo and include a **README** with specific instructions/steps for how to run/test your mini-system. We will check for coding cleanliness and best practices and **may download and test** the system locally to verify your work)

Coding Challenge - The beauty of speech and the art of conversations

Every day we come into the office, thinking of novel ways to help families with their financial decisions. One of the many challenges we've had to overcome has to do with "making Clerkie smart enough to not only answer questions but to also hold a conversation and offer insightful recommendations that our users might not have thought about during their conversations". We've figured out how to solve this fascinating problem and would love to invite you into the fold.

For the purpose of this coding challenge, we've drastically simplified the aforementioned problem and reduced its scope to the simple (yet realistic) scenario below. Enjoy!

Scenario: Mary is a new user at Clerkie who wants to buy her dream house. She thinks she has $200,000 saved up, but she is not sure... She wants to know if she can afford this new house in Los Angeles (which is worth $2.3M).

Mary logs in and is thinking of asking Clerkie 3 categories of questions (which by the way can come in multiple forms):

- **Category 1:** Check Balance (e.g., "How much do I have saved up in my account ending in (x9898)?", "How much money do I have stashed up in my Bank of America?", "What is my current BoA savings balance?", etc)
- **Category 2:** Budgeting (e.g., "How much wiggle room do I have in my budget?", "How much spare money do I have?", "How much money can I save each month?", etc)
- **Category 3:** House affordability (e.g., "Can I afford a $2.3 million house?", "Can I buy a $2.3M crib?", etc)

Objective and System Requirements

Using your knowledge of AI / ML, your objective is to **build a mini-system capable of helping Mary make the right financial decision.**

Your system should be **written in Python and/or Node.js (on AWS or on your local machine)** and should be able to execute the following 2 tasks:

- accurately **identify and understand an incoming user request** from the 3 categories above
    - recognize Mary's requests (questions) and **classify them in the proper category** (e.g., (1) "Can I afford this $2.3 million house?" > belongs to Category 3. (2) "How much money is in my bank account?" belongs to category 1. (3) "Can I buy a $2M bugatti?" does not belong to any of the 3 known categories)
    - recognize and classify Mary's request **regardless of how the question is asked** (e.g., "Can I afford this $2.3M house?", "Can I buy a $2.3M crib?", "Can I get a mortgage for a $2.3 million pad?", etc)
    - give an appropriate answer **if the incoming requests doesn't belong** in the 3 categories above (e.g., "I'm sorry, but I did not understand your request... I'm still learning so I can only help you with 3 things...)

  - recognize, **extract and tag the relevant name entities** in the requests (e.g., recognize that BoA is a bank + recognize that BoA and Bank of America are equivalent + recognize that BoA is a different bank from Chase or CitiBank)
- **answer** the request **correctly** based on the 3 categories above (this is especially **important** since your system is dealing with money – financial answers and advice must be accurate 😊)

Tip #1: Feel free to use your preferred ML model to build your system (RNN, CNN, GAN and/or Reinforcement ML models typically work well for NLP)

Tip #2: If you decide to use AWS, be thoughtful about your server costs. You will not need to use the most powerful EC2 instances for this exercise. You can easily use smaller EC2 instances (e.g., micro, medium, etc) to generate your feature sets / training data and you could leverage more powerful Px or Gx GPU EC2 instances to train your NN.

Some of us like to go above and beyond or may just want to commit more time than others to refine their AI systems. For those kindred spirits, we've laid out a few extra credit ideas (3 in total) that you can choose to implement. **Remember, these extra credits are not required! You can choose to do them all or none of them** 😊

Extra credit #1: able to **suggest an additional topic** of conversation **OR additional factors to consider** for each request
Extra credit #2: able to **learn from past mistakes** to avoid making the same mistake (e.g., if the system encountered a question that belongs in one of the 3 categories above but it classified and answered that question incorrectly before, it can collect the feedback and classify it correctly the next time)
Extra credit #3: able to **learn from new interactions to create new categories** of questions and answers (e.g., if the system encountered a question that it did not recognize OR that does not belong in any of the 3 categories above, it can learn from that event to create and answer a new category of questions)

Assumptions

- Mary has **4 accounts**: (1) a Bank of America checking account ending in (x9898) with a balance of $9,000, (2) a Bank of America savings account with a balance of $100,000, (3) a Chase savings account ending in (x9898) with a balance of $80,000 and a (4) maxed out credit card with a balance of $15,000
- Mary **makes $6,000 each month** and spends $3,000 on **rent**, $1,000 on **utilities and groceries**, $1,000 on **shopping** and $500 on **restaurant dining** each month
- The house that Mary wants to buy is in a nice neighborhood in **Beverly Hills**, where home prices have **increased by 40%** in the last 3 years
- In the last 4 months, comparable home sales have **declined by 10%** and home prices have **declined by a modest 5%**
- Currently, to **qualify for a $1.5M+ mortgage**, the banks require a 20% down-payment from potential home buyers
- The Annual Percentage Rate (APR) on a 30yr mortgage with a 20% down-payment is 5%