Contents:

Job posting that the answers is targeting

Job Description

Alexa is the groundbreaking cloud-based intelligent agent that powers Echo and other devices designed around your voice. Our team is creating the science and technology behind Alexa. We're working hard, having fun, and making history. Come join our team! You will have an enormous opportunity to impact the customer experience, design, architecture, and implementation of a cutting edge product used every day by people you know.

We're looking for a passionate, talented, and inventive scientist to help build industry-leading conversational technologies that customers love. Our mission is to push the envelope in Artificial Intelligence (AI), Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), Machine Learning (ML), Dialog Management and Audio Signal Processing, in order to provide the best-possible experience for our customers.As an Applied Artificial Intelligence Scientist, you will work with talented peers to develop novel algorithms and modeling techniques to advance the state of the art in spoken language understanding and dialog systems. You will leverage Amazon's heterogeneous data sources and large-scale computing resources to accelerate advances in artificial intelligence. You will collaborate with other scientists and work in a fast paced team environment. Your work will directly impact our customers in the form of novel products and services that make use of speech and language technology.

Basic Qualifications

- Graduate degree (MS or PhD) in Electrical Engineering, Computer Sciences, or Mathematics with relevant work experience
- Familiarity with standard artificial intelligence and machine learning techniques, scientific thinking, and the ability to invent
- Familiarity with programming languages such as C/C++, Java, Perl or Python

Preferred Qualifications

- PhD with specialization in speech recognition, natural language processing, or machine learning with at least 3 years of related work experience
- Strong publication record
- Strong software development skills
- Experience working effectively with science, data processing, and software engineering teams
- Proven track record of innovation in creating novel algorithms and advancing the state of the art
- Entrepreneurial spirit combined with strong architectural and problem solving skills

- Excellent written and spoken communication skills.

Answers

1. We A/B tested two styles for a sign-up button on our company's product page. **100** visitors viewed page **A**, out of which **20** clicked on the button; whereas, **70** visitors viewed page **B**, and only **15** of them clicked on the button. Can you confidently say that page **A** is a better choice, or page **B**? Why?

   Answer: When people click on button after viewing page A, we get conditional probability. Probability of Click (C) given A, i.e. P(C|A)=20/100 = 20%. When viewers click button after viewing page B, their success rate is P(C|B) =15/70 = 21% (had to use a calculator). So, we can say that even if number of clicks for A is more than B, B is slightly better since the conditional probability is higher. Unfortunately, this is not really very conclusive as the numbers are not statistically significant. Typically, more than 1000 users should be present in each group (and we need to make sure they are adjusted for demographics like age, etc.). Also, if we have some more data, we could perhaps discover more categories for which the success rate of A is higher (e.g. college students).

   I feel like this approach is exciting even for voice-based agents like Alexa. For example, I get "Alexa Skills" emails from your team announcing different skills. So if you divide the users into A/B groups and send two different emails about the new skills to see which users which use, we can use the ratio of number of people who tried the skill to the number of people who opened the email to see how effectively we can convince people to use new Alexa skills that make more revenue for Amazon.

2. Can you devise a scheme to group Twitter users by looking only at their tweets? No demographic, geographic or other identifying information is available to you, just the messages they've posted, in plain text, and a timestamp for each message. In JSON format, they look like this:

   ```
   { "user_id": 3, "timestamp": "2016-03-22_11-31-20", "tweet": "It's #dinner-time!" }
   ```

   Assuming you have a stream of these tweets coming in, describe the process of collecting and analyzing them, what transformations/algorithms you would apply, how you would train and test your model, and present the results.

   Answer: Looking at the above JSON, I think we can extract some features that can be used for unsupervised clustering. First, we can extract the days of the week (e.g. weekday, weekend, day of the week, etc.) and time of day (e.g. early morning, morning, noon, afternoon, evening, night, late night) to figure out when the user is active. Second, from the tweets, we can use the hash-tags as an indicator of the topics that the user is interested in (I am assuming here, that re-tweet/follow data is not available). We can then create the feature set for each user:

	weekday,	weekend,	Mon,	⋯,	Sun,	early morning,	⋯,	late night,	#dinner-time,	⋯
user1	1	4	0		3			1	1	
user2	⋯									

   Collection: As the user tweet comes in, we can add to the table (NoSql table since we don't know schema before hand), and increment the count of the tweets.

Analysis: We can then perform clustering based on this data (after normalizations, of course). To note though, that this dataset will be very sparse as the users topics they use in hash tags may be too many, and each user will use only a few. So we either need an algorithm that can handle sparse data (e.g. Amazon's DSSTNE) or perform feature reduction by preprocessing. For example, we can identify the subset of important topics (e.g. based on popularity), or create some semantic similarity (e.g. using hypernyms, etc.) measure to reduce dimensionality.

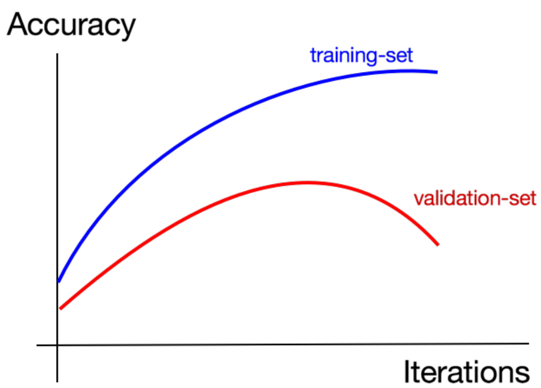
Test: In unsupervised learning the purpose of the clustering is very important to create the metric for testing. For example, if the above clustering is done for recommendation of whom to "follow", then we can have "the number of people following others in the same cluster" as a metric.

Presentation of results: Presentation strategy also depends on the purpose of clustering. For example, in the above, I would choose show the different people as nodes, each node color depending on the cluster, and show people who follow each other with the same color edge. Then use some graph visualization tool to simplify the layout. This visualization will help figure out if the clusters have good connectivity, and we can also compare different variations clusters with each run with different parameters.

I would love to learn more about Amazon's DSSTNE. I have experience in using Deep Learning platforms like Tensor Flow, and Keras. But I feel recommendation algorithms for sparse data using DSSTNE might be a learning opportunity.

3. In a classification setting, given a dataset of labeled examples and a machine-learning model you're trying to fit, describe a strategy to detect and prevent over fitting.

   Answer: One simple strategy I use is to plot the accuracy of the training and testing data on the same graph as below. Initially, one can see that both training and testing accuracy increase. However, if the training accuracy keeps increasing and the testing accuracy starts going down, it is a good sign that the learning model is not generalizing for the unknown data.



   This is because it is learning more specific and complicated definitions for the training set, which do not work with the testing set.

   A good approach to prevent over fitting here would be to stop the training after the X number of iterations, where after X iterations training and testing accuracy diverges. Of course there are other strategies – cross validation, regularization, etc. that should be used in combination with this approach to avoid such over fitting.

4. Your team is designing the next generation user experience for your flagship 3D modeling tool. Specifically, you have been tasked with implementing a smart context menu that learns from a modeler's usage of menu options and shows the ones that would be most beneficial. E.g. I often use **Edit** > **Surface** > **Smooth Surface**, and wish I could just right click and there would be a **Smooth Surface** option just like **Cut**, **Copy** and **Paste**. Note that not all commands make sense in all contexts, for instance I need to have a surface selected to smooth it. How would you go about designing a learning system/agent to enable this behavior?

   Answer: It seems from your framing of the problem e.g. in "for instance, I need to have a surface selected to smooth it" that a good candidate for the context is the previous performed action. Actually, there would be two ways to represent context – one is the state of the workbench/app, and the second would be the action/s you performed recently. So using the previous action/s as a context makes sense.

   Now, we need to create a model to predict the next action from the previous actions. It might be a good idea to start with a Markov assumption. That is, your next action only depends on your most recent action (and the ones before it are unnecessary). Of course, this assumption can be relaxed with the domain knowledge of what is the general length of common action sequences, but let's ignore for now. Then, we can have a simple 2 node Markov-net – the first node is the variable for most frequent action, and the second node (dependent on the first) can be the next action. We can learn the prior and tables using data of sequence of actions taken pairwise (some standard Bayes-net learning algorithms/APIs can be used here).

   We can see if this approach works by testing for accuracy using some train/test data split and by beta testing with users. If not, we can try using n-recent actions (where n is the length of action sequence from domain knowledge) or use a state of the model (e.g. how many objects are there, how many objects selected, etc.)

   This might also be a good idea to use for Alexa, where we can present proactive information when the actions of the users strongly predict the next action or information request. Like, everyone who asks for the news in the morning also asks for traffic updates.

5. Give an example of a situation where regularization is necessary for learning a good model. How about one where regularization doesn't make sense?

   Answer: In Machine Learning algorithms, often the algorithms can run away and create really specialized definitions for the classes, or in the case of regression – fit a complex curve. In such cases, it is important to make sure that the model doesn't over fit. One of the techniques used for this is regularization, where the loss used to update weights of the model has an additional normalization term. For example,

   *Loss = argmin((Pi-Ai) | for I = 1 to n) + lambda * N(w)*

   Here the normalization function can be any standard loss function as L1 or L2 loss.

   Now, there's no one a place where regularization is necessary. Regularization can be helpful to reduce over fitting in most algorithms needing weights trying to optimize the loss function.

   Regularization is not always needed. In cases where you have a large dataset and few features, regularization may result in the model becoming too general and lose its complexity. In such cases, we can make the model as complex as possible by avoiding regularization, which will help the model learn complex definitions of classes/conditions for regressions.

6. Your neighborhood grocery store would like to give targeted coupons to its customers, ones that are likely to be useful to them. Given that you can access the purchase history of each customer and catalog of store items, how would you design a system that suggests which coupons they should be given? Can you measure how well the system is performing?

   Answer: This is a classic clustering problem. Since the product purchase history might be very sparse (since products may have slight variations, brands have similar products, etc.) it might not be a good idea to cluster directly on purchased data.

   So, we first identify attributes about the person based on the products they buy. For example, someone who buys baby diapers, will definitely be have a baby; the average cost of groceries can help estimate the household expenses. To do this, we can use domain experts to create a Naïve-Bayes classifier – that given the product categories from the product catalog, we can infer attributes of the person. We could use a frequency threshold to count only the products brought frequently. Another attribute we can use is time of day/day of the week to get interesting categories (e.g. personally, I like to buy late night dessert). Third, and most important is having some household identification method – here a club card – like Safeway stores in Silicon Valley – is very useful as people in the same household would tend to have the same card. It would be important to phase out old data – for example, babies grow, and over multiple years, they will not need diapers. Now, by using these strategies we can create our data ready for clustering. Each training instance will be a household (club-card id) and the features would be the attributes identified for the person.

   Then, we can use a clustering technique like k-means clustering to find the different clusters. But these clusters aren't enough to recommend purchases. By using associative rule mining, we can find the product category dependencies between the clusters. For example "Baby diaper buying households also buy baby food" or "households that buy organic food also buy organic baby food". These association rules can then be used as basics for coupon recommendation.

   We then consider the business purpose of the coupon recommendation – e.g. increase sales, increase profit and increase customer satisfaction. Perhaps there is a company willing to sponsor a campaign for their product category – For example, California Pizza Kitchen might want to target people who might be likely to buy frozen pizza. Or the store might want to switch people to a more profitable brand.

   To identify how well the system is performing, we can perform an A/B test where the control group is randomly awarded the coupon and the cluster is assigned coupon based on the association rules. We can track the use of the coupon with bar codes etc. to see if the user used the coupon, and identify the success rate of the campaign.

   Finally, the UX of the couponing is important. Currently, I get coupons when I finish billing so that I can buy it next time. But, perhaps a better UX might be a push notification or a quick kiosk when I enter so that I will be more inclined to purchase items.

7. If you were hired for your machine learning position starting today, how do you see your role evolving over the next year? What are your long-term career goals, and how does this position help you achieve them?

   Answer: First of all, I would be thrilled to work with the Alexa team. I have been an early user and have seen Alexa evolve to the awesome assistant it is now. I also understand the underlying technology having worked on similar Agent systems in my day job. I think this team would be a great fit where I can contribute my expertise in developing novel algorithms for improved interaction in the next year or two. With access to usage data, I feel like I can actively improve the UX with the help of personalization to improve individual customer satisfaction.

   One thing I would love to do at the Alexa team is to strengthen my skillset in Agent technologies. One of those is Natural Language Processing and Understanding (NLP/NLU). I think Alexa has one of the best voice recognition systems in the world. And as Alexa's skillsets evolve, the need for natural language understanding will be stronger. I think that working at the Alexa team will give me the unique learning opportunity to learn these technologies in a way that I can also contribute to. Since I am passionate about both learning new skills and making impact with the product UX, I think this will be a great opportunity to learn about NLP/NLU.

   Over the long run, I want to be able to lead the team of applied researchers to expand Alexa to more domains than the home. I am passionate about education and I think that one of the roles Alexa can naturally expand to is tutoring kids at home. I think that with the help of the Alexa team, I can definitely make the impactful applications I look forward to building.