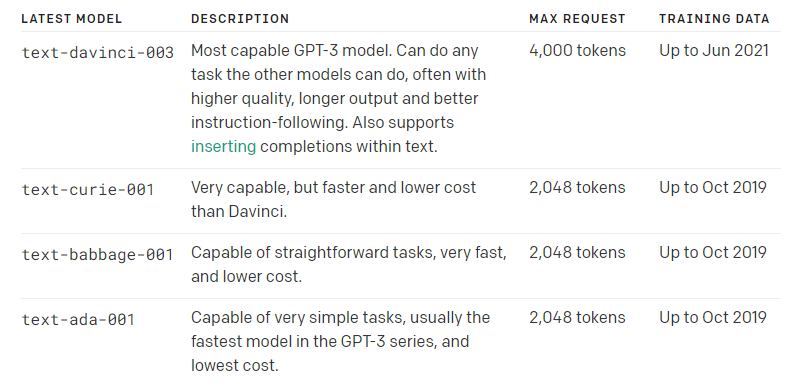
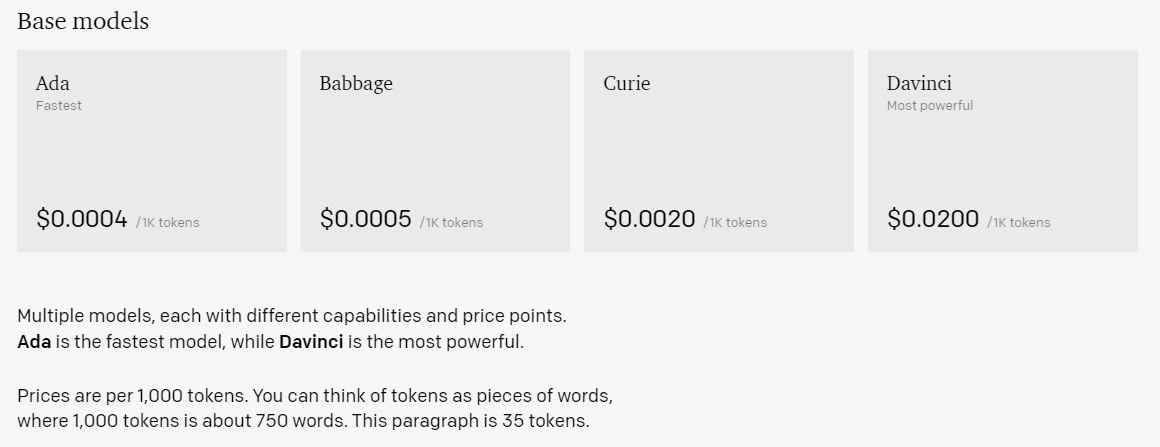
**# # # About Alexa Skills Development**

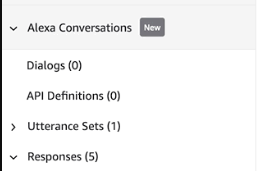
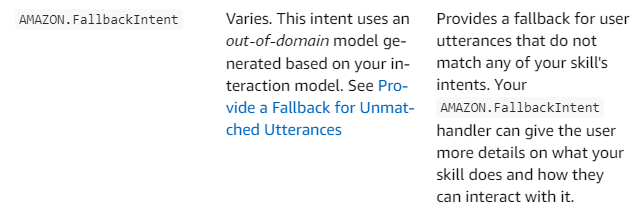
**Notes and questions:**

* It is fine for passive conversations, where Alexa guides with complex statements or questions but expecting simple answers.
* Is there a way to make Alexa **wait for an answer**?
* Is it possible to make a call to One AI for **emotion recognition** on the messages?  
  \* <https://www.oneai.com/skill/emotions>  
  \* The pricing (<https://www.oneai.com/pricing>).
* How **Alexa conversations** work?  
  \* <https://developer.amazon.com/en-US/docs/alexa/conversations/about-alexa-conversations.html>, <https://developer.amazon.com/en-US/docs/alexa/conversations/alexa-conversations-guided-development.html>
* I think what could really give Alexa Skills power is the use of **external services**, to send the recognized text and retrieve an answer.  
  \* I could use OpenAI API -> <https://beta.openai.com/docs/quickstart>, the only problem is that I’d require to pay.
  + Models (<https://beta.openai.com/docs/models/gpt-3>).  
    
  + Pricing (<https://openai.com/api/pricing/>).  
      
    \* Prices are per 1,000 tokens. You can think of tokens as pieces of words, where 1,000 tokens is about 750 words. This paragraph is 35 tokens.  
    \* Accordingly to Google a women say 20,000 words per day in average, it would be $8 mxn of processing.
* Is it possible for Alexa to listen for certain commands while it is speaking? Like “Stop please”, not having to say “Alexa”.
* How feasible is to **design a conversation around a single topic** as a tree? Is it possible to cover an entire space of possibilities?
  + The Akinator model of making questions to make smaller the scope (more specific) and at the end give an answer.
    - It would be fine for Alexa to ask for semantic context, just single words interactions, yes/no and subjects. And like Akinator guess what the person wants.
  + The cooking recipes Skill would require to wait instead of saying all the elements in the list. “Alexa continue” should be the trigger to continue with the recipe, or just wait for the person to say the word “continue” or “stop”.
  + Is it possible to use all the Alexa features in the skills?  
    \* <https://www.hubino.com/resources/how-to-integrate-amazon-alexa-with-google-search/>
    - Internet search.
    - Messaging.
    - Timers.
    - Reminders.
    - Lists.
    - Maps.

\* Looks like it can be done with AMAZON.SearchQuery (<https://developer.amazon.com/en-US/blogs/alexa/alexa-skills-kit/2018/02/enhance-speech-recognition-of-your-alexa-skills-with-phrase-slots-and-amazon-searchquery>).

* + I’d be fine to facilitate internet searches with a little bit of conversation.  
    \* I don’t know what Alexa is expecting from me, and when it says “I don’t get it” that’s frustrating.
    - Alexa, I want to search something.
    - Ok, tell me the topic in one word.
    - <Topic>.
    - Etc, etc.
  + …
* About voice-based **game design** (<http://www.independent-software.com/coding-an-amazon-alexa-skill-tips-and-tricks-for-game-development.html>).
  + After Alexa has finished her response text and other audio, the **player is required to answer within 8 seconds**.
  + After that, **Alexa will remind the player** that she’s waiting for a response by speaking a reprompt (an additional bit of text you can specify in your code).
  + Alexa will then wait **another 8 seconds**.
  + If she receives no response within that time, **she’ll end the skill** without a message (except a soft “ding” sound).

**Questions:**

* Is it possible to make HTTP requests from a NodeJS AWS Lambda function?  
  **R.** Yes, using *async* and *wait*.
* How to initialize another intent from Alexa Skill?   
  **R.** With delegate functions.
  + …
* How does [Alexa conversations](https://developer.amazon.com/en-US/docs/alexa/conversations/about-alexa-conversations.html) work?  
  **R.** It is another way to define interactions, an entire new menu in the developer console. ***We define conversations and hand over them info collection*** from the Skill (<https://developer.amazon.com/en-US/docs/alexa/conversations/hand-off-dialog-management.html>).  
  
  + The *FallbackIntent* seems relevant.  
    
  + **Examples…**  
    \* <https://github.com/alexa-samples/intro-to-alexa-conversations-demo>  
    \* <https://github.com/alexa-samples/alexa-conversations-pizza-reference-skill/blob/main/index.js>  
    \* <https://github.com/alexa-samples/skill-sample-nodejs-alexa-conversations-weather-bot>  
    \* <https://github.com/alexa-samples/skill-sample-nodejs-alexa-conversations-pet-match/tree/master/final>  
    \* <https://developer.amazon.com/en-US/docs/alexa/workshops/build-multi-turn-skills/create-skill/index.html>

**Ideas:**

* It would be fine to create a YouTube channel for this Alexa device, and program it to present topics (<https://www.youtube.com/@Blogxia>).
* Select a topic and talk about it, “Alexa Conversation Series”. Like an specific Lego or a place visited. The idea is to provide the experience of hearing an passionate person.
  + Ask after every end of commentary if the user is still interested on that.
  + Define multiple [yes/no] | [substantive/verb/adjective selection list] **question structures** to evaluate if the user is still interested in the conversation.
  + I can define each commentary as a set of subject (s) and predicate (verbs, adjectives), an idea composed of certain elements, so that I can fill the question structures dynamically.
* It could be a good idea to have a conversation cloud (a lot of curious data about an specific topic) and based on the emotions reflected on the response/reaction of the user decide to follow to other piece of conversation on the same topic.  
  \* <https://www.oneai.com/pricing>
  + User chooses a topic.
  + Alexa gives a piece of conversation, and asks, do you perceive it interesting?, does that sound funny?, how does this make you feel?
  + Based on response Alexa decides to change topic, say “just one more” or ask to the user to change to an aleatory topic “would you prefer to hear about \_\_\_\_?”.  
    \* Start + Topic + End  
     (Start = “Just one more” | “”)  
     (End = “What’s your opinion?” | “Is your opinion positive?”)  
    \* Question
  + Interest gauge metric.  
    \* Based on the interest of the user in the topic Alexa decides to continue with the topic, if it goes too low, Alexa can change the topic or to ask the user for other topic.
  + If Alexa does not understand:  
    \* Alexa won’t attempt to guess (annoying), or it will try to continue the conversation, say I didn’t get you or repeat what the user said as a dialog (when less than 5 words where said).
    - A) Repeats what the user said as a question.
    - B) Variations of “I didn’t get you”.
* Role adventures (as if an art piece would being painted in front of you, or like seeing a place using your hands).  
  “Are you interested on going forward to the river?” -> No -> “You know, there’s a rock in front of you”
  + Describe a situation as a collection of subjects, verbs and adjectives.
  + On the “I don’t understand” handler analyze what the user said and look for the elements of the collection.
    - Verb + Subject = Action over Subject
    - Etc.
* Other use case could be to program routines that call skills during the day to ask for conversation (to never forget the skills), proactivity
* It seems there are enterprises invested on developing for Alexa.  
  \* <https://cedextech.com/alexa-skill-development/>

**Considering integration with OpenAI:**

* It should be possible to create **applications with a single purpose**, guiding the AI to respond in certain manner, **injecting context** to the interactions (**e.g.** asking the OpenAI on every interaction to respond as a pirate).
* Probably **publicity** will be inserted in OpenAI powered applications using this kind of context injection, using a similar model than Google (which detects a specific keyword and throws advertisements as the first results). This could be done by asking the OpenAI to say something good about an specific product when the particular keyword is mentioned.