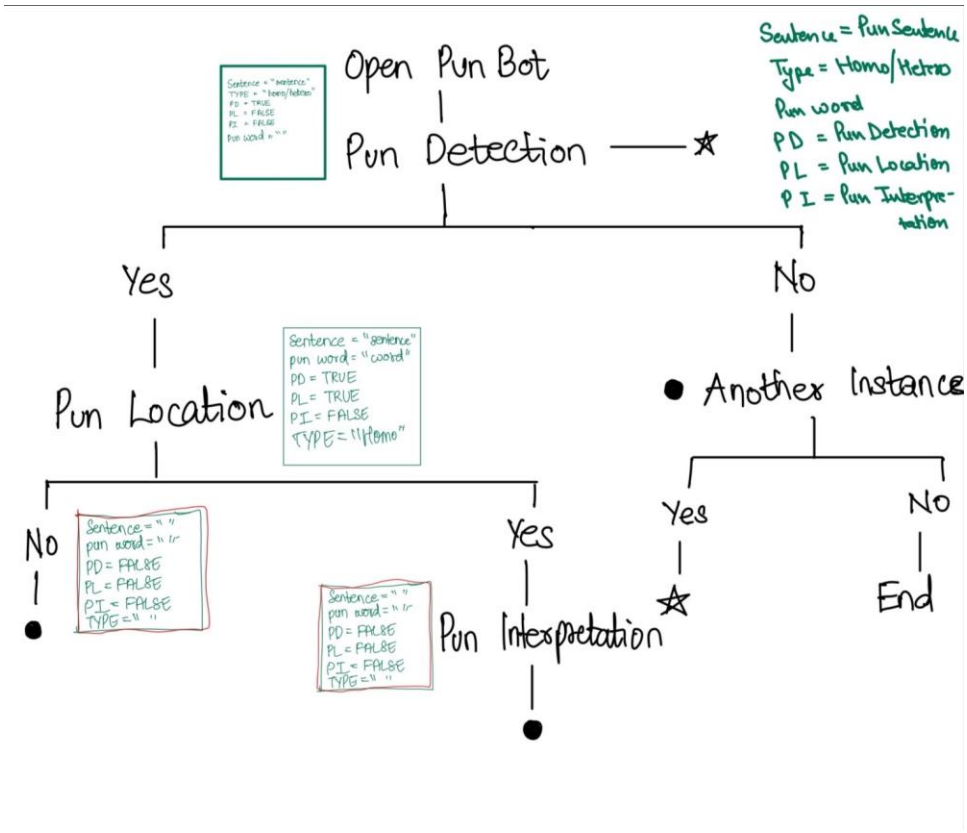
CNIT 519 NLT

Project Report 3

Conversational AI for Pun Detection, Interpretation and Location

[Alexa System Design]



In this system, we managed conversations using one global Intent called PunMeaning. We created multiple handlers to handle different stages of conversations. We have used session attributes to keep track of conversation state and data, and helper functions (handlers) to check if the session attributes indicate whether the pun has been detected/located/interpreted. We have created 3 session attributes namely sentence, punWord, punType to store input sentence, pun-word, pun-type respectively. Based on the state of these session attributes, Alexa decides the response. If the user wants to continue to the next step Alexa proceeds to the next intent, and if the user doesn't want it, Alexa goes back to the initial stage of the conversation by calling the first intent. If the user chooses to go back at any stage, we clear all the values from the session storage.

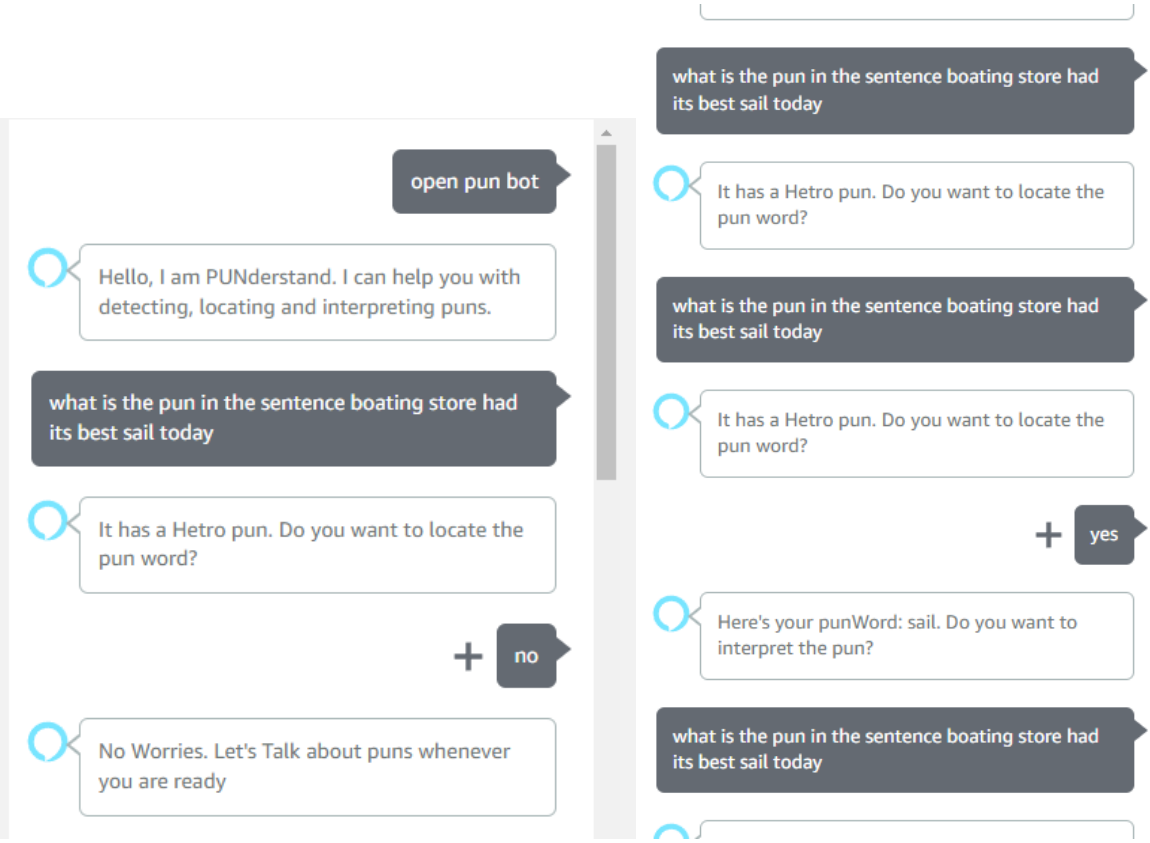
We have built a flask application that calls pun detection/location/interpretation code based on the intent of the request received from Alexa. We used ngrok to expose flask application hosted on local server running on your machine to the internet. Thus, the endpoint url provided by the ngrok can be used by Alexa to make various calls to the flask application.

Since we should handle homographic/heterographic puns differently, we added a classification code to classify homographic/heterographic puns, using the T5 classifier. After that, we used the pun detection/location/interpretation code we built for projects 1 and 2 in our flask application.

Challenges faced integrating Alexa and flask application are

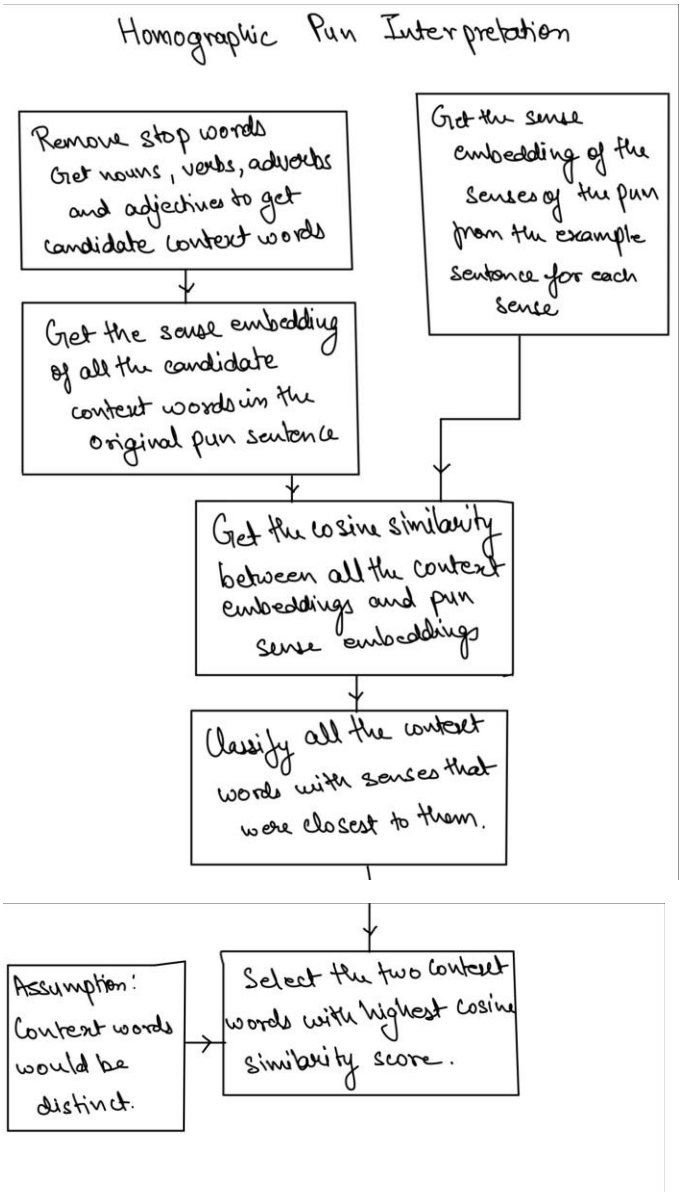
1. As Alexa has a timeout for 8sec, it became very vital for us to obtain the responses from our flask application for pun detection, location and interpretation code within 8 seconds. However, we found the pun location code we planned to use needed more time and hence, we switched the pun location code to T5.
2. Before jumping to T5, we have tried the following optimization:
   Built a different flask application for pun location and have used ngrok to obtain a public endpoint URL for pun location to reduce the computation time. However, we found that we cannot host two ngrok url's on local at once. Hence, we shifted our approach to run pun location on local server and send the response back to our first flask application which uses ngrok to host the application online. However, this approach increased the computation time because multiple reroutes of the request.

The conversation flow is as follows: User opens pun bot by invocation word 'pun bot'. After that, the user provides a sentence with the utterance 'what is the pun in the sentence {sentence}.' We can use other utterances such as 'Identify the pun in the {sentence}', and 'Find the pun in the {sentence}'. This invokes Alexa to run a homo/hetero pun classification code, and Alexa tells the user the type of pun and asks if the user wants to locate the pun. If the user says 'yes', it proceeds to the pun location code and tells the user which word is the pun, and asks user whether the user wants to interpret the pun. If the user says 'no', Alexa calls back to the first intent, and waits for the user to give another sentence. It goes the same with the interpretation.



[Improvement]

We tried to improve upon the **homographic pun interpretation** as follows:

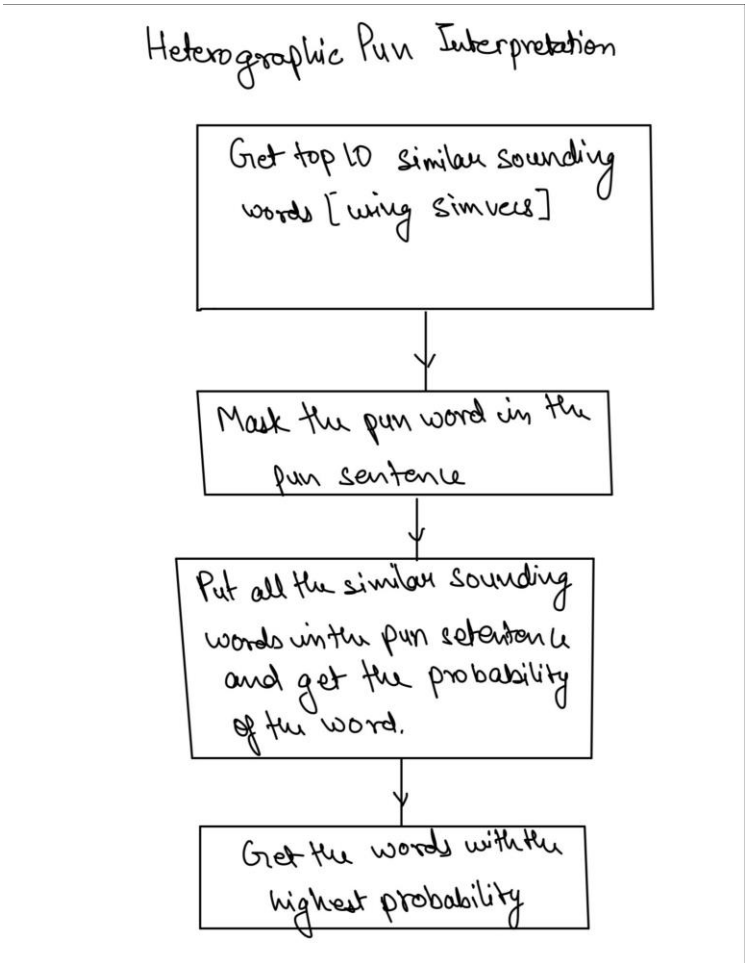


We tried to **improve heterographic pun interpretation** using 2 approaches.

1. Ipa phonetic based cost calculation: We calculate the sound similarity between words using ipa based similarity metrics. We use cost matrix provided by [Ghio, A., Lalain, M., Giusti, L., Fredouille, C., & Woisard, V. (2020, May). How to compare automatically two phonological strings: Application to intelligibility measurement in the case of atypical speech. In 12th conference on language resources and evaluation (LREC 2020) (pp. 1682-1687). Firstly, we get all words from the WordNet corpus and get ipa phonetics from the original word and all words in the corpus. Based on ipa phonetics and similarity score metrics, we calculate the similarity score between the original word and all words in WordNet. We also calculate the threshold value for the similarity score between the

source word and target word using the Semeval dataset. After that, we select the candidate words from the WordNet word list that have a similarity score higher than the threshold value. If we can't find the ipa phonetic from metrics, we use average score value. Since we couldn't find the metrics that provide similarity scores for all ipa phonetics, too many words were selected as candidates. Also, when the length of the original word and candidate word is different, similarity score didn't fully reflect the phonetics.

2. Masking model approach: To get the similar sounding words, we referenced a study that introduced an approach inspired by human sound perception to compute phonetic similarity between words. The methodology leverages this metric to construct a continuous vector embedding space, grouping phonetically similar words. "Sharma, R., Dhawan, K., & Pailla, B. (2021). Phonetic word embeddings. *arXiv preprint arXiv:2109.14796*."



- Masking: Mask the identified pun word and get it predicted using BERT (base-uncased) model. If it is different from the original pun word and is a homophone, we got our interpretation right.
- If not, tokenize the words of the sentence using BERT tokenizer.
- Find the token id of the masked pun word identified in the sentence.
- Find the probabilities: Calculate the probability scores for the BERT vocabulary for the masked word outputs. The output will be a probability distribution predicted for the masked word.
- Tokenize the homophonic words found after step 1.
- Find the probability of each homophonic word using the probability distribution
- Return the word with maximum probability. That would be the interpretation of the pun or the second sense of the pun.

[Video demo]

Kindly find the link to our Video Demonstration here Video Demo