SSP-final-report

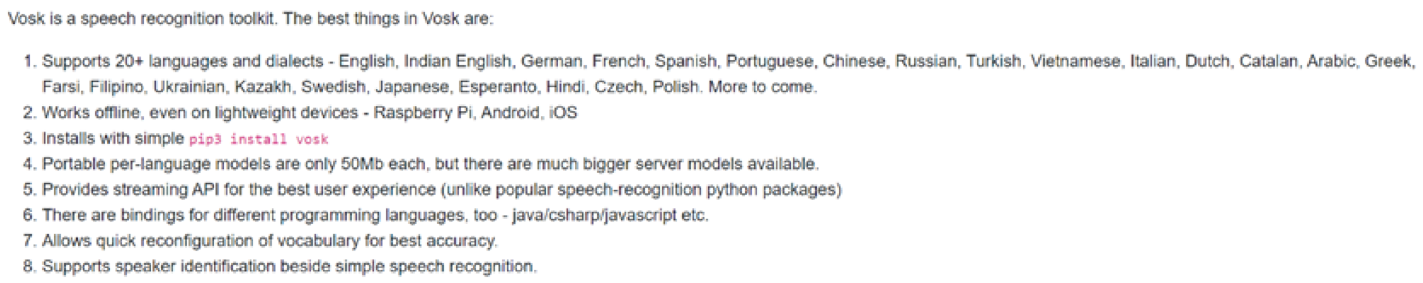
Mianzhi Pan, Chao Chen, Nan Tang

System Architecture

Our system is designed using a main cycle where four parts are speech recognition, adding punctuation marks, generating response and speaking the response is implemented in turn.

We set the wakeup word and the sleep word. Only when the wakeup word is said can we start to talk to the speaker. If sleep word is detected, the speaker will fall asleep until being woken up again. The wakeup word we use are “你好” and “Hello”. The sleep word we use are “再见” and “Goodbye”.

Our system can recognize both Chinese and English language. We use the vosk method in SpeechRecognize package. The parameters are not shared between the recognition of two languages. We can change the language when using the system. Here is the introduction to vosk in its official website[1].



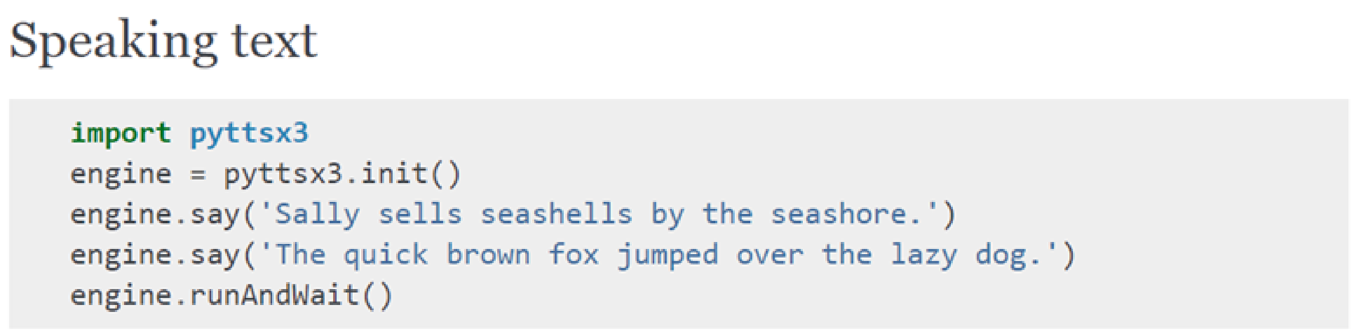
Vosk is a speech recognition toolkit. The best things in Vosk are:

1. Supports 20+ languages and dialects - English, Indian English, German, French, Spanish, Portuguese, Chinese, Russian, Turkish, Vietnamese, Italian, Dutch, Catalan, Arabic, Greek, Farsi, Filipino, Ukrainian, Kazakh, Swedish, Japanese, Esperanto, Hindi, Czech, Polish. More to come.
2. Works offline, even on lightweight devices - Raspberry Pi, Android, iOS
3. Installs with simple `pip3 install vosk`
4. Portable per-language models are only 50Mb each, but there are much bigger server models available.
5. Provides streaming API for the best user experience (unlike popular speech-recognition python packages)
6. There are bindings for different programming languages, too - java/csharp/javascript etc.
7. Allows quick reconfiguration of vocabulary for best accuracy.
8. Supports speaker identification beside simple speech recognition.

Figure 1: Introduction to vosk

Once the words are recognized, we need to add punctuation symbols to organize them into a whole sentence. The function we use is paddlehub with respect to Chinese and Punctuator with respect to English.

When a speech is recognized, we will put it into the NLU model and generate a reply. We use the pyttsx3 package to speak the reply. Below is a brief example of speaking text listed on the official website[2].



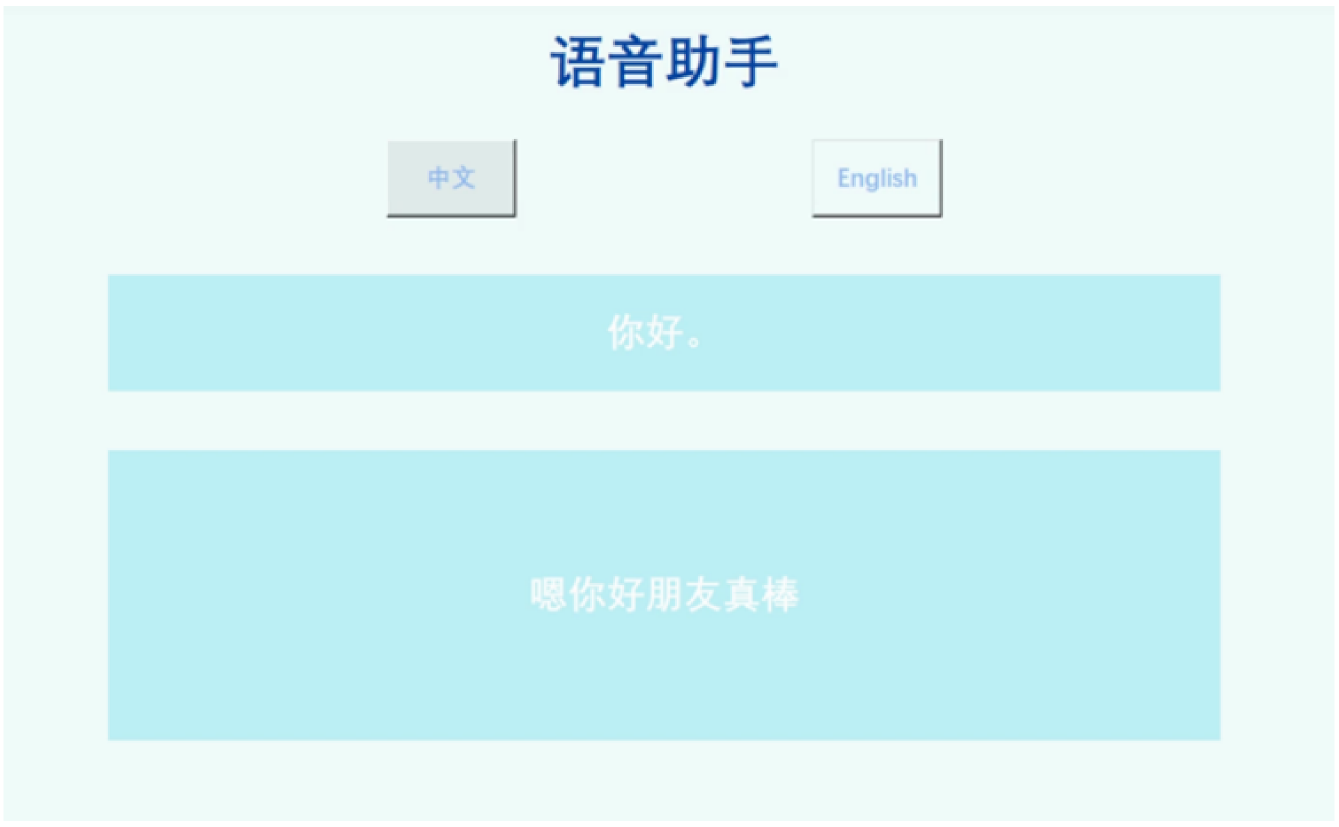
Speaking text

```
import pyttsx3
engine = pyttsx3.init()
engine.say('Sally sells seashells by the seashore.')
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Figure 2: Usage of pyttx3

We design a brief GUI using tkinter which supports both Chinese and English. There are two respective text box to display the speech recognized and the reply. To prevent the contradiction of two threads from speech recognition and command from keyboard, we ban the keyboard from setting the change of language when recognizing speech.

After recognition and output, there will be a 2 second time for the change of language from keyboard.



你好。

嗯你好朋友真棒

Figure 3: A demo of our GUI

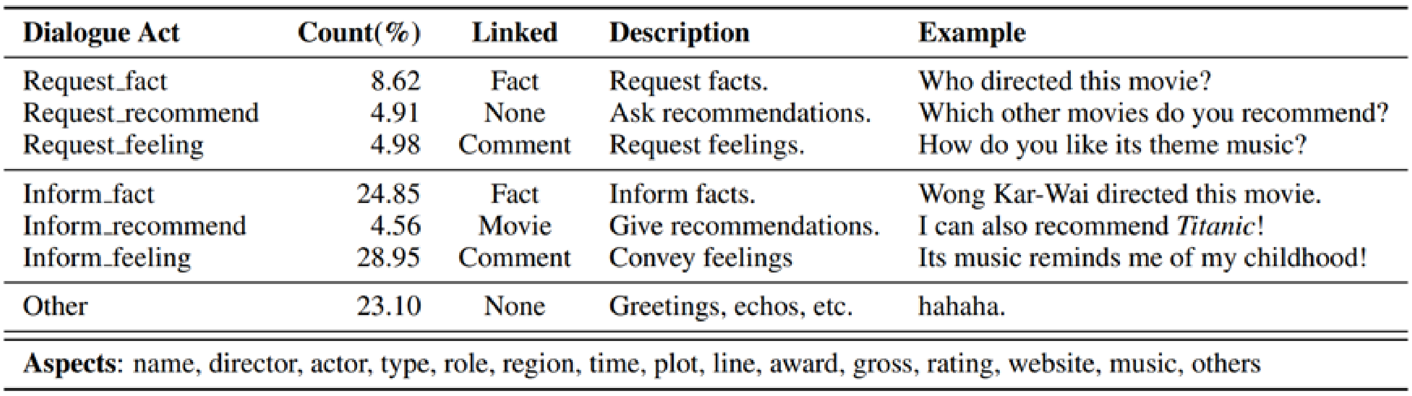
Natural Language Understanding Module

Rasa is one of the most popular frameworks for automated conversation systems. However, in order to generate a response, it has to first recognize users' intent and named entities contained in their message. Such a conventional pipeline is slightly complicated while we prefer an end-to-end conversation model. From this motivation, we construct our model based on GPT2-chitchat and finetune it on a large quantity of human dialogue about the movie.

GPT2 is a transformer-based large language model trained in auto-regressive manner, which can generate the next token given previous tokens. Further trained on 500k general-domain chinese dialogue corpus, GPT2-chitchat can perform simple chat with human. Referring to Moviechats [3], we transfer GPT-chitchat to movie domain. In the following, we detailed our method in four parts: Dataset, model architecture, training process and inference process.

Dataset

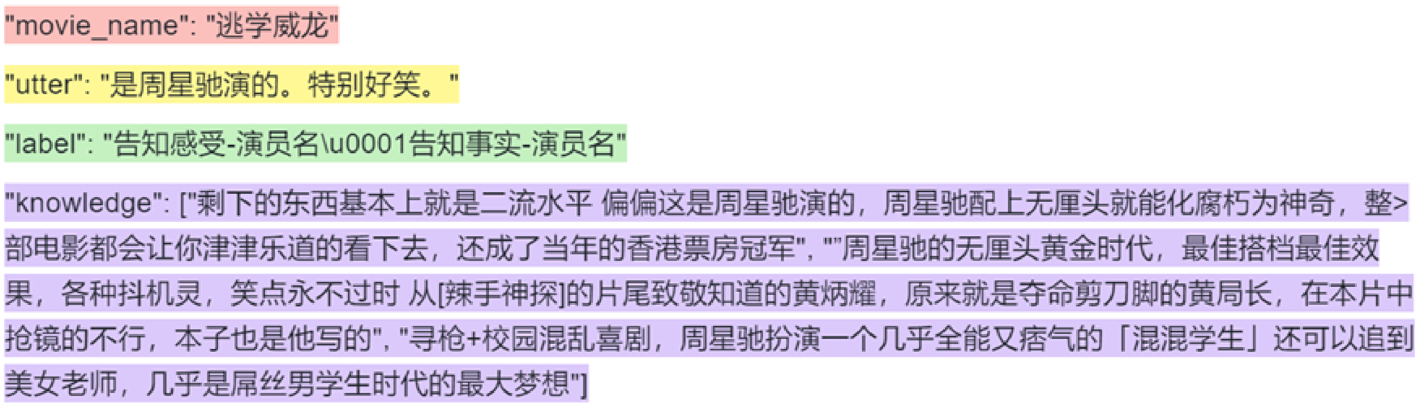
We use the movie-domian chinese dialogue dataset proposed in [3]. This dataset consists of 246,141 dialogues with 3,010,650 turns. Each turn is annotated with dialog acts and entities (Table 1). For utterances labeled with *Inform/Request_fact*, the dataset linked grounded knowledge to them. We show one dialogue turn sample in Figure 1.



Dialogue Act	Count(%)	Linked	Description	Example
Request_fact	8.62	Fact	Request facts.	Who directed this movie?
Request_recommend	4.91	None	Ask recommendations.	Which other movies do you recommend?
Request_feeling	4.98	Comment	Request feelings.	How do you like its theme music?
Inform_fact	24.85	Fact	Inform facts.	Wong Kar-Wai directed this movie.
Inform_recommend	4.56	Movie	Give recommendations.	I can also recommend *Titanic*!
Inform_feeling	28.95	Comment	Convey feelings	Its music reminds me of my childhood!
Other	23.10	None	Greetings, echos, etc.	hahaha.

Aspects: name, director, actor, type, role, region, time, plot, line, award, gross, rating, website, music, others

Table 1: Counts, type of linked knowledge, descriptions and examples of the dialogue acts.



"movie_name": "逃学威龙"

"utter": "是周星驰演的。特别好笑。"

"label": "告知感受-演员名\u0001告知事实-演员名"

"knowledge": ["剩下的东西基本上就是二流水平 偏偏这是周星驰演的，周星驰配上无厘头就能化腐朽为神奇，整> 部电影都会让你津津乐道的看下去，还成了当年的香港票房冠军", ""周星驰的无厘头黄金时代，最佳搭档最佳效果，各种抖机灵，笑点永不过时 从[辣手神探]的片尾致敬知道的黄炳耀，原来就是夺命剪刀脚的黄局长，在本片中抢镜的不行，本子也是他写的", "寻枪+校园混乱喜剧，周星驰扮演一个几乎全能又痞气的「混混学生」还可以追到美女老师，几乎是屌丝男学生时代的最大梦想"]

Figure 4: One dialogue turn sample

Model architecture

The backbone of our model is the GPT-2. In the experiment, we use the interface from huggingface transformer and load the pretrained GPT2-chitchat [4]. As an auto-regressive model, GPT2-chitchat can generate replies word by word given previous utterances.

Training process

GPT2-chitchat is trained on open-domain dialogue corpus. To adapt it to movie domain, we further finetuned it on the Moviechats dataset. During training, history dialogue, users' input, dialogue acts, grounded knowledge and response are concatenated into a sequence, and some special tokens are inserted between different types of text. For example, *[start_know]* and *[end_know]* are concated at the head and tail of the grounded knowledge respectively.

We continue to employ the auto-regressive manner. Given previous tokens, the model predicts the next token and we calculated the cross-entropy loss with regard to the ground-truth label. We take AdamW as the optimizer with a linear-decayed learning-rate schedule and a peak learning rate at 1e-4. We train the model for 8epochs with a batch size of 32. The whole training process takes 25 hours on 2 v100.

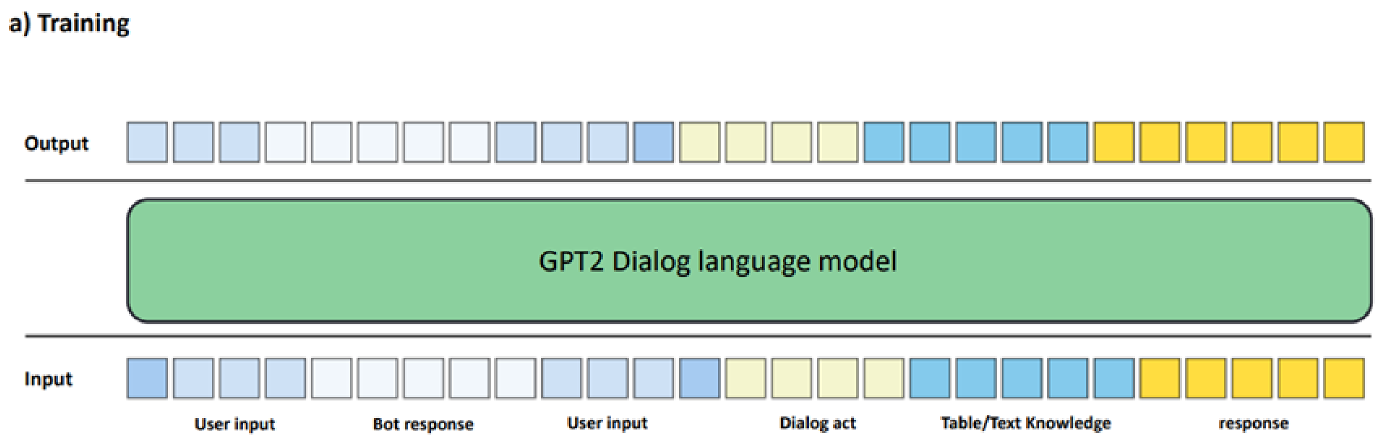


Figure 5: Training process

Inference process

During inference, we input the concatnation of history dialogue and the user's utterance in the current turn to the model. And the model will generate the reply word by word until it comes to a *[end_response]* token or reaches the max length set by the developer. In order to further improve performance, dialog act and grounded knowledge can be appended from external database. This function requires additional modules and we haven't deployed that yet.

b) Inference

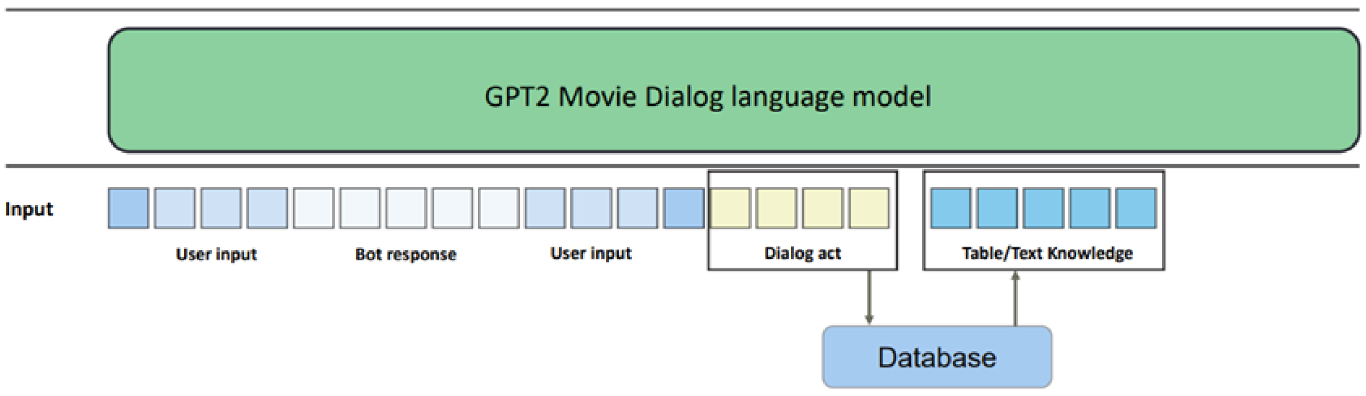


Figure 6: Inference process

Limitations

Our system has some limitations. We list part of them below.

- The time spent on recognition and plus punctuation marks can be reduced.
- The whole system is still single-threaded.
- The vosk model we use is not the best model to recognize speech.
- We have not achieved the conversation of agent in English.
- The performance of our NLU model can be improved.

References

[1] https://alphacephei.com/vosk/index

[2] https://pyttsx3.readthedocs.io/en/latest/engine.html#examples

[3] Su, Hui, et al. "Moviechats: Chat like humans in a closed domain." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

[4] https://github.com/yangjianxin1/GPT2-chitchat

The task of each member:

Mianzhi Pan: Design the NLU model and write the code, write the report and ppt

Chao Chen: Design the speech recognition part, design the whole system and its GUI and write the code

Nan Tang: Design the TTS part, write the report and ppt and speak the report