**Topic**: Spoken Dialog System

# Spoken Dialogue System

## Introduction

Spoken dialogue systems are considered a breed of interfaces that enable humans to communicate with machines naturally and efficiently using a conversational paradigm. [1]

the most prominent component in today’s virtual personal assistants (VPAs). Among these VPAs, Microsoft’s Cortana, Apple’s Siri, Amazon Alexa, Google Assistant, and Facebook’s M, have incorporated SDS modules in various devices, which allow users to speak naturally in order to ﬁnish tasks more eﬃciently. [5]

Uses several components related to known NLP applications [1] [3]:

1. Speech Recognition
2. Speech Synthesis
3. NLU and NLG (natural language understanding, natural language generation)
4. Discourse modeling
5. Dialogue management
6. Dialogue State Tracking
7. Text To Speech

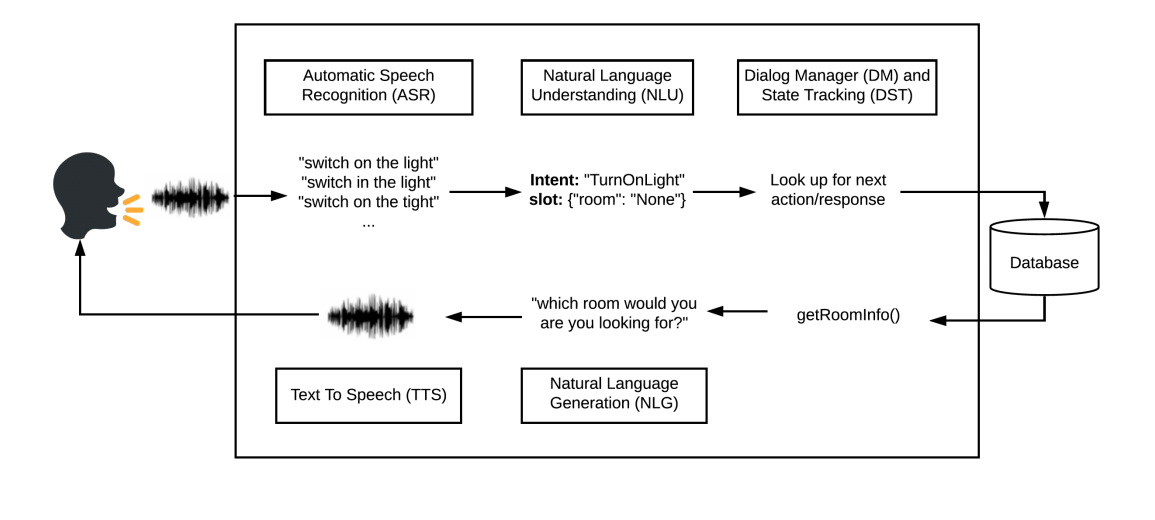


Figure 1: Depiction of a typical dialogue system [3]

widely used mainly in telephone-based information and transaction services. They are highly susceptible to variations in the speech signal caused by environmental and user variations, it is crucial to increase the robustness of these systems. [2]

One of the difficulties posed by SDSs is that, even when the domain of a task is defined, the vocabulary related to this task may not necessarily be clear to the users. it may be impractical or even impossible to provide users with explicit information about supported vocabulary and out-of-vocabulary (OOV) words. Thus, how to correctly reject the OOVs spoken by users, instead of recognizing them as the closest words in the dictionary, presents another difficult but crucial issue. [2]

Most NLG systems in common use employ rules and heuristics and tend to generate rigid and stylized responses without the natural variation of human language. They are also not easily scaled to systems covering multiple domains and languages. [4]

Spoken Dialogue Systems are usually divided into two types — **Task Oriented** and Non-Task Oriented. Task oriented systems are built for specialized use cases such as a banking app using which a user can withdraw or deposit money. Non-Task Oriented Dialogue Systems are built for a general use case, and form a majority of Dialogue Systems including Siri, Alexa. [6]

## Building a Dialogue system

The traditional conversational systems have rather complex and/or modular pipelines. [5]

The advance of deep learning technologies has recently risen the applications of neural models to dialogue modeling. Nevertheless, applying deep learning technologies for building robust and scalable dialogue systems is still a challenging task and an open research area as it requires deeper understanding of the classic pipelines as well as detailed knowledge on the benchmark of the models of the prior work and the recent state-of-the-art work. [5]

a typical Dialogue system looks like this [5]:

Diagram

Description automatically generated

Figure 2: depiction of dialog system (2) [5]

In a Dialogue system what we actually get from the user when he uses our system is either speech or text. If it is speech, we can run it through ASR and get the text(utterance). The first thing you need to do when you get the utterance from the user, is to understand what does the user want, and this is the **intent classification problem**. [5] the intent being a **form** that a user needs to fill in, And each intent has a set of **fields** or so-called slots that must be filled in to execute the user request. [5]

So, let’s say we have two slots here like FROM and TO. And we need a slot tagger to extract slots from the user utterance. Whenever we get the utterance from the user. think of it as a sequence tagging and we can solve it as by sequence tagging tasks using BIO Scheme coding. [5]

Another challenge in dialogue systems is to **handle history information** when next utterance is dependent on previous one. Here we need to add context to our intent classifier and slot tagger. [5]

Another challenge in Dialogue systems is when user suddenly changes context. e.g. Imagine that at first user says, **“Give me directions from L.A.”**, and then we ask, **“Where do you want to go?”** and this time, the user says, **“Forget about it, let’s eat some KFC chicken first.”** So, this is where we need to understand that the intent has changed and we should forget about all the previous slots that we had and all the previous information that we had because we don’t need them anymore. [5] Now the intent classifier gives us navigation find and the category, which is a slot and it has the value of KFC chicken. Then, system makes a query to the database or knowledge base like Yelp and dialog manager understands, “Okay, let’s start a new context and find some KFC chicken.” and the assistant outputs, “Okay, here are nearby KFC chicken places” [5]

**dialog manager** is responsible for two tasks. The first one is dialog state tracking. So we need to understand what the user wanted throughout the conversation and track that state. And also, it does dialog policy managing. E.g. there is a certain policy, which says that, okay, if the state is the following then we need to query some information from the user or request some information from the user or we just inform the user about something. And we can also query backend services like Google Maps or Yelp [5]

when we are ready to give users some information, we use **natural language generation(NLG)** box that outputs the speech for the user so that this is a conversation. [5]

It is important to note these systems still make errors, and the individual blocks are trained separately and then joined together. This means that an NLU block is not trained to handle the errors made by the STT block, and errors propagate through the system. [6]

s

## Speech Recognition

Modern speech recognition systems have come a long way since their ancient counterparts. They can recognize speech from multiple speakers and have enormous vocabularies in numerous languages. [7]

The first component of speech recognition is, of course, speech. (1) Speech must be converted from physical sound to an electrical signal with a microphone, and (2) then to digital data with an analog-to-digital converter. (3) Once digitized, several models can be used to transcribe the audio to text. [7]

Most modern speech recognition systems rely on what is known as **a Hidden Markov Model** (**HMM**). This approach works on the assumption that a speech signal, when viewed on a short enough timescale (say, ten milliseconds), can be reasonably approximated as a stationary process—that is, a process in which statistical properties do not change over time. [7]

In a typical HMM, the speech signal is divided into 10-millisecond fragments. The power spectrum of each fragment, which is essentially a plot of the signal’s power as a function of frequency, is mapped to a vector of real numbers known as cepstral coefficients. The dimension of this vector is usually small—sometimes as low as 10, although more accurate systems may have dimension 32 or more. The final output of the HMM is a sequence of these vectors. [7]

To decode the speech into text, groups of vectors are matched to one or more phonemes—a fundamental unit of speech. This calculation requires training, since the sound of a phoneme varies from speaker to speaker, and even varies from one utterance to another by the same speaker. A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes. [7]

One can imagine that this whole process may be computationally expensive. In many modern speech recognition systems, neural networks are used to simplify the speech signal using techniques for feature transformation and dimensionality reduction before HMM recognition. Voice activity detectors (VADs) are also used to reduce an audio signal to only the portions that are likely to contain speech. This prevents the recognizer from wasting time analyzing unnecessary parts of the signal. [7]

A handful of packages for speech recognition exist on PyPI. A few of them include:

* [apiai](https://pypi.org/project/apiai/)
* [assemblyai](https://pypi.org/project/assemblyai/)
* [google-cloud-speech](https://pypi.org/project/google-cloud-speech/)
* [pocketsphinx](https://pypi.org/project/pocketsphinx/)
* [SpeechRecognition](https://pypi.org/project/SpeechRecognition/)
* [watson-developer-cloud](https://pypi.org/project/watson-developer-cloud/)
* [wit](https://pypi.org/project/wit/)

Some of these packages—such as wit and apiai—offer built-in features, like [natural language processing](https://realpython.com/nltk-nlp-python/) for identifying a speaker’s intent, which go beyond basic speech recognition. Others, like google-cloud-speech, focus solely on speech-to-text conversion.

## Implementation

### Fluent speech corpus

Link: <https://www.kaggle.com/datasets/tommyngx/fluent-speech-corpus>

Dataset contains 97 speakers saying 248 different phrases. The 248 utterances map to 31 unique intents, that are divided into three slots: action, object, and location.

The goal in preparing this dataset was to provide a benchmark for end-to-end spoken language understanding models.

**/wavs/speakers/**: Contains 16kHz single channel .wav files containing the speech audio recorded by the speakers, files are gathered into directories divided by the randomly generated speaker ID each participant was given.

**/data/speakerdemographics.csv**: A CSV file with the demographics for each speaker. These include: self-reported speaking ability, first language spoken, current language used for work or school, gender, and age range.

/data/traindata.csv, /data/validdata.csv, /data/testdata.csv: A csv file describing each audio file in the training set, validation set, and test set, respectively. Each line contains the following information:

* **path** - Path to the .wav file
* **speakerId** - Anonymized alphanumeric code for the speaker of this audio
* **transcript** - The prompt that the speaker was asked to read
* **action** - One of 'change language', 'activate', 'deactivate', 'increase', 'decrease', 'bring'
* **object** - One of 'none', 'music', 'lights', 'volume', 'heat', 'lamp', 'newspaper', 'juice', 'socks', **'shoes'**, 'Chinese', 'Korean', 'English', 'German'
* **location** - One of 'none', 'kitchen', 'bedroom', 'washroom'

### Open Speech Corpora

Link: <https://github.com/coqui-ai/open-speech-corpora>

A list of open speech corpora for Speech Technology research and development.

### Speech Recognition

Link: <https://realpython.com/python-speech-recognition/#how-speech-recognition-works-an-overview>

Notebook: speech\_recognition\_moh\_1.ipynb

[1] <https://link.springer.com/chapter/10.1007/978-3-540-49127-9_35>

[2] <https://www.sciencedirect.com/topics/computer-science/spoken-dialog-system>

- 8.5 SPOKEN DIALOGUE SYSTEMS AND THEIR DIFFICULTIES

[3] <https://arxiv.org/pdf/2001.06463v1.pdf>

- <https://paperswithcode.com/paper/plato-dialogue-system-a-flexible>

[4] <https://paperswithcode.com/paper/semantically-conditioned-lstm-based-natural>

[5] <https://medium.com/@nisar.shah1/introduction-to-dialogue-systems-part-1-475a06ab78ad>

[6] <https://akshatgupta57.medium.com/spoken-language-understanding-on-building-spoken-dialogue-systems-214490b31cc3>

[7] <https://realpython.com/python-speech-recognition/#how-speech-recognition-works-an-overview>

f