Speech-to-CDQL: Context Definition and Query Language from Natural Language for Smart Home Environment

Submitted as Master Dissertation for SIT724

SUBMISSION DATE

T3-2021

Ngoc Dung Huynh

STUDENT ID 219235988

COURSE - Master of Data Science (S777)

Supervised by: Dr. Mohamed Reda Bouadjenek, Dr. Imran Razzak, and Dr. Ali Hassani

Acknowledgement

I would like to thank my main supervisors, **Dr. Mohamed Reda Bouadjenek** and **Dr. Imran Razzak**, who have continuously helped and guided me in the right direction of the project. They have always paid attention and shared their knowledge with me. Working with them, I learned valuable data science skills such as Machine Learning, Deep Learning, Natural Language Processing, Speech Recognition, and Translation Machines. Moreover, I also obtained other essential skills such as critical thinking, problem-solving, and academic writing. I would also like to thank other researchers who made invaluable contributions to the project: Dr. Ali Hassani, Dr. Kevin Lee, Dr. Chetan Arora, and Prof Arkady Zaslavsky. Their feedbacks are crucial to me in improving my project planning and paper writing.

Publications

List of pending publications:

- **Ngoc Dung Huynh**, Mohamed Reda Bouadjenek, Imran Razzak, Kevin Lee, Chetan Arora, Ali Hassani, and Arkady Zaslavsky, Adversarial Attacks on Speech Recognition Systems for Mission-Critical Applications: A Survey.
- **Ngoc Dung Huynh**, Mohamed Reda Bouadjenek, Imran Razzak, Kevin Lee, Chetan Arora, Ali Hassani, and Arkady Zaslavsky, Translating Natural Language Texts to CDQL Expressions: A Smart Home.
- **Ngoc Dung Huynh**, Mohamed Reda Bouadjenek, Imran Razzak, Kevin Lee, Chetan Arora, Ali Hassani, and Arkady Zaslavsky, Jarvis: A Tool to Translate Natural Language Textsto CDQL Expressions in Smart Home Environment.

Abstract

Nowadays, the usage of IoT devices has become very popular. Its research gives rise to many innovative applications in healthcare, communication, and smart homes. Especially, smart homes have been a hot research topic and attracted many researchers in recent years. Smart devices in smart homes such as temperature sensors and smart TVs send collected data to relational databases allowing developers to query them using different query languages such as Context Definition and Query Language (CDQL). However, CDQL requires understanding its formulas for updating and searching data content in the host databases. Therefore, this paper introduces a user interface named Speech-to-CDQL to allow users from different backgrounds to control a smart home using a voice-based interface directly. The model has two components: a speech recognition system (Google speech recognition) and a text-2-CDQL system based on Encoder-Decoder RNN architectures. Three encoder-decoder architectures are implemented for the text-2-CDQL system: basic architecture, Bahdanau attention, and Luong attention. We use the Finite Automata approach to improve the result. The result shows that the accuracy and the Word Error Rate (WER) of the Bahdanau attention model are the best: 93% and 0.02%. The code of the project is located at `https://github.com/parkerhuynh/IEDG`.

Contents

List of Figures

List of Tables

1 Introduction

Since the last century, the usage of IoT devices has been increasing rapidly around the world. There is no single definition for IoT, but it can be considered a system of multiple devices connected to form a network to exchange data [59, 34]. In other words, the responsibility of an IoT system is to connect electronic devices and allow data sharing among them. Electronic devices consist of billions of physical devices, from smartphones to laptops, smart sensors, wearable fitness trackers, and many others. IoT applications can be found in different fields such as healthcare, smart cities, smart homes, and smart factories. These applications allow electronic devices to communicate or interact over the Internet smoothly and can also be managed and controlled when required. As a result, IoT brings many benefits to consumers, businesses, and governmental organizations.

Smart home technology is one of the most popular and practical applications of IoT. As a result, it has been a hot research topic that attracts the attention of both business and government organizations. A smart home is a house that uses the newest technologies and computerization to control different sensors or smart devices in a house. In recent years, smart home service shave also evolved to help make homes smarter. Smart home projects have constantly been creating new services to improve the living standard of inhabitants. The basic services of smart homes based on IoT are to detect changes in the smart home environment via sensors such as temperature and human sensors and to monitor human health via wearable devices. In addition, with the development of natural language processing, the voice-activated control feature allows people to control smart devices such as smart TV and smart lights in their homes by voice commands.

The benefits of smart homes are indisputable. First of all, a smart home brings health-related benefits. For example, air purification can improve the indoor smart-home environment to enhance human health [73]. In addition, wearable devices can monitor human health every moment, detect life-threatening diseases early, and even provide distant medical care when necessary [67]. Secondly, some smart devices can monitor the energy usage of a smart home or even each device in a smart home. As a result, residents can monitor and effectively control their energy consumption leading to carbon emission reduction [22]. Moreover, some smart homes replace traditional energy sources such as coal, oil, and natural gas with renewable energy resources like wind, solar, biomass, and geothermal energy [78]. Therefore, smart home has become an effective method for improving the environment. Finally, smart homes bring financial benefits. By reducing energy consumption, smart homes allow residents to reduce their electric bills. Also, early detection of cancers through wearable devices helps patients significantly lower treatment costs.

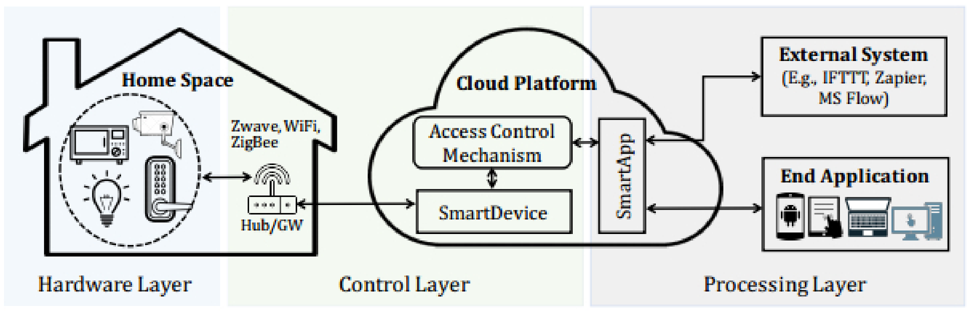


Figure 1: Smart Home System Architecture.

Figure 1 [48] shows a basic smart home system architecture. Firstly, smart devices such as entertainment devices, climate sensors, sockets, doorbells, lights, and even security systems are connected to a network. A cloud platform shares the data from these smart devices to create an IoT system. In some cases, these smart devices can be controlled over the Internet through smartphones, tablets, and laptops. However, if these smart devices come from different products or services, a smart home hub is required to sync the data of these devices. In other words, the smart home hub acts as a third software that allows users to control everything and have the devices from different brands interact with one another. Users can also connect smartphones to a smart home hub to control their homes via their smartphones. Some famous smart home hubs are Home Mini, Google Home, Home Max, Amazon Echo, Samsung SmartThings Hub, and Apple HomePod mini.

Nowadays, the volume of data grows exponentially. The data is from various devices such as smartphones or sensors. Therefore, it is essential to store and analyze this data. Cloud data platforms aim to restore and analyze data. Some famous examples of cloud data platforms are Amazon Web Service, Microsoft Azure, and Google Cloud Platform. CooaS introduced by [30] is defined as a contextual data platform created for operationalizing context-awareness in the IoT domain. This platform allows context (or textual data) providers and context (or textual data) consumers to exchange context or contextual data between IoT entities. Consumers can send query requests to corresponding providers to extract and update context or contextual data about IoT entities from the platform.

Generally, query languages are used to extract, analyze, and update data from cloud data platforms. In fact, there are thousands of query languages, and each query languages have its own syntax to implement. Therefore, these query languages are familiar only to those with programming experience. The popular query languages are Oracle PL/SQL, SPARQL, GraphQL, Oracle PL/SQL, Prisma, and JSON API. SQL is the most popular language used to access most cloud platforms. CDQL Context Definition and Query Language (CDQL) introduced Hassani et al. [30] is considered a query language created for CoaaS. It is used as an incoming request to extract or update context or textual data from CoaaS.

1.1 Aims & Objectives

CDQL is usually difficult to use because of the complex syntax. Additionally, even for an expert, implementing CDQL could be a challenge since it requires an understanding of the database's schema and the roles of the entities in the query. In other words, only someone with query language experience can effectively exploit relational databases using CDQL. Therefore, building natural language interfaces is for relational databases is essential because it enables developers from different backgrounds or consumers easy and direct access to these databases using their voice or text input. Therefore, **the aim of this paper is to propose a natural language interface called Speech-to-CDQL that converts the natural language from speech form to CDQL for controlling a smart home**. In other words, the proposed interface allows all people from different backgrounds to access a smart home database via their speech instead of queries. Some objectives are listed below to achieve the aim of this project:

- Conduct a survey of Speech Recognition systems and text-to-query systems.
- Conduct a simulated IoT system for smart home.
- Use Google Speech Recognition system to convert human speech into text.
- Use different encoder-decoder architectures to create a text-to-CDQL system which converts natural language text into CDQL.
- Use the finite automaton of the CDQL language to improve the result.

1.2 Thesis Organization

The remaining of this thesis is organized as follows:

- **Section 2** presents the background needed as well as the basic concepts used throughout this thesis, including CooaS, CDQL, Finite Automata, Speech Recognition systems, and text-to-query systems.
- **Section 3** provides the comprehensive review of speech recognition systems and text-to-query system. Since most previous research work only focused on SQL, the most popular language, we will focus on reviewing text-to-SQL systems.
- **Section 4** illustrates our proposal IoT system for Smart Home. This system is developed for evaluating our proposal user interface. The scenery is to extract the data from some popular smart devices in a smart home which is generated by an IoT simulator using natural language speech.
- **Section 5** presents our proposal user interface named speech-to-CDQL, which is created by combining two models: a speech recognition system and a text-to-CDQL system. We first introduce the pre-trained speech recognition system and then provide our approaches for the text-to-CDQL system.
- **Section 6** firstly presents the method to generate the data set to train the text-text-CDQL system. After that, this section shows the metrics (Word Error Rate and Accuracy) used to evaluate our models. Thirdly, the implementation details is shown. Finally, the discussion and analysis will be presented.
- **Section 8** concludes the research and shows the limitations as well as our future work.

2 Background

This section presents the background of the main concepts addressed and required in this manuscript. In Section 2.1, we present the basic concept of the textual data platform CooaS. Then, in Section 2.2, we present the background knowledge of CDQL how to create a CDQL request. In Section 2.3 and Section 2.4, we give a background of speech recognition system and text-to-query, respectively. In Section 5.3 we introduce Finite Automata, which is used to improve our model.

2.1 CoaaS

Along with the IoT's development in many fields, the data types are also constantly evolving and heterogeneous across areas. The data from smart devices is updated every moment, leading to a large amount of data that can be analyzed to optimize resources. Additionally, the need for data storage, extraction, and analysis is increasing rapidly. As a result, using relational databases is becoming popular in storing, managing, updating, and extracting data for IoT development as they fulfill those requirements [56]. In relational databases (known as Relational Database Management System (RDBMS)), data is stored in tables, and rows are referred to as records. Popular examples of relational databases include CoaaS [30].

Cooas (Known as the Context-as-a-Service platform) is defined as a context management middle software created for operationalizing context-awareness in the IoT domain. This platform allows context (or textual data) providers and context (or textual data) consumers to exchange context or contextual data between IoT entities. Consumers can send query requests to corresponding providers to extract and update context or contextual data about IoT entities from the platform. Context updates consist of updates of the IoT entities' states and are handled by CoaaS to monitor situations. Figure 2 [30] illustrates the entire view of the CoaaS platform.

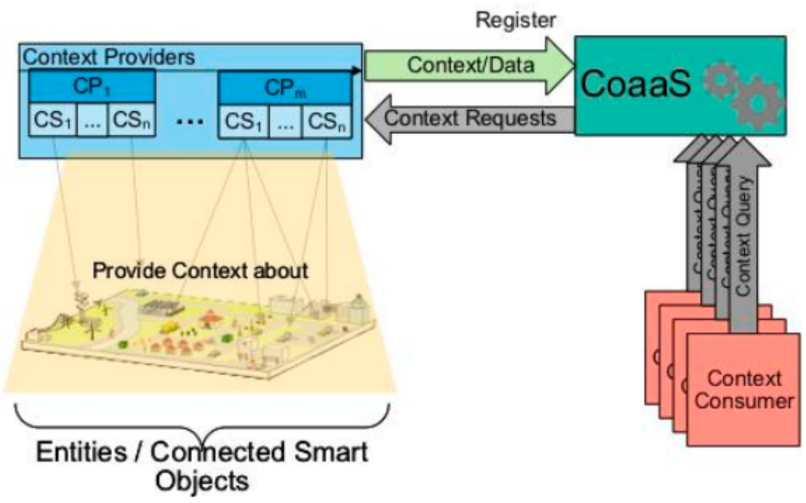


Figure 2: CooaS Big Picture.

CooaS has four components: Communication and Security Manager, Context Storage Management System, Context Reasoning Engine, and Context Query Engine. The architecture of Cooas is presented in Figure 3 [30]. Firstly, the communication and security manager processes incoming and outgoing messages containing context queries, context updates, and context responses. Additionally, this component also ensures the system's safety from network attacks by checking incoming messages and authenticating them if they are safe. Secondly, The context storage management system saves the descriptions of context service, caches contextual information, and processes context delivery. Thirdly, the context reasoning engine infers situations from raw data collected by sensors. Lastly, the context reasoning engine parses the incoming queries, generates the query execution, and returns the query result or updates the context.

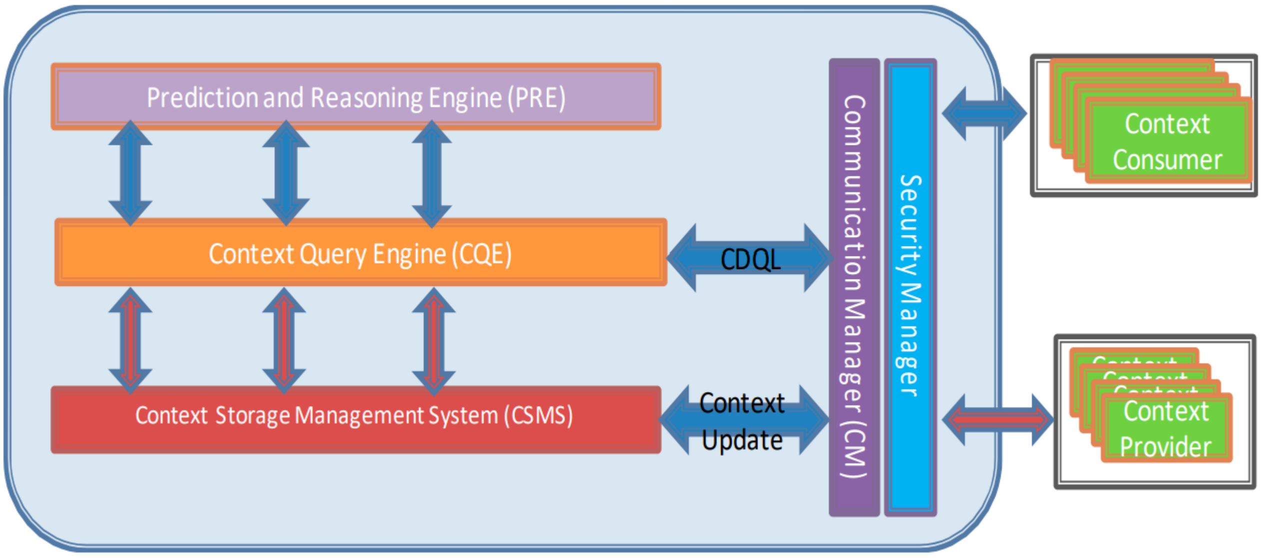

Prediction and Reasoning Engine (PRE)
Context Query Engine (CQE)
CDQL
Context Storage Management System (CSMS)
Context Update
Communication Manager (CM)
Security Manager
Context Consumer
Context Provider

Figure 3: CooaS's Architecture.

2.2 Context Definition and Query Language (CDQL)

Relational databases have been used to store a vast amount of today's data and applied in many areas such as healthcare [32], financial markets [9], and customer relations management [51]. However, these databases require an understanding of query languages identified as programming languages for updating and searching data content from the host databases. Popular query languages are SQL, AQL, SPARQL, Datalog, and DMX. However, these query languages are ineffective when working with contextual data. Context Definition and Query Language (CDQL) introduced Hassani et al. [30] is considered a query language created for CoaaS. It is used as an incoming request to extract or update context or textual data from CoaaS (Figure 3). This approach has two main components: Context Query Language (CQL) and Context Definition Language (CDL). The former is a query language that asserts contextual information requirements without understanding details of the data structures, while the latter aims to define the situation and level content.

2.2.1 Context Query Language (CQL)

CQL needs to have three mandatory clauses to operate successfully: PREFIX, SELECT, and DEFINE:

- **PREFIX:** The purpose of the PREFIX clause is to define any abbreviations from URLs that we want to reference in a query. In other words, the PREFIX clause is used as a simplified word to replace a long URL name. Figure 4 show two examples of PREFIX clause.

```
prefix dogont:http://elite.polito.it/ontologies/dogont.owl,
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns.
```

Figure 4: Example of PREFIX clause.

- **SELECT:** The SELECT clause defines only those records that fulfill a specified condition. It is used to choose the specific results of a query. Each query's result can be a set of values where each value in the set can be illustrated as either a CONTEXT-ATTRIBUTE or a FUNCTION-CALL.
  - A CONTEXT-ATTRIBUTE: describes a feature of an entity. This component concludes two other parts: CONTEXT-ENTITY-ID and IDENTIFIER. CONTEXT-ENTITY-ID defines the IoT entity that the context attributes will be queried from. The value for CONTEXT-ENTITY-ID can be any of the entities known to the IoT system. IDENTIFIER decides the interesting information extracted, such as temperature and humidity for light level. Users also extract all entity data by using an asterisk (*) wildcard.
  - FUNCTION-CALL: allows querying high-level context requests. This component enables users to extract the IOT entity by using their custom function.

```
select (targetCarpark.*, distance(targetCarpark, destinationLocation.geo , walking)).
```

Figure 5: Example of SELECT clause.

Figure 5 presents an example of the SELECT clause. The first part of this example *targetCarpark.** is an example of CONTEXT-ATTRIBUTE. This part presents all the available attributes of an entity with $ID = targetCarpark$. On the other hand, the second part *distance(targetCarpark, destinationLocation.geo ,' walking')* is an example of FUNCTION-CALL. This custom function calculates the distance between the selected car parks and the driver's destination.

- **DEFINE:** is the final mandatory clause that allows querying contextual information from multiple entities by determining the entities that are related in a query. Each entity is represented by four parts: CONTEXT-ENTITY-ID, ENTITY-TYPE, CONDITION, and SORT-BY. CONTEXT-ENTITY-ID defines the entity's name, while ENTITY-TYPE determines the type of entity. CONDITION gives strict conditions to filter out unnecessary entities from all available entities. Finally, SORT-BY is utilized to sort the extracted entities in descending or ascending order. Figure 6 shows an example of the DEFINE clause.

```
define
entity device is from smarthome where device.hasState.ison = true and device.devicename
= "Temperature" and device.isin.buildingenvironment = "living room".
sort by asc
```

Figure 6: Example of DEFINE clause.

Apart from that, it also has two other optional clauses, namely SUBSCRIPTION and SET.

- **SUBSCRIPTION:** monitors IoT entities, discovers changes in situations, and adjusts to them automatically. In addition, SUBSCRIPTION allows users to get periodic changes or subscribe to a specific problem. For example, SUBSCRIPTION monitors the humidity of one specific room every 10 minutes. The result will be sent to users every 10 minutes if the humidity of the room changes. SUBSCRIPTION concludes either a WHEN or EVERY statement.
  - EVERY: is used to extract context within a certain periodic period of time. The value of this statement is set under ISO 8601 standard in the format P[n]Y[n]M[n]DT[n]H[n]M[n]S[n]MS. For example, *P1Y2M3DT4H5M6S7MS* is represented for an interval of *1 year, 2 months, 3 days, 4 hours, 7 minutes, 15 seconds, and 20 milliseconds*. Figure 7 presents an example of the EVERY clause.
  - WHEN: is used to create a situation-based query that identifies and monitors exciting situations. For example, it is assumed that a user found a parking spot and went there. Unfortunately, the area is in an unavailable case during the trip. This statement allows the user to find another location as soon as possible to park the car. Figure 8 presents an example of the WHEN clause.

```
select (destinationLocation.weather.airTemperature))
every pT5M
define
```

Figure 7: Example of EVERY clause.

```
select (targetCarpark.*)
when isFull(selectedParking, car, event) > 0.5
define
```

Figure 8: Example of WHEN clause.

- **SET:** This clause has 3 components: CALLBACK, META, and OUTPUT.
    - CALLBACK: is used to determine how the result of the query is sent to customers.
    - META: is utilized to set a minimum value of data.
    - OUTPUT: is used to define the structure of the result that will be sent back to customers in JSON/XML.

Five basic clauses of CDL is presented in Figure 9. Figure 10 shows a simple example of CDQL and Figure 11 presents a complex CDQL.

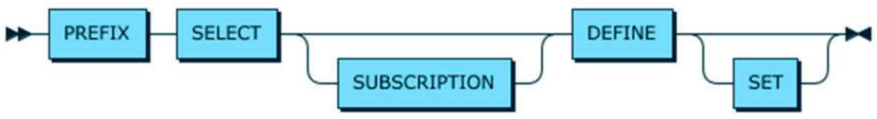
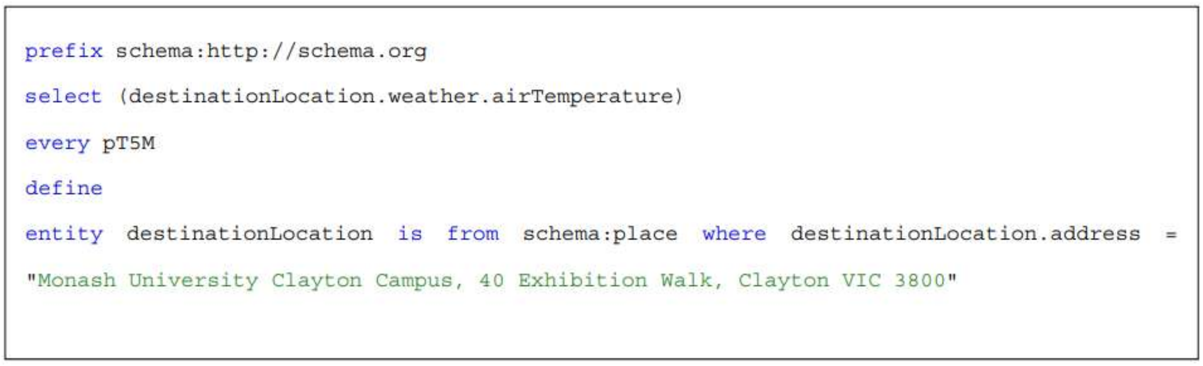


Figure 9: Basic CQL Architecture.

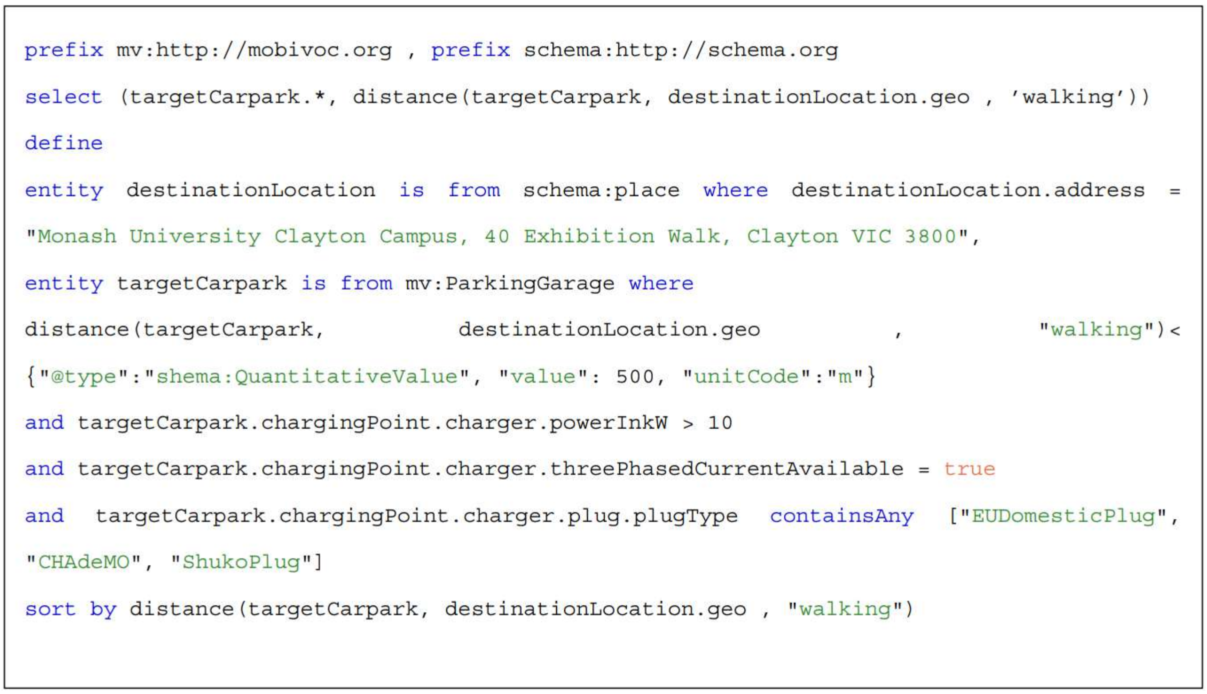
2.2.2 Context Definition Language (CDL)

Although the built-in functions support common cases well, it is still necessary if it is possible to create functions specific to particular use cases. Therefore, CDL allows users to create their custom functions by providing the CREATE-FUNCTION, defining aggregation and reasoning functions. Two important CDL functions are aggregation function and situation functions.



```
prefix schema:http://schema.org
select (destinationLocation.weather.airTemperature)
every pT5M
define
entity  destinationLocation  is  from  schema:place  where  destinationLocation.address  =
"Monash University Clayton Campus, 40 Exhibition Walk, Clayton VIC 3800"
```

Figure 10: Example of simple CDQL.



```
prefix mv:http://mobivoc.org , prefix schema:http://schema.org
select (targetCarpark.*, distance(targetCarpark, destinationLocation.geo , 'walking'))
define
entity  destinationLocation  is  from  schema:place  where  destinationLocation.address  =
"Monash University Clayton Campus, 40 Exhibition Walk, Clayton VIC 3800",
entity targetCarpark is from mv:ParkingGarage where
distance(targetCarpark,          destinationLocation.geo          ,          "walking")<
{"@type":"shema:QuantitativeValue", "value": 500, "unitCode":"m"}
and targetCarpark.chargingPoint.charger.powerInkW > 10
and targetCarpark.chargingPoint.charger.threePhasedCurrentAvailable = true
and  targetCarpark.chargingPoint.charger.plug.plugType  containsAny  ["EUDomesticPlug",
"CHAdeMO", "ShukoPlug"]
sort by distance(targetCarpark, destinationLocation.geo , "walking")
```

Figure 11: Example of complex CDQL.

2.3 Speech Recognition System

Recent technological advancements in Artificial Intelligence (AI) have revived the concept of direct communication with computer systems. Most organizations are swallowed by technological hype to quest for personalized, efficient, and convenient customer interaction. A simple but efficient solution is Conversational User Interfaces (CUI) [45, 5, 24]. CUIs for mission-critical applications is the next great leap forward, with dual-way interaction between machine and human, helping end-users solve their problems with voice support.

In the early days, Hidden Markov Model (HMM) [23] was the primary tool for speech recognition. However, the development of this traditional method has saturated in terms of both latency and accuracy. With Deep Learning (DL) advancements [40, 39], a neural network replaces some of the standard system components. In recent years, the trend has been to design an end-to-end neural network and leverage a massive amount of data to improve the accuracy of speech conversational systems. Deep learning has been applied to develop such conversational systems for mission-critical applications, such as recognizing the transcription of medical speech in healthcare [21]. Speech recognition is the core component of a voice-based conversational system, aiming to convert a speech from an audio form into a textual format for easier downstream processing. In the end-to-end models, the modules of the traditional system (acoustic model, pronunciation model, and language model) are jointly optimized in a single system. Examples of end-to-end models include CTC-based models [25] and Attention-based models [12, 8, 17, 16, 14].

2.4 Text-to-Query System

With the ubiquity of personal devices such as smartphones in the last five years, the amount of data created in the previous decade has increased rapidly. A big part of this data is structured and stored in relational databases to be updated and searched using a query language. However, query languages require time to get familiar with complex syntaxes. Meanwhile, most people do not have time to learn and understand the query languages [76]. Therefore, tools called text-to-query systems can convert human body language into increasingly popular query languages. As a result, people from different backgrounds can access these databases easily via their natural human language.

In the early days, Syntactic Parsing was the primary technique for the text-to-query system. In this approach, the whole natural human text is divided into words. These words are then processed and analyzed in the grammatical categories to define the rule for each word in the input text. Finally, depending on the rule of each word, the words will be mapped to the corresponding SQL formula. This approach seems to be a tedious task, and it also obtains a low accuracy as well as spends a lot of time to covert text to query languages. However, in the last decade, huge efforts in Natural Language Processing have bridged the gap between people and computers. In other words, the core of this concept is to convert natural human language into query languages automatically. This approach, also called Semantic Parsing, uses deep learning models such as RNN, LSTM, or encoder-decoder architecture to translate natural language text into a logical form.

3 Literature Review

This section is divided into two subsections: Speech Recognition System and Text-to-Query System. The former aims to review popular approaches to converting speech into text. Meanwhile, the latter gives an overview of text-to-query systems.

3.1 Speech Recognition System

Integrating voice recognition technologies with conversational systems opens the door to many real-world applications. Speech recognition provides a natural interface for human interaction and is becoming a widely adopted input method for various smart devices. It allows machines to process human voices resembling an average human. Automatic speech recognition is considered one of the most complex systems which involve mathematics, statistics, and linguistics.

A speech recognition system operates in two steps: feature extraction and text decoder. The purpose of feature extraction is to convert a speech signal to a predetermined number of frequency components. It is also called Front-end Processing and is implemented by transforming the human speech waveform into parametric representation for subsequent processing and analysis. There are different methods for feature extraction such as Mel Frequency Cepstral Coefficients (MFCC) [52], Perceptual Linear Prediction (PLP) [31], linear predictive coding (LPC) [52], Discrete wavelet transform (DWT)[11], Linear prediction cepstral coefficient (LPCC) [2], Fast Fourier Transform (FFT) [42] and Line spectral frequencies (LSF) [50]. Among these techniques, MFCC is the most popular method for feature extraction [4].

Text decoding aims to convert the audio features into words by searching the most likely sequence of words. Unfortunately, searching all possible sequences is inefficient. Thus this task is mainly carried out by machine learning systems such as the Viterbi algorithm (limit the number of searches and find the optimal path in polynomial time) or deep learning. Earlier, HMM was favored for speech recognition due to its simplicity which is replaced by deep learning due to its flexibility and predicting power. However, unlike uni-model deep learning or

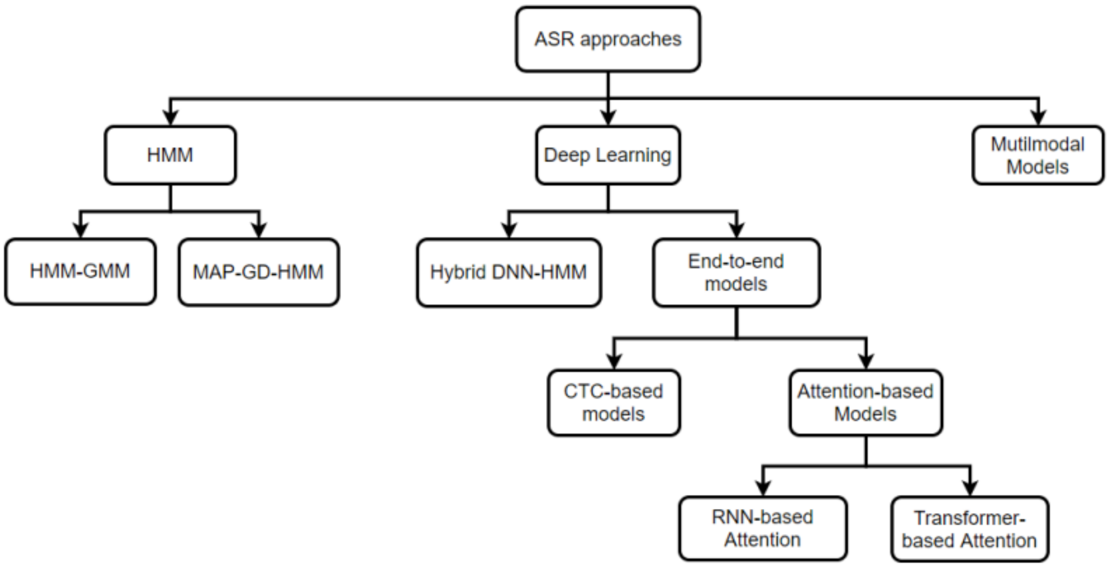


Figure 12: Taxonomy of Speech Recognition.

HMM, multimodal systems (audio plus virtual features) have recently shown better performance. Popular algorithms for text decoding in speech recognition systems are classified in Figure 12.

3.1.1 Hidden Markov Model (HMM)

Searching all possible sequences is inefficient. HMM is a statistical approach to estimating hidden information from visual signals. A speech recognition system is modeled as a Markov process with unknown parameters, signified by the known observable parameters [23]. Integration of Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) outperformed HMM and GMM based systems. The decoder in GMM-HMM consists of 3 separately trained modules: acoustic, pronunciation, and language. The acoustic module takes the feature extraction input to predict phonemes using a Gaussian Mixture Model. The pronunciation module is an HMM that maps the phonemes expected at the acoustic module to word sequence. The final module is the language module, such as an n-gram language model, which aims to estimate the probabilities of the next word based on the preceding words [23, 35, 66]. GMM-HMM achieves about 21.2% of the word error rate (WER) on the Switchboard data and 35.4% of WER on the CallHome dataset [66]. A complex speech recognition system involves a large vocabulary as word boundaries are challenging to identify. HMM can be less complex and have fewer states for a small vocabulary (i.e., one acoustic model per state). However, speech recognition for extensive and continuous vocabulary requires context-dependent modeling, significantly increasing the number of states. Clustering based on a decision tree can be used to share acoustic models.

GMMs may not be the optimal choice for modeling the distribution of the features of speech data. Another approach to model the distribution is via the maximum of a posterior (MAP) of the generalized Dirichlet (GD)-based HMM (MAP-GD-HMM) [3]. The model MAP-GD-HMM shows better performance compared to HMM-GMM on the TIMIT dataset.

3.1.2 Deep Learning

Deep learning is well known for its applications in image recognition, but its use in speech recognition is gaining more and more attention. There are many variations of deep learning architecture for speech recognition because of its flexibility and predicting power. Some popular architectures based on Deep Learning for speech recognition are Hybrid DNN-HMM and CTC-based models.

Hybrid HMM and Deep Neural Network (NN-HMM) models are the earliest Deep Learning approach where a DNN replaces the acoustic module. The results show that DNN-HMMs can obtain a lower WER than HMM-GMMs [46, 1]. However, Long Short Term Memory neural networks (LSTMs) [27, 41, 79] and Time Delay Neural Networks (TDNNs) [55, 53] have replaced DNNs and GMMs because these models have shown the improvement of performance in speech recognition. In traditional speech recognition systems such as HMM-based models and DNN-HMM, three modules (acoustic, pronunciation, and language models) are separately trained, requiring longer training time and larger memory size. For instance, an n-gram language model requires several gigabytes [35]. In recent years, the trend of building speech recognition systems has been to develop an end-to-end speech recognition system based on neural networks. In these models, the modules of the traditional system (acoustic model, pronunciation model, and language model) are trained jointly in a single system. In other words, the network can map the input speech sequence to a sequence (seq2seq) of either graphemes, characters, or words. This approach can resolve the limitation of the traditional system in which the overall system may not be optimal even though individual components are separately optimal. End-to-end architecture systems can be classified into Connectionist Temporal Classification (CTC)-based models and Attention-based models.

CTC Based models introduced by [25] is considered the first architecture of end-to-end models. Speech recognition system based on CTC, also called RNN-CTC models, includes Recurrent Neural Networks (RNNs), which are the main component for sequence processing, and a CTC loss function that augments the set of target labels with an additional "blank" symbol.

Deep Speech is the most popular CTC-based speech recognition system [28, 6]. The acoustic and pronunciation modules are jointly trained in the RNN-CTC model. However, it still has a limitation: it cannot learn the language module because of the conditional independence assumptions. This system uses Markov assumptions to resolve the seq2seq problem with a forward-backward algorithm [26] followed by Convolutional Neural Networks (CNNs) or RNN-CTC systems [75, 60].

Generally, CTC-based models still require a language model, and attention-based models are motivated to overcome this limitation of CTC models. Dissimilar to CTC-based models, attention-based models directly train traditional speech recognition systems (acoustic, pronunciation, and language modules) since they do not have conditional-independence assumptions. Therefore, it does not require a language model, leading to a huge amount of memory being saved [12, 8]. Attention-based models consist of two parts: encoder and decoder (also called attention-based decoder). The purpose of the encoder is to map the acoustic input into a higher representation. Meanwhile, the decoder part takes the output from the encoder part and generates the output symbol based on the full sequence of the preceding predictions. There are 2 major architectures of attention-based models: RNN-based encoder-decoders architecture [12, 8, 17, 16, 14] and Transformer-based encoder architecture [64, 36, 20].

Recently, RNN based encoder-decoder architecture has been widely used for speech recognition. The encoder, a recurrent network, takes the input followed by the decoder, another RNN, looks at the output of the last hidden state from the encoder. LSTMs are usually used as RNNs in the encoder and decoder modules [16]. However, this architecture has a drawback: the decoder only sees the last hidden state from the encoder. Therefore, the previous states may not be appropriately used in long-range dependencies, leading to the loss of information. To resolve this issue, the decoder has access to all the hidden states, and relevant input steps in attention-based architectures [12, 8, 17, 14].

Attention-based RNN is considered one of the best ways to capture the time dependencies in sequences. However, it usually takes a long time to train due to its sequential nature because hidden states are created one step at a time. In addition, the backpropagation algorithm for weight updating is also time-consuming. Therefore, a new approach called a Transformer network is designed to resolve the limitation of the RNN-based architectures. This system is still an encoder-decoder architecture, but it does not use RNNs to model time series data. Therefore, this system does not rely on backpropagation to update the weights.

The encoder of a typical Transformer network consists of 6 identical layers. Each includes two layers: a multi-head self-attention module and a position-wise feed-forward neural network.

The decoder also has six identical layers, each containing an additional layer called a multi-head self-attention function. This layer allows Transformer networks to perform the attention to multiple steps simultaneously. As a result, the training time of a Transformer network can reduce effectively compared to RNN-based architectures [64]. The first application of Transformer Network for speech recognition is Speech-Transformer [20]. The author only added a few CNNs before inputting the features to the Transformer network to reduce the difference in the dimensions of the input and output sequences. As a result, it achieved about 10.9% of WER on the Wall Street Journal (WSJ) speech recognition dataset. This architecture is improved by integrating CTC loss into Speech-Transformer. As a result, the WER reduces to 4.5% on WSJ. The authors point out that the Transformer Network obtains higher accuracy and shorter training time than the RNN-based Attention model [36].

3.1.3 Mutilmodal models

The integration of multiple modalities while talking helps people understand each other better. In other words, combining the context with our conversation helps to convey more accurately what we mean. Generally, a unimodal speech recognition system is trained on the data that contains speech as the input and a sequence of words as the labels. However, the input to a multimodal model includes both speech and image. There are two main steps in this approach. The first step is to extract the audio and visual features. The second step is to integrate these features into one vector to train the model [61]. For example, Mamyrbayev et al. [47] combine human voices and images of the lip, face, and gestures for speech recognition. As a result, multimodal models can improve the accuracy compared to unimodal models. Therefore, multimodal speech recognition models that combine audio and visual modalities have become more prevalent in speech and natural language processing communities.

Like general speech recognition, there are several different methods to extract audio features, such as MFCC. For image feature extraction, the general method is using CNNs such as ResNet [61, 10, 62], and Region Convolutional Neural Network (RCNN) [29]. The most important aspect of this system is combining the audio features and images features. This process can be implemented in both encoder and decoder. For encoder feature fusion, we can apply the technique called Shift Adaptation [62]. For decoder feature fusion, the most popular method uses Early Decoder Fusion to integrate these features [61, 62, 29]. Apart from that, Weighted Early Decoder Fusion, Middle Decoder Fusion, and Hierarchical Attention over Features can also be used to combine the audio and image features [62]. The result showed that multimodal systems obtain lower WER compared to unimodal models, especially under noise conditions [61, 10, 62, 29, 47]. Weighted Early Decoder Fusion achieves the lowest WER, increasing

1.40% on the augmented dataset. Finally, Hierarchical Attention over features achieves the best recovery rate of masked words, with an improvement of 4% over an attention-based model [62].

speech recognition system	**Dataset**	**WER (%)**	**Accuracy(%)**
HMM-GMM [66]	Switchboard	21.2	N/A
HMM-GMM [66]	CallHome	36.4	N/A
HMM-GMM [3]	TIMIT	N/A	50
MAP-GD-HMM [3]	TIMIT	N/A	93.33
DNN-HMM [66]	Switchboard	14.2	N/A
DNN-HMM [66]	CallHome	25.7	N/A
LSTM-HMM [57]	Switchboard	7.2	N/A
LSTM-HMM [57]	CallHome	12.7	N/A
TDNN-HMM [55]	Switchboard	9.2	N/A
TDNN-HMM [55]	CallHome	17.3	N/A
LSTM-CTC (Bigram Language Model) [26]	Wall Street Journal	13.5	N/A
LSTM-CTC (No linguistic information) [26]	Wall Street Journal	27.3	N/A
LSTM-CTC (Trigram language model) [26]	Wall Street Journal	8.2	N/A
LSTM-CTC[26]	Wall Street Journal	8.2	N/A
CNN-LSTM-TCT [75]	Wall Street Journal	10.5	N/A
LSTM-CTC [13]	Medical Dataset	20.1	
Attention RNN-based [12]	Wall Street Journal	10.3	N/A
Attention RNN-based [21]	Medical Dataset	15.4	N/A
Attention RNN-based [13]	Medical Dataset	18.3	
Transfomer Network [20]	Wall Street Journal	10.9	N/A
Transfomer Network [36]	Wall Street Journal	4.5	N/A
Unimodal Model (Attention RNN-based) [62]	Flickr8K	13.7	N/A
Multimodal [62]	Flickr8K	13.4	N/A

Table 1: Comparison of speech recognition systems.

3.2 Text-to-Query System

This aspect has only attracted the attention of researchers in recent years. There are many query languages to access different relational databases. Structured Query Language (SQL) is considered the most popular language. Most relational databases use SQL to extract and update data. Therefore, most researches focus on the conversion from text into SQL.

There are two main approaches to text-to-query: Syntactic Parsing and Semantic Parser (using Deep Learning).

In **Syntactic Parsing**, the idea is to analyze the grammatical categories to define the rule for each word in the input text. Then, depending on the rule of each word, researchers use a syntax tree to map the keywords of the input text to the corresponding SQL formula. Therefore, the input text must follow a strict syntax for proper parsing. Otherwise, the input text cannot be recognized and not translated to SQL. Overall speaking, this approach is not intelligent and requires lots of hand-engineering effort [38, 71].

The core of the **Semantic Parsing** approach is to use Deep Learning models to translate natural language text into a logical form. In other words, we train a sequence-to-sequence network using a vast number of natural language sentences and queries. There are two problems that text-to-query systems need to solve:

- How do text-to-query systems map natural language to column names?
- How to deal with different natural language questions that express the same meaning?

To resolve the first problem, all previous works used the Attention Encoder-Decoder RNN-based model to create a machine translation model [77, 68, 74, 70, 19, 58, 33, 69]. Attention models allow the systems to focus on the query words most relevant to each label during the training. However, general machine translation models that translate a natural language to another natural language only use a natural language as an input. Meanwhile, text-to-query systems combine both natural language and columns of relational datasets as input. As a result, the attention models can focus more on relevant words during training. Table 2 shows some popular text-to-query algorithms.

For the second problem, previous papers used the contextualized word representation of one of the existing pre-trained Natural language models such as BERT [18], RoBERTa [43], XLNET [72], Word2Vec [49], Glove [54] to convert words to embeddings. They are fed to the model and the column's embedding of the database. This solution makes the attention model understand the meaning of the text by using the surrounding text to create a context. As a result, the model can process the same questions that are spoken in so many different ways.

Model	Dataset	Accuracy (%)
Seq2SQL [76]	WikiSQL	59.4
SQLNet [70]	WikiSQL	68.0
TypeSQL [74]	WikiSQL	73.5
Coarse-to-Fine [19]	WikiSQL	78.5
Graph theory method [38]	Arabic language-SQL	68.43
Annotated Seq2seq [68]	WikiSQL	82.2
Robust Text-to-SQL [68]	WikiSQL	83.8
IncSQL [58]	WikiSQL	87.1
SQLizer[71]	MAS, IMDB, YELP	88.0
SQLOVA [33]	WikiSQL	88.7
SQLOVA + EG[33]	WikiSQL	92.1

Table 2: Comparison of text-to-SQL systems.

4 IoT System Architecture for Smart Home

The designed architecture is presented in Figure 13. The architecture of the tool has three main components: the user interface, the database (CooaS platform), and the IoT data generator:

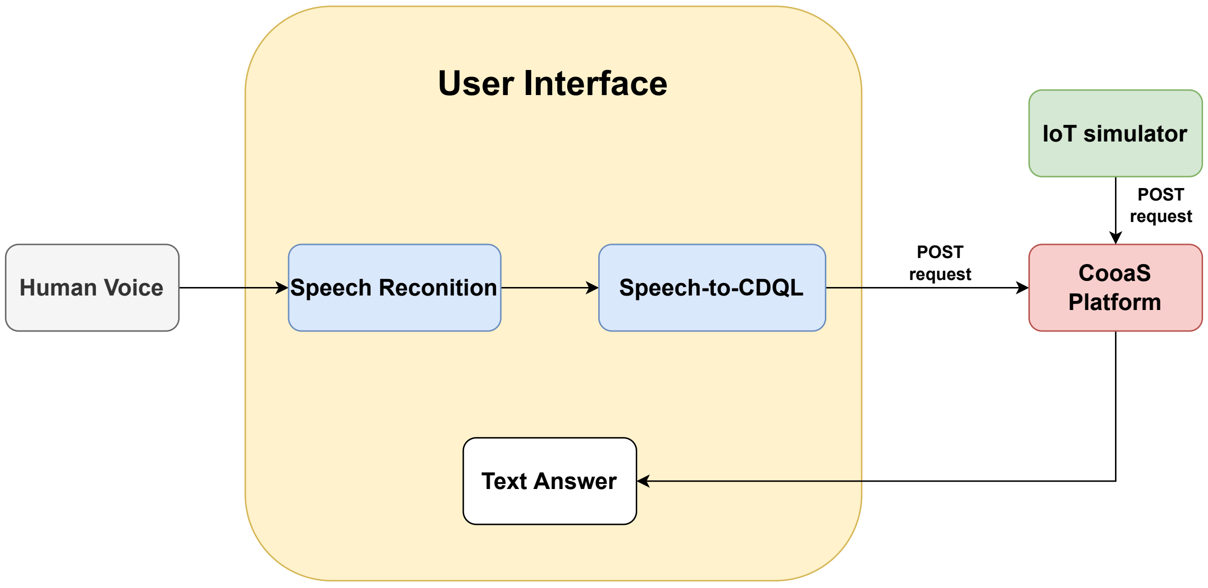


Figure 13: Smart Home User Interface Architecture.

- **User Interface (UI):** This component is the main part of the tool. The responsibility of the UI is to take the human voice question and transform it into the most likely CDQL query. The UI hosts two machine learning systems: speech recognition and text-to-CDQL. Firstly, the speech recognition system takes the human speech request and converts it into the transcript. Secondly, the text-to-CDQL converts the transcript from the speech recognition system and generates the CDQL query result. After that, it will send the output to the CooaS platform via a POST request. Finally, the CooaS platform returns the result that is shown on the UI.

- **CooaS platform:** This component serves two purposes. Firstly, it stores the data generated from the IoT simulator. Secondly, it receives the CDQL query from the UI, executes it, and returns the result to the UI.

- **IoT simulator:** This element acts as a data generator that simulates a smart home environment containing various devices such as temperature sensors, humidity sensors, and smart TV. These sensors are located in various building environments such as living room and bedrooms. The simulated smart home is presented in Figure 14. Smart device-generated data are sent to the platform via POST requests in JSON format that consists of 2 main parts: Attributes and EntityType. The Attributes part contains all entities of smart devices, such as the device's name, state, value, environment building, timestamp, and unit measurement. On the other hand, the EntityType part defines the destination of the raw data in the database. Figure 15 gives the example of data of one temperature sensor in a living room.

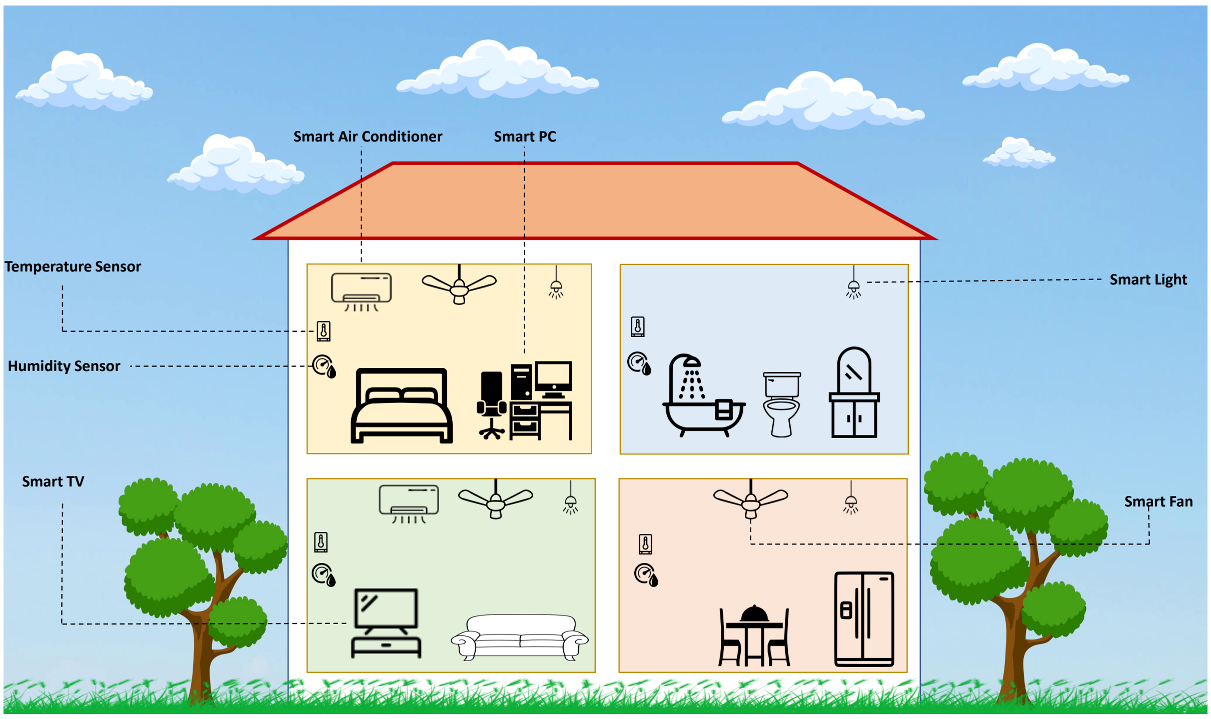


Figure 14: Smart Home.

```
{
Attributes: {"deviceName": "temperature", "hasState": true, "id": "temp 1", "isIn":
"livingroom", "stateValue": 23,"ts": 1639366667892, "unitMeasurement": "degrees celsius"},
EntityType: {"namespace": "http://elite.polito.it/ontologies/dogont.owl", "type":
"smarthome"}
}
```

Figure 15: Data Simulation of Temperature Sensor in the living room

5 Methodology

As mentioned in the above section, the Speech-to-CDQL UI is based on two models to convert human speech to CDQL: speech recognition and text-to-CDQL. Firstly, the speech recognition system takes the human speech request and converts it into the transcript. Secondly, the text-to-CDQL converts the transcript from the speech recognition system and generates the CDQL query result.

5.1 Speech Recognition System

For speech recognition, we use an existing model called *Listen, Attend, and Spell* introduced by Google [12]. This model is created based on the attention encoder-decoder architecture that originates from translation work. *Listen* is the encoder, *Attend* is the attention mechanism, and the Spell is the decoder. The architecture of the model is illustrated in Figure 16.

The encoder *Listener* is created by one or more RNNs aims to take the input sequence of filter bank spectra feature and then encodes them into hidden states stored into a denser representation. The decoder *Spell* built by RNN predicts the word given previous decoder hidden states, the preceding predicted character, and the previous context that is computed by the attention mechanism *Attend*. This model achieves about 10.3% of Word Error Rate. This architecture is elaborated in detail in the next section as it is inspired by the attention translation network [63, 15, 44, 7].

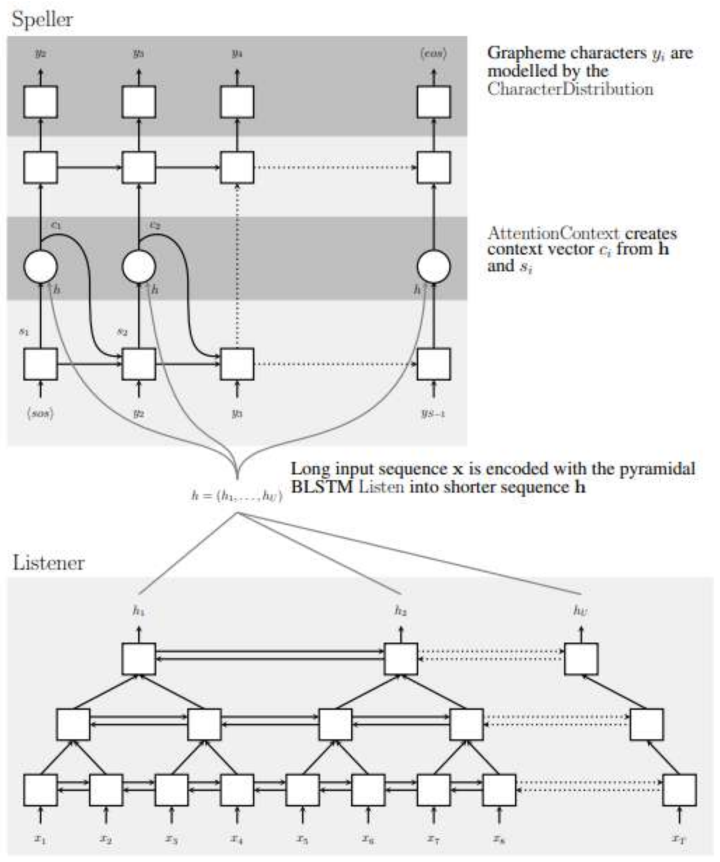

Speller
Grapheme characters y_i are modelled by the CharacterDistribution
AttentionContext creates context vector c_i from h and s_i
Long input sequence x is encoded with the pyramidal BLSTM Listen into shorter sequence h
$h = (h_1, \ldots, h_U)$
Listener

Figure 16: *Listen, Attend, and Spell* Architecture.

5.2 Text-to-CDQL system

In this section, we present our approaches for building a text-to-CDQL system. Section 5.2.1 describes the baseline model, which is created based on a basic encoder-decoder architecture. To improve the baseline model, we use the attention encoder-decoder architecture (known as Attention Neural Network) in Section 5.2.2. We use both Bahdanau's and Luong's attention functions to apply to the attention architecture.

5.2.1 Basic Encoder-Decoder Architecture

The Sequence-to-sequence Encoder-Decoder model was firstly introduced in two research papers [63, 15]. This architecture aims to resolve the sequence-to-sequence nature of machine translation where input sequences differ in length from output sequences. The Encoder-Decoder model has two main parts: Encoder and Decoders. The encoder is an RNN that takes the input

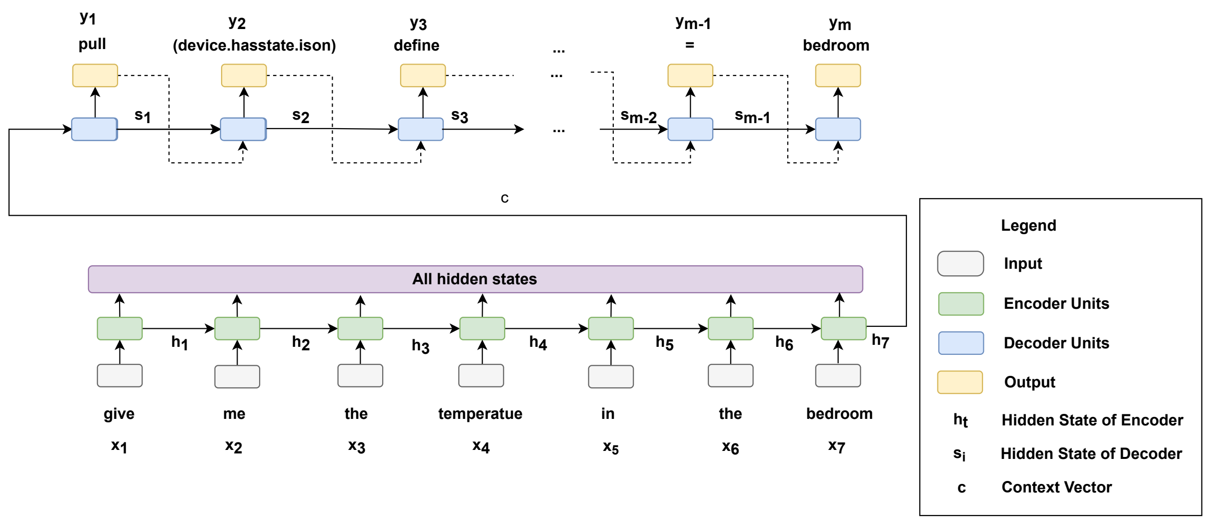


Figure 17: Basic Encoder-Decoder Architecture.

sequence and converts it into hidden states. Consequently, the decoder takes only the encoder's last hidden state and then generates the sequence output.

Firstly, $x = (x_1, x_2, x_3, ..., x_n)$ is assumed as a input sequence vector where n is the length of the input sequence. The encoder that is a Recurrent Neural Network (RNN) such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) takes only one element from the input sequence x and generate one hidden state h_t at a time. That means that the encoder will take n time steps to read the entire input sequence and generate n hidden states. The hidden state h_t is computed using the following formula:

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

Where f is an RNN. After that, the encoder returns the last hidden state as a context vector c that is considered a container consisting of all information of the input sequence:

$$c = h_n \tag{2}$$

$y = (y_1, y_2, y_3, ..., y_m)$ is assumed a target sequence where m is the length of the sequence. The decoder takes the context vector c and all the preceding predicted words $(y_1, y_2, .., y_{t-1})$ to predict the next y_t. In other words, y is predicted by decomposing the joint probability into the ordered conditionals:

$$p(y) = \prod_{t=1}^{m} p\left(y_t \mid \{y_1, y_2, ..., y_{t-1}\}, c\right) \tag{3}$$

With an RNN, each conditional probability is modeled as:

$$p\left(y_t \mid \{y_1, y_2, ..., y_{t-1}\}, c\right) = g\left(y_{t-1}, s_t, c\right) \tag{4}$$

Where g is a non-linear function; s_t is the hidden state of the decoder.

5.2.2 Attention Encoder-Decoder Architecture

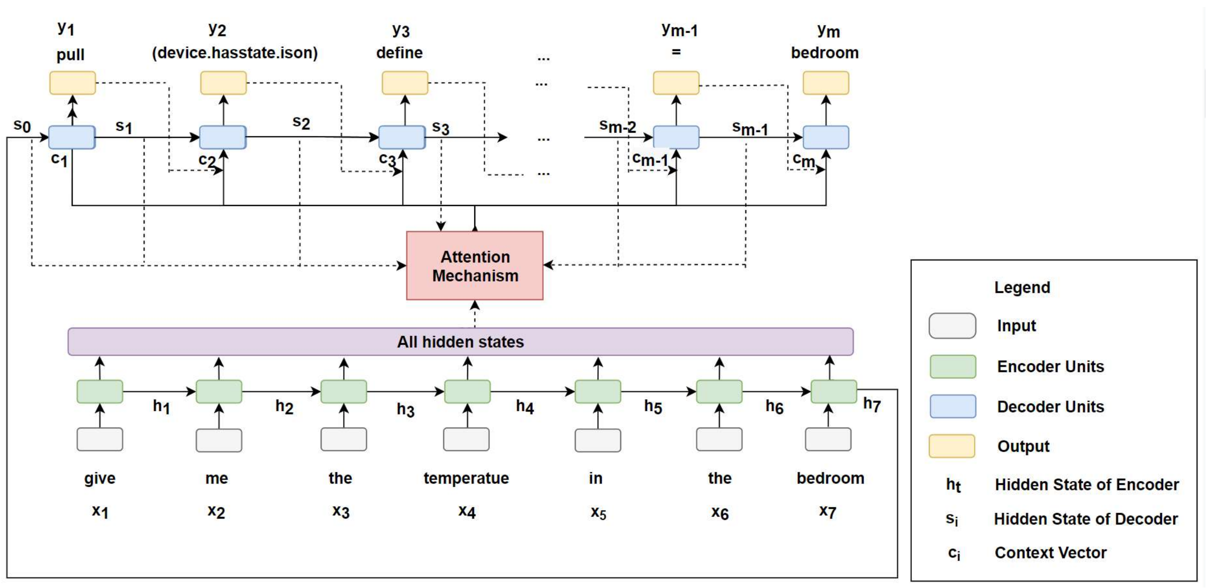


Figure 18: Attention Encoder-Decoder Architecture.

Unfortunately, the basic architecture takes only the encoder's last hidden state, representing the information of the whole input sequence. It may cause loss of information if the input sequence is very long, leading to a lower performance [7]. Therefore, Attention-based Neural Machine Translation is introduced to improve the basic Encoder-Decoder Machine Translation [7, 44]. Instead of taking only the last hidden state of the encoder, the decoder accesses all hidden states. The decoder uses attention to focus on parts of the hidden states of the encoder selectively. The attention takes a sequence of vectors as input for each word and returns an "attention" vector.

Firstly, the encoder of the Attention Encoder-Decoder Architecture has the same function as the basic Encoder-Decoder architecture. However, instead of returning only the last hidden state h_n, the encoder of the attention architecture return all hidden states of the encoder $(h_1, h_2, ..., h_n)$ are used for the decoder's process. In addition, the last hidden state is also used to initialize the decoder h_n.

The decoder is also an RNN, but it is more complex than the decoder of the basic architecture. The decoder also aims to produce the output words by emphasizing the most appropriate information in the input sentence using the Attention Mechanism. Similar to the decoder of the basic architecture, the next word is also predicted given the all preceding words the context vector c_i as follows:

$$p(y) = \prod_{i=1}^{m} p\left(y_t | \left\{y_1, ..., y_{i-1}\right\}, c_i\right) \tag{5}$$

With an RNN, each conditional probability is modeled as:

$$p\left(y_t | \left\{y_1, ..., y_{i-1}\right\}, c_i\right) = g\left(y_{i-1}, s_i, c_i\right) \tag{6}$$

where g is non-linear function, s_i and c_i are the hidden RNN state of the decoder and the context vector at step i. The RNN hidden state of the decoder s_i is computed giving the previous hidden state s_{i-1}, the preceding predicted word y_{i-1} and the context vector c_i.

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{7}$$

However, the vector context of the attention (known as an "attention" vector) is more complex than the basic architecture. It may be considered as a summary of the input sequence and is computed each time step as a weighted sum of all the hidden states of the encoder:

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j \tag{8}$$

Where a_{ij} is the attention weight of each hidden state of the encoder and is computed as the following formula:

$$\alpha_{ij} = \frac{\exp\left(attend\_score\left(s_{i-1}, h_j\right)\right)}{\sum\limits_{k=1}^{n} \exp\left(attend\_score\left(s_{i-1}, h_k\right)\right)} \tag{9}$$

$attend\_score$ provides a more weighted context from the encoder to the decoder. A learning mechanism that the decoder can interpret was giving more 'attention' to the subsequent encoding network when predicting outputs at each step in the output sequence. The entire process of Attention-based architecture is presented in Figure 18.

There are several approaches to calculate the attend score function $attend\_score\left(s_{i-1}, h_j\right)$. Bahdanau's score function [7] and Luong's score function [44] is the true common approaches:

$$attend\_score\left(s_{i-1}, h_j\right) = \begin{cases} s_{i-1} W h_j & [\,Luong's\,function] \\ v_a^T \tanh\left(W_a \left[s_{i-1}; h_j\right]\right) & [Bahdanau's\,function] \end{cases} \tag{10}$$

5.3 Finite Automata

Finite Automata (or Finite State Machine) is an abstract mathematical construct used to reject or accept a sequence of symbols. It takes string symbols as inputs and changes its states accordingly. Its applications are found in pattern matching or formal language syntax checking. Formally, it is defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, in which:

- Q is the set of possible states

- Σ is the set of input symbols

- δ is the transition function: $Q \times \Sigma - > Q$

- q_0 is the initial state

- F is the set of accept states

A sequence of input symbols $a_1 a_2 ... a_n$ in Σ is considered valid if there is a sequence of states $s_1 s_2 ... s_n$ satisfies the following conditions:

- $s_0 = q_0$

- $s_{i+1} = \delta(s_i, a_{i+1})$

- $s_n \in F$

A finite automaton can be visually represented as a state diagram, essentially a directed graph. The graph's nodes show the states and the edges of the input symbols. An example of the finitr automata's CDQL is shown in Figure 19.

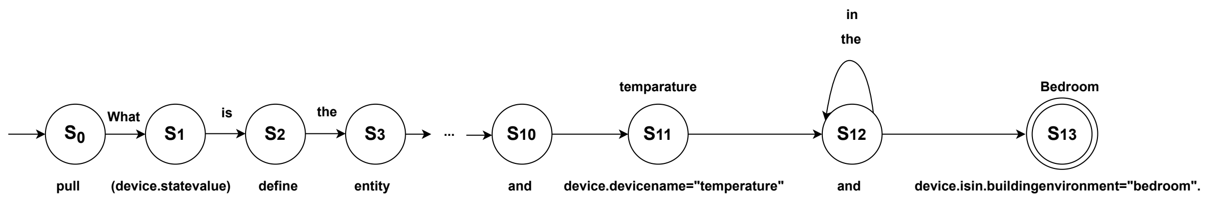


Figure 19: An Example of Finite Automata.

Our paper uses Beam search with CDQL finite automata as a final decision-making layer to choose the best output given target variables. We use the beam search with a Beamwidth is of 3 to look for the top 3 most likely CDQL sequences. The CDQL sequences which are not valid according to the finite automaton that checks the CDQL syntax are filtered out during the beam searching process. The most likely sequence as a result of the CDQL-aware beam search is used to evaluate the model.

6 Evaluation

In this section, firstly, we present the method to create the data set to train our text-to-CDQL models in Section 6.1. After that, we describe how our models are evaluated in 6.2. Then, we conduct several experiments to compare our approaches in Section 6.3. Finally, we discuss and analyze the results in Section 6.4.

6.1 Data Generation

Unlike some popular query languages such as SQL, there are no existing datasets that convert the natural language in textual form to CDQL. Therefore, creating a dataset with text as the input and CDQL as the output is essential to train the text-to-CDQL model. The data set was created using query templates that are an alignment between a natural-language question and its respective CDQL query, with entities replaced by placeholders, as in Figure 20.

Records in the dataset are thus created by replacing <A> with smart devices such as temperature and humidity sensors and <B> with building environments such as living room and

Text: what is the current <A> in the <B>?.
CDQL: pull (device.statevalue) define entity device is from smarthome
where device.hasState.ison = true and device.devicename = "<A>" and
device.isin.buildingenvironment = "<B>".

Figure 20: Example Data Template for Extracting Measurement in One Specific Room

bedroom. Figure 21 shows some examples in the dataset.

Text: what is the current temperature in the living room?.
CDQL: pull (device.statevalue) define entity device is from smarthome
where device.hasState.ison = true and device.devicename =
"temperature" and device.isin.buildingenvironment = "living room".

Text: what is the current humidity in the bedroom?.
CDQL: pull (device.statevalue) define entity device is from smarthome
where device.hasState.ison = true and device.devicename = "humidity" and
device.isin.buildingenvironment = "bedroom".

Figure 21: Example Records

6.2 Metrics

The accuracy is used to evaluate the model. That means that a prediction is correct if it is identical to the ground truth. The reason is that even if the query has only one wrong character, the query cannot be executed.

Apart from the accuracy, similar to translation machines, the word error rate (WER) [37] is popular to evaluate speech recognition systems and translation machines. It is also used to measure the difference between the prediction and the ground truth as the output sequence's length and the ground truth sequence's length may not be the same. Additionally, WER is the number of errors divided by the total words. Therefore, WER is popularly used for evaluating sequence-to-sequence systems.

WER is computed as follow:

$$WER = \frac{S + D + I}{N} \tag{11}$$

S is the number of substitutions performed in the prediction compared to the ground truth; D is the number of deletions; I is the number of insertions; N is the number of words in the ground truth. Since WER measures the prediction's error against the ground truth, the lower WER source means the better model.

We will provide an example to point out how does WER works. It is assumed that the predicted sequence and the target sequence are shown in Figure 22.

Target Sequence: "my name is andy and i am an engineer".
Predicted Sequence: "my name is peter parker and im an engineer".

Figure 22: Example Sequences

For this solution, The word *andy* got substituted *peter* and *i* got substituted *im*. Therefore, $S = 2$. Additionally, there are one new inserted word which is *parker*, so $I = 1$. Apart from that, There is a word dropped - *am*. Therefore, $D = 1$. $N = 9$ because the lengh of the ground truth is nine. Based on these information, WER of the example can be calculated as following:

$$WER = \frac{2+1+1}{9} = \frac{4}{9} =\approx 0.44. \tag{12}$$

6.3 Implementation details

As mentioned in the section 5, because Google Speech API is used for the speech recognition system, we only need to experiment with the text-to-CDQL system. Therefore, we create three models: basic encoder-decoder architecture, Bahdanau, and Luong attention Encoder-Decoder architecture. These models are built using the Tensorflow package, and they are trained with the ADAM optimizer. The learning rate is set to 10^{-5}. The batch size is set to 64. The number of epochs is set to 100. The type of RNN in both encoder and decoder of all models is GRU with 1024 hidden units. We use one 12GB NVIDIA Tesla K80 GPU, and the training environment is Google Colab. All experiments were performed on 4800 training records and evaluated on 256 testing records.

6.4 Discussion and analysis

Figure 23 shows WER and the accuracy of these models. The WER of the Bahdanau Attention Model is the lowest at about 0% after 40 epochs and plateaus at this value in the

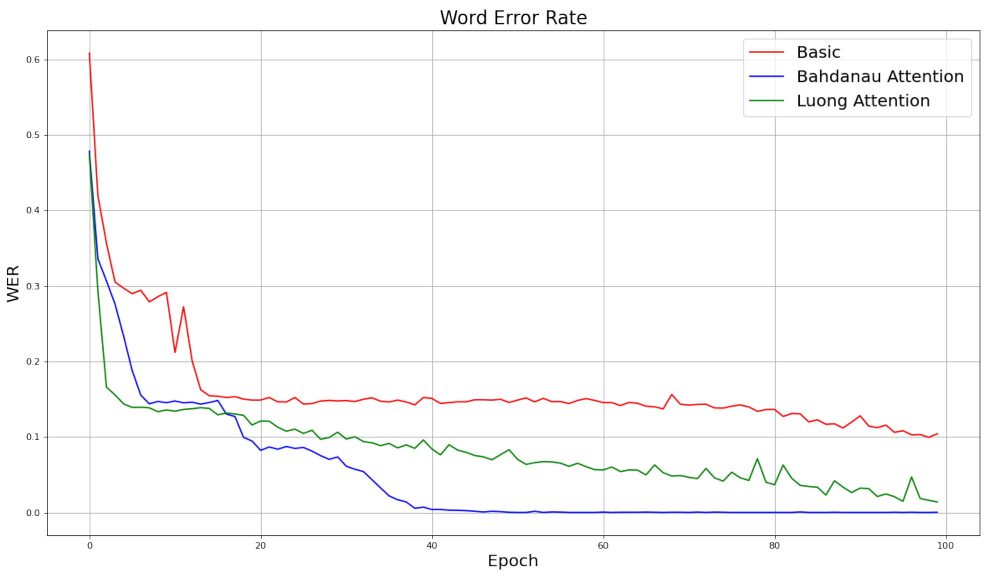


Figure 23: Word Error Rate.

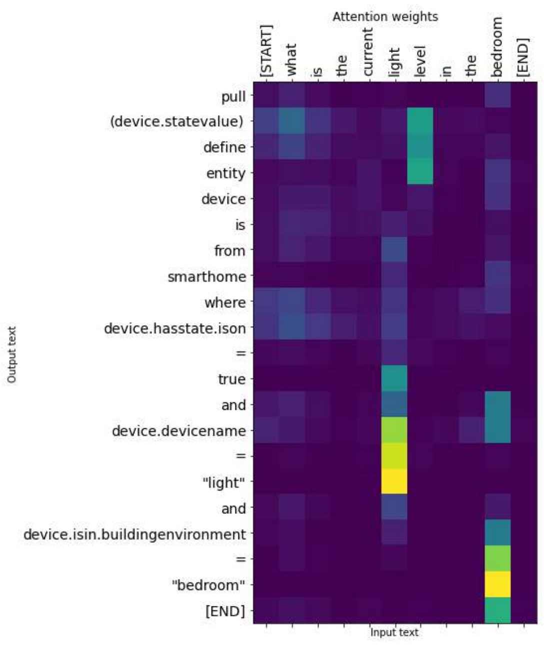
Model	WER %	Accuracy (%)	Running times (ms)
Basic	11	2.3	3395
Bahdanau Attention	0.02	93	5042
Luong Attention	1.3	82	5656

Table 3: Comparison of models.

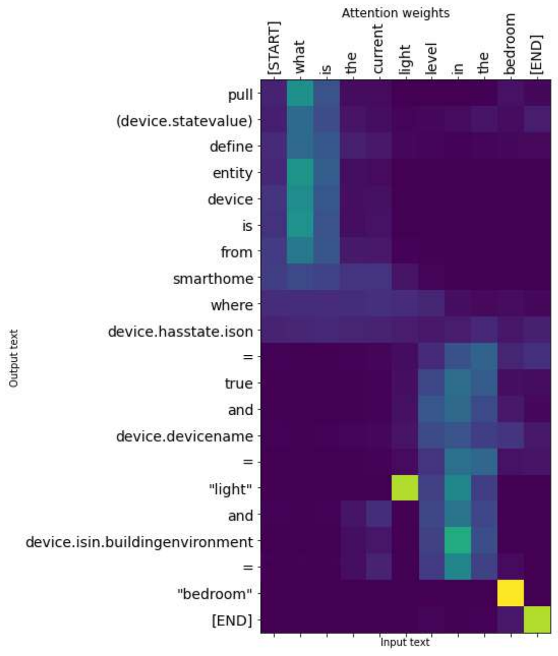
remaining epochs. The error of the basic model is the highest number at 10%. Finally, the Luong attention model reaches about 1.3% of WER.

According to Figure 25, the Bahdanau attention model obtains the highest accuracy at 93%. On the other hand, the accuracy of the Luong attention is at 82%. On the other hand, the accuracy of the basic model is only 2.3%. That means that the basic model does not perform very well for the text-to-CDQL system. Additionally, looking at 3, the training time of the basic model is very short, with only 3395 seconds for 100 epochs. On the other hand, both attention models spend over 5000 seconds training. Overall speaking, based on the accuracy and WER metric, the Bahdanau attention is the best model among the three models for converting text into CDQL because it reaches the highest accuracy and the lowest error. On the other hand, the basic encoder-decoder model is ineffective for the text-to-CDQL system.

The basic encoder-decoder architecture works ineffectively due to its limit in modeling capability. The reason is that the decoder of the basic architecture takes only the last hidden state of the encoder architecture. As a result, this architecture is ineffective for long sequences



(a) Bahdanau Attention



(b) Luong Attention

Figure 24: Attention Weights.

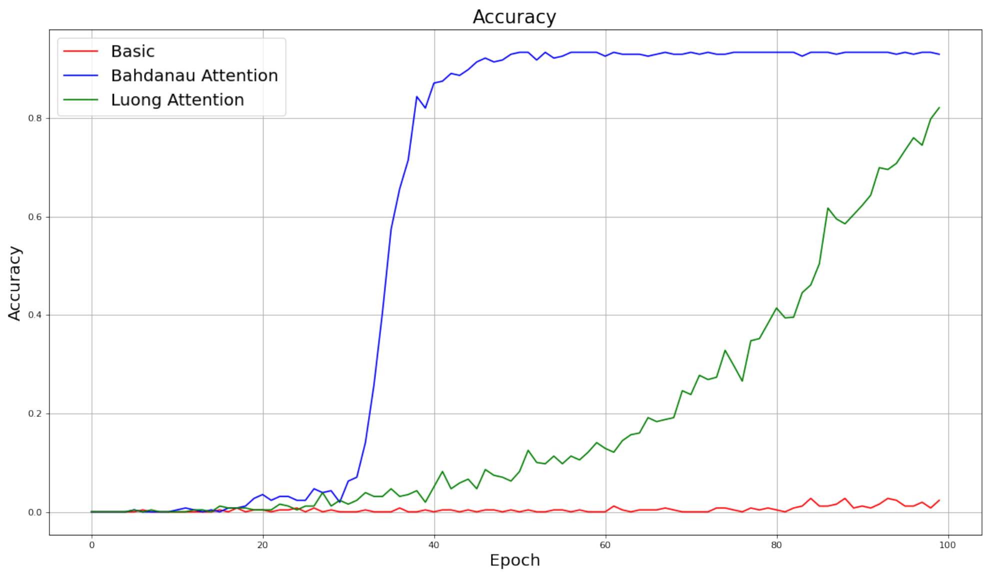


Figure 25: Accuracy.

due to the vanishing and exploding gradient problems. However, the short training time is the only advantage of the basic architecture since it requires less computation than the other two architectures.

On the other hand, the encoder-decoder architectures with attention significantly improve the performance. The reason is that there is no way to give more importance to some of the input words than others while translating the sentence in the decoder of the basic encoder-decoder architecture. Meanwhile, the decoders of the Luong attention and Bahdanau attention architectures take in all hidden states of the encoder. It thus keeps as much information as the encoder. Additionally, the attention-based architectures use the attention score to focus more on essential words. Therefore, attention-based architectures work more effectively compared to basic architecture.

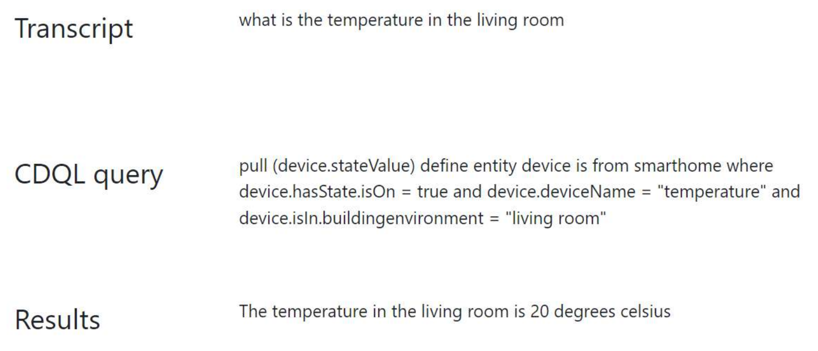
Figure 24 presents a visualization of attention alignments of Bahdanau and Luong attention with the same sentence. According to Figure 24a, the syntax in the output CDQL *device.devicename = "light"* pays attention to *light* in the input text. The syntax in the output CDQL *device.devicename = "bedroom"* focuses on *bedroom*. It means that when having these words (*light* and *bedroom*) in the input sequence, the probability of the formulas in the output sequence will be higher. Therefore, the accuracy of the model is high. By contrast, only *"light"* in the output CDQL of the Luong attention model pay its attention to *light* in the input text. Similarly, only *"bedroom"* in the output CDQL pay its attention to *bedroom* in the input text. In other words, the Luong attention model only allows the words in the output sequence to focus on the same words on the input sequence. Therefore, this partly explains why Bahdanau Attention works better than Luong Attention in this task.

7 User Interface Demonstration

Figure 26 shows the screenshot of the user interface for controlling a smart home. The top side of the UI shows the organization's logo, the project's name, and the logo of the smart home application. The bottom of the banner is the main content of the UI that presents 3 parts: **Transcript**, **CDQL Query**, and **Result**:

- **Transcript** shows the text converted from the human speech by the speech recognition system.
- **CDQL Query** presents the CDQL query generated from the transcript by the Text-to-CDQL system.
- **Result** illustrates the answer of the request that returned from the CooaS platform.

In the scenario of this demonstration, users ask about one measurement in a specific building


Transcript
what is the temperature in the living room

CDQL query
pull (device.stateValue) define entity device is from smarthome where device.hasState.isOn = true and device.deviceName = "temperature" and device.isIn.buildingenvironment = "living room"

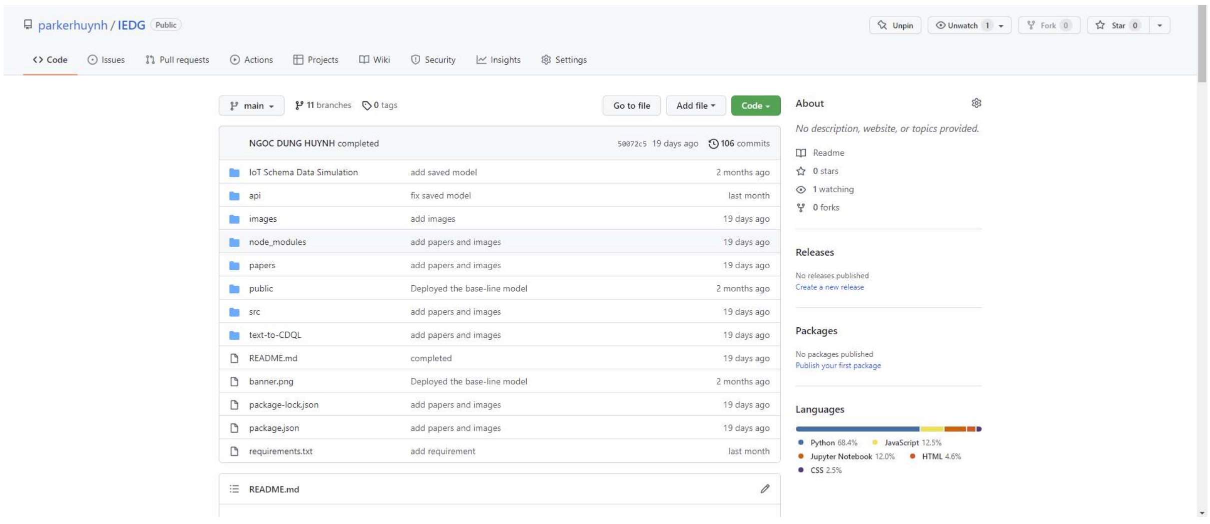
Results
The temperature in the living room is 20 degrees celsius

Figure 26: Main Interface of the Demonstration.

environment of the smart home. Firstly, to wake up the UI, the users need to utter ***"Hi Jarvis"*** or ***"Jarvis"***. After that, a *beep* is sounded, and the UI is ready to operate. If the users talk without the waked-up words, the UI will not wake up, and there will be no beep.

After waking up the UI, users can ask the UI about the temperature of the living room with *"What is the temperate in the living room"*. The speech recognition system takes the voiced question and converts it into the transcript presented in the transcript section. The transcript will be then fed to the Text-to-CDQL system to generate the CDQL query as *"pull (device.stateValue) define entity device is from smart home where device.hasState.isOn = true and device.deviceName = "temperature" and device.isIn.buildingenvironment = "living room")"* presented in the CDQL query section. Finally, the CDQL query is sent to the platform to extract the value of the temperature in the living room, which is shown in the Result section as *"The temperature in the living room is 20 degrees celsius"*. These results will disappear after 3 seconds to make space for another question. Apart from asking about the temperate of one specific room, users can ask about the humidity, the light level, or the volume level of the smart TV. In addition, users can also ask the UI more contextual questions. For example, if we ask *"what is the temperature in the smart home"*. The UI will extract data from all the temperature sensors in the smart home and return the average temperature.

In another scenario, users can also ask about the state of one smart device, such as *"Is the light in the living room turning on?"* or *"Give me the state of the light in the living room?"*. The CDQL query will be *"pull (device.hasState.isOn) define entity device is from smart home where* *device.deviceName = "light" and device.isIn.buildingenvironment = "living room"*. The result is *"Yes, the light in the living room is on."*.


parkerhuynh / IEDG Public

Unpin | Unwatch 1 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main | 11 branches | 0 tags | Go to file | Add file | Code

NGOC DUNG HUYNH completed	50072c5 19 days ago	106 commits
IoT Schema Data Simulation	add saved model	2 months ago
api	fix saved model	last month
images	add images	19 days ago
node_modules	add papers and images	19 days ago
papers	add papers and images	19 days ago
public	Deployed the base-line model	2 months ago
src	add papers and images	19 days ago
text-to-CDQL	add papers and images	19 days ago
README.md	completed	19 days ago
banner.png	Deployed the base-line model	2 months ago
package-lock.json	add papers and images	19 days ago
package.json	add papers and images	19 days ago
requirements.txt	add requirement	last month

README.md

About

No description, website, or topics provided.

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Python 68.4% ● JavaScript 12.5%
● Jupyter Notebook 12.0% ● HTML 4.6%
● CSS 2.5%

Figure 27: GitHub Screenshot.

Figure 27 present the screenshot of the project's code. The code of the project is located at **https://github.com/parkerhuynh/IEDG.**. The first stage of deploying the User Interface is to install the required packages such as *Tensowflow* and *tensorflow-text* via the following command:

```
pip install -r requirements.txt
```

Apart from that, we also need to install *npm* to integrate the user interface. Depending on the operating system such as Windows, Linux, and Mac, the method to install npm can be found at the link: **https://docs.npmjs.com/downloading-and-installing-node-js-and-npm**.

Because a pre-trained model is used for speech recognition, we do not need to train this component. Therefore, the second stage is to create the text-to-CDQL system. Firstly, We create the data to train and evaluate the model as follows:

```
python text-to-CDQL\data_generator.py
```

After that, we train and save the text-to-CDQL system. Before training the model, we should set the parameters of your model, such as the number of hidden units, epochs, learning rate, and the type of the model in the file *config.py*. To train the model, run:

```
python text-to-CDQL\training.py
```

The last stage is to integrate the user interface by running:

```
npm start
python api/backend.py
```

8 Conclusion and Future Work

This paper proposes a system to convert queries in natural language form to CDQL form. It is based on two models: speech recognition and text-to-CDQL models. We use Google speech recognition for the speech recognition part, while the text-to-CDQL model is based on the encoder-decoder architecture. Three encoder-decoder architectures are evaluated: basic encoder-decoder, Luong attention, and Bahdanau attention. The results show that the Bahdanau attention is the best model since it achieves the highest accuracy and lowest WER. On the other hand, the performance of the Luong attention model is moderate. The basic encoder-decoder architecture does not work very well for the text-to-CDQL task. Generally speaking, two attention models are better and can be used for the text-to-CDQL system.

The limitation of this paper is that the proposed system only supports simple questions for controlling a smart home, such as extracting data from the CooaS platform. Furthermore, it does not support updating requests such as turning on or turning off smart devices. Therefore, in the future, our work will solve more complex CDQL queries.

Apart from that, the encoder-decoder architectures based on RNN are considered outdated. Therefore, we will use more advanced algorithms such as Transformer [65] to improve the performance of the text-to-CDQL system. No less important, the system is based on two different models trained separately. Therefore, a speech-to-CDQL is another potential idea to improve accuracy and running time. Finally, because this work only implements the application in the smart home topic, we want to target other applications in smart cities or businesses.

References

[1] M. A, G. G, Hinton, and P. G, *Understanding how deep belief networks perform acoustic modelling*, IEEE International Conference on Acoustics, 66 (2012), pp. 4273–4276.

[2] A. Alatwi, S. So, and K. K. Paliwal, *Perceptually motivated linear prediction cepstral features for network speech recognition*, 10th International Conference on Signal Processing and Communication Systems (ICSPCS), (2017).

[3] S. Ali and N. Bouguila, *Maximum a posteriori approximation of hidden markov models for proportional sequential data modeling with simultaneous feature selection*, IEEE Transactions on Neural Networks and Learning Systems, (2021), pp. 1–12.

[4] S. A. Alim and N. K. A. Rashid, *Some Commonly Used Speech Feature Extraction Algorithms*, Intech Open, 2018.

[5] J. Allen, G. Ferguson, and A. Stent, *An architecture for more realistic conversational systems*, in Proceedings of the 6th International Conference on Intelligent User Interfaces, IUI '01, New York, NY, USA, 2001, Association for Computing Machinery, pp. 1–8.

[6] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, *Deep speech 2: End-to-end speech recognition in english and mandarin*, arXiv:1512.02595, (2015).

[7] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016.

[8] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, *End-to-end attention-based large vocabulary speech recognition*, arXiv:1508.04395, (2016).

[9] T. Beck, A. Demirgüç-Kunt, and R. Levine, *A new database on the structure and development of the financial sector*, The World Bank Economic Review, 14 (2000), pp. 597–605.

[10] O. Caglayan, R. Sanabriay, S. Palaskary, L. Barrault, and F. Metze, *Multimodal grounding for sequence-to-sequence speech recognition*, arXiv:1811.03865v2, (2019).

[11] A. Chamoli, A. Semwal, and N. Saikia, *Detection of emotion in analysis of speech using linear predictive coding techniques (l.p.c)*, International Conference on Inventive Systems and Control (ICISC), (2017).

[12] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, *Listen, attend and spell: A neural network for large vocabulary conversational speech recognition*, 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2016).

[13] C.-C. Chiu, A. Tripathi, K. Chou, C. Co, N. Jaitly, D. Jaunzeikare, A. Kannan, P. Nguyen, H. Sak, A. Sankar, J. Tansuwan, N. Wan, Y. Wu, and X. Zhang, *Speech recognition for medical conversations*, arXiv:1711.07274, (2018).

[14] K. Cho, B. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, arXiv:1406.1078, (2014).

[15] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014.

[16] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, *End-to-end continuous speech recognition using attention-based recurrent nn: First results*, arXiv:1412.1602, (2014).

[17] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, *Attention-based models for speech recognition*, arXiv:1506.07503v1, (2015).

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[19] L. Dong and M. Lapata, *Coarse-to-fine decoding for neural semantic parsing*, 2018.

[20] L. Dong, S. X. Xu, and B. Xu, *Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition*, 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2018).

[21] E. Edwards, W. Salloum, G. Finley, J. Fone, G. Cardiff, M. Miller, and D. Suendermann-Oeft, *Medical speech recognition: Reaching parity with humans*, Springer International Publishing, (2017), pp. 512–524.

[22] H. Elkhorchani and K. Grayaa, *Novel home energy management system using wireless communication technologies for carbon emission reduction within a smart grid*, Journal of Cleaner Production, 135 (2016), pp. 950–962.

[23] M. Franzese and A. Iuliano, *An introduction to hidden markov models*, IEEE ASSP Magazine, 3 (1986), pp. 4 – 16.

[24] J. Gao, M. Galley, and L. Li, *Neural approaches to conversational ai*, in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18, New York, NY, USA, 2018, Association for Computing Machinery, pp. 1371–1374.

[25] A. Graves, S. Fernandez, and K. Sanjeev, *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks*, ICML '06: Proceedings of the 23rd international conference on Machine learning, (2006), pp. 369–376.

[26] A. Graves and N. Jaitly, *Towards end-to-end speech recognition with recurrent neural networks*, in International conference on machine learning, PMLR, 2014, pp. 1764–1772.

[27] A. Graves, N. Jaitly, and A.-r. Mohamed, *Hybrid speech recognition with deep bidirectional lstm*, IEEE Workshop on Automatic Speech Recognition and Understanding, (2013).

[28] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng, *Deep speech: Scaling up end-to-end speech recognition*, arXiv:1412.5567, (2014).

[29] D. Harwath and J. Glass, *Deep multimodal semantic embeddings for speech and images*, arXiv:1511.03690, (2015).

[30] A. Hassani, A. Medvedev, P.-D. Haghighi, S. Ling, A. Zaslavsky, and P.-P. Jayaraman, *Context definition and query language: Conceptual specification, implementation, and evaluation*, Selected Papers from the 2nd Global IoT Summit: IoT Technologies and Applications for the Benefit of Society), (2019).

[31] H. Hermansky, *Perceptual linear predictive (plp) analysis of speech*, The Journal of the Acoustical Society of America, 87 (1990).

[32] R. Hillestad, J. Bigelow, A. Bower, F. Girosi, R. Meili, R. Scoville, and R. Taylor, *Can electronic medical record systems transform health care? potential health benefits, savings, and costs*, Health affairs (Project Hope), 24 (2005), pp. 1103–17.

[33] W. Hwang, J. Yim, S. Park, and M. Seo, *A comprehensive exploration on wikisql with table-aware word contextualization*, 2019.

[34] Y. Jie, J. Yong-Pe, L. Jun, G. Yun, and X. Wei, *Smart home system based on iot technologies*, IEEE: 2013 International Conference on Computational and Information Sciences, (2013).

[35] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Third Edition draft)*, Pearson, 2020.

[36] S. Karita, N. Soplin, S. Watanabe, M. Delcroix, A. Ogawa, and T. Nakatani, *Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration*, INTERSPEECH 2019, (2019).

[37] D. Klakow and J. Peters, *Testing the correlation of word error rate and perplexity*, Speech Communication, 38 (2002), pp. 19–28.

[38] L. Koutti, M. Machkour, and H. Bais, *An arabic natural language interface for querying relational databases based on natural language processing and graph theory methods*, International Journal of Reasoning-based Intelligent Systems, 10 (2018), p. 155.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., vol. 25, Curran Associates, Inc., 2012.

[40] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.

[41] L. Liu, Y.-C. Lin, and J. Reid, *Improving the performance of the lstm and hmm model via hybridization*, arXiv:1907.04670v4, (2021).

[42] W. Liu, Q. Liao, F. Qiao, W. Xia, C. Wang, and F. Lombardi, *Approximate designs for fast fourier transform (fft) with application to speech recognition*, IEEE Transactions on Circuits and Systems I: Regular Papers, 66 (2019), pp. 4727 – 4739.

[43] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, 2019.

[44] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, 2015.

[45] S. Lyu, A. Rana, S. Sanner, and M. R. Bouadjenek, *A workflow analysis of context-driven conversational recommendation*, in Proceedings of the 30th International Conference on the World Wide Web (WWW-21), Ljubljana, Slovenia, 2021. To appear.

[46] A. Maas, P. Qik, Z. Xie, A. Hannun, C. Lengerich, D. Jurafsky, and A. Ng, *Building dnn acoustic models for large vocabulary speech recognition*, arXiv:1406.7806, (2015).

[47] O. Z. Mamyrbayev, K. Alimhan, B. Amirgaliyev, B. Zhumazhanov, D. Mussayeva, and F. Gusmanova, *Multimodal systems for speech recognition*, International Journal of Mobile Communications, 18 (2020), pp. 314–326.

[48] J. Mao, Q. Lin, and J. Bian, *Application of learning algorithms in smart home iot system security*, Mathematical Foundations of Computing, 1 (2018), pp. 63–76.

[49] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013.

[50] H. Mukherjee, S. Obaidullah, O. Santosh, S. Phadikar, and K. Roy, *Line spectral frequency-based features and extreme learning machine for voice activity detection from audio signal*, International Journal of Speech Technology, 21 (2017), pp. 753–760.

[51] E. Ngai, L. Xiu, and D. Chau, *Application of data mining techniques in customer relationship management: A literature review and classification*, Expert Syst. Appl., 36 (2009), pp. 2592–2602.

[52] P. Paramonov, *Fast algorithm for isolated words recognition based on hidden markov model stationary distribution*, 2017 IEEE 4th International Conference on Soft Computing & Machine Intelligence (ISCMI), (2017).

[53] V. Peddinti, D. Povey, and S. Khudanpur, *A time delay neural network architecture for efficient modeling of long temporal contexts*, in Sixteenth annual conference of the international speech communication association, 2015.

[54] J. Pennington, R. Socher, and C. Manning, *Glove: Global vectors for word representation*, vol. 14, 01 2014, pp. 1532–1543.

[55] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, *Purely sequence-trained neural networks for asr based on lattice-free mmi.*, in Interspeech, 2016, pp. 2751–2755.

[56] S. Rautmare and D. M. Bhalerao, *Mysql and nosql database comparison for iot application*, in 2016 IEEE International Conference on Advances in Computer Applications (ICACA), 2016, pp. 235–238.

[57] G. Saon, G. Kurata, T. Sercum, S. Audhkhasi, Kartik amd Thomas, D. Dimitriadis, X. Cui, M. Ramabhadran, Bhuvana amd Picheny, L.-L. Lim, B. Roomi, and P. Hall, *English conversational telephone speech recognition by humans and machines*, arXiv:1703.02136, (2017).

[58] T. Shi, K. Tatwawadi, K. Chakrabarti, Y. Mao, O. Polozov, and W. Chen, *Incsql: Training incremental text-to-sql parsers with non-deterministic oracles*, 2018.

[59] S. M. S. A. K. Shikder, *A smart home automation and metering system using internet of things (iot)*, IEEE: 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), (2019).

[60] B. Shillingford, Y. Assael, M. Hoffman, T. Paine, C. Hughes, U. Prabhu, H. Liao, H. Sak, K. Rao, L. Bennett, M. Mulville, B. Coppin, B. Laurie, A. Senior, and N. Freitas, *Large-scale visual speech recognition*, arXiv:1807.05162, (2018).

[61] T. Srinivasan, R. Sanabria, and F. Metze, *Looking enhances listening: Recovering missing speech using images*, arXiv:2002.05639, (2020).

[62] T. Srinivasan, R. Sanabria, F. Metze, and D. Elliott, *Multimodal speech recognition with unstructured audio masking*, arXiv:1511.03690, (2020).

[63] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, 2014.

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, arXiv:1706.03762v5, (2017).

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2017.

[66] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, *Sequence-discriminative training of deep neural networks.*, in Interspeech, vol. 2013, 2013, pp. 2345–2349.

[67] K. Walsh and A. Callan, *Perceptions, preferences, and acceptance of information and communication technologies in older-adult community care settings in ireland: A case-study and ranked-care program analysis*, Ageing International, 36 (2010), pp. 102–122.

[68] C. Wang, K. Tatwawadi, M. Brockschmidt, P.-S. Huang, Y. Mao, O. Polozov, and R. Singh, *Robust text-to-sql generation with execution-guided decoding*, 2018.

[69] W. Wang, Y. Tian, H. Xiong, H. Wang, and W.-S. Ku, *A transfer-learnable natural language interface for databases*, 2018.

[70] X. Xu, C. Liu, and D. Song, *Sqlnet: Generating structured queries from natural language without reinforcement learning*, 2017.

[71] N. Yaghmazadeh, Y. Wang, I. Dillig, and T. Dillig, *Sqlizer: Query synthesis from natural language*, Proc. ACM Program. Lang., 1 (2017).

[72] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, 2020.

[73] Y. Yoda, K. Tamura, S. Adachi, N. Otani, S. Nakayama, and M. Shima, *Effects of the use of air purifier on indoor environment and respiratory system among healthy adults*, International Journal of Environmental Research and Public Health, 17 (2020), p. 3687.

[74] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, *Typesql: Knowledge-based type-aware neural text-to-sql generation*, 2018.

[75] Y. Zhang, W. Chan, and N. Jaitly, *Very deep convolutional networks for end-to-end speech recognition*, 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2017), pp. 4845–4849.

[76] V. Zhong, C. Xiong, and R. Socher, *Seq2sql: Generating structured queries from natural language using reinforcement learning*, CoRR, abs/1709.00103 (2017).

[77] V. Zhong, C. Xiong, and R. Socher, *Seq2sql: Generating structured queries from natural language using reinforcement learning*, arXiv preprint arXiv:1709.00103, (2017).

[78] B. Zhou, W. Li, K. W. Chan, Y. Cao, Y. Kuang, X. Liu, and X. Wang, *Smart home energy management systems: Concept, configurations, and scheduling strategies*, Renewable and Sustainable Energy Reviews, 61 (2016), pp. 30–40.

[79] W. Zhou, R. Schlüter, and H. Ney, *Full-sum decoding for hybrid hmm based speech recognition using lstm language model*, arXiv:2004.00967, (2020).