

**Capstone Project – Cover Sheet**

Project

**Instruction:**

* Marks will be awarded for good presentation and thoroughness in your approach.
* Referencing Code: If you use some code, or ideas for code, which are taken or adapted from another source (book, magazine, internet, discussion forum, etc), then this **must** be cited and referenced using the APA Name convention within your source code. Failure to reference code properly is considered as plagiarism.
* Complete this cover sheet and attach it to your project.

|  |  |  |
| --- | --- | --- |
| **Student declaration:** | | |
| *I declare that:* | 1. *I understand what is meant by plagiarism* 2. *The implication of plagiarism have been explained to me by our lecturer* 3. *This project is all my work and I have acknowledged any use of the published or unpublished works of other people.* | |
| *ckc*  Student Signature: | | 15/03/2024  Date: |

|  |  |  |  |
| --- | --- | --- | --- |
| Assignment Title | CoinBert: Co-interactive Bert Joint Intent Detection and Slot Filling for Task-Based Dialogue System | Intake Code | APDMF2303AI |
| Student Name | Chong Kah Chun Kelvin | TPNo | TP073978 |
| Supervisor | Mr Raheem Mafas | 2nd Marker | Mr Lee Kim Keong |

Acknowledgements

First, I would like to thank my supervisor Mr. Raheem Mafas for guiding me throughout my project and providing me valuable advises whenever I face problems. I would also like to thank Mr. Lee Kim Keong for reading through my project paper and providing me guidance. Lastly, I would like to show my appreciation to my family and friends who have helped me on my survey and providing me encouragement whenever I faced difficulties while doing this project. Without the support of these individuals, this project would not have been possible. I am grateful for their help.

Table of Contents

[**1.0** **Introduction** 6](#_Toc161365065)

[**1.1 Background** 6](#_Toc161365066)

[**1.2 Problem Statement** 7](#_Toc161365067)

[**1.3 Aim of Study & Objectives** 8](#_Toc161365068)

[**2.0** **Basics** 8](#_Toc161365069)

[**2.1 Sequence Modelling Problem** 8](#_Toc161365070)

[**2.2 Sequential Architecture** 9](#_Toc161365071)

[*2.2.1 Artificial Neural Network (ANN)* 10](#_Toc161365072)

[*2.2.2 Recurrent Neural Network (RNN)* 11](#_Toc161365073)

[*2.2.3 Long Short-Term Memory (LSTM)* 12](#_Toc161365074)

[**2.3 Transformer Architecture** 13](#_Toc161365075)

[**2.4 Word Embeddings** 15](#_Toc161365076)

[2.5 Sequence Labelling Algorithms 15](#_Toc161365077)

[**3.0 Related Works** 16](#_Toc161365078)

[**3.1** **Joint Intent Detection and Slot Filling** 16](#_Toc161365079)

[**3.2 Transfer Learning** 19](#_Toc161365080)

[**3.3 Evaluation Metrics** 20](#_Toc161365081)

[**3.4 Benchmark Datasets** 21](#_Toc161365082)

[**3.5 Related Works Models Performance** 22](#_Toc161365083)

[**3.6 Related Works Experiment Settings** 22](#_Toc161365084)

[**4** **Methodology** 23](#_Toc161365085)

[**4.1 Model Architecture** 23](#_Toc161365086)

[**4.2 Shared Encoder** 25](#_Toc161365087)

[**4.3 Cross Attention Module** 26](#_Toc161365088)

[**4.4 Decoder** 28](#_Toc161365089)

[**4.5 Data Processing** 29](#_Toc161365090)

[**5** **Dataset** 30](#_Toc161365091)

[**5.1** **Benchmark Dataset Selection** 30](#_Toc161365092)

[**5.2 Custom Dataset Collection & Annotation** 31](#_Toc161365093)

[*5.2.1 Data Collection* 31](#_Toc161365094)

[*5.2.2 Data Annotation* 32](#_Toc161365095)

[**6** **Experiment & Evaluation** 32](#_Toc161365096)

[**6.1** **Setup** 33](#_Toc161365097)

[**6.2 Benchmark Models** 34](#_Toc161365098)

[**6.3 Metrics** 35](#_Toc161365099)

[**7** **Results** 36](#_Toc161365100)

[**7.1** **Benchmark** 36](#_Toc161365101)

[**8** **Critical Analysis** 38](#_Toc161365102)

[**9** **Conclusion** 38](#_Toc161365103)

[**References** 39](#_Toc161365104)

List Of Figures

[Figure 1 IOB Slot Tag Example 8](#_Toc161365105)

[Figure 2 ANN Architecture (Azlah et al., 2019) 9](#_Toc161365106)

[Figure 3 RNN Architecture (Feng et al., 2017) 10](#_Toc161365107)

[Figure 4 LSTM Architecture (Houdt et al., 2020) 11](#_Toc161365108)

[Figure 5 Transformer Architecture (Vaswani et al., 2017) 12](#_Toc161365109)

[Figure 6 Self-attention Mechanism (Vaswani et al., 2017) 13](#_Toc161365110)

[Figure 7 SlotRefine Architecture (Wu et al., 2020) 16](#_Toc161365111)

[Figure 8 FAN Architecture (Huang et al., 2022) 17](#_Toc161365112)

[Figure 9 Ctran Architecture (Rafiepour & Sartakhti, 2023) 18](#_Toc161365113)

[Figure 10 Proposed CoinBert Architecture 22](#_Toc161365114)

[Figure 11 Shared Encoder component of CoinBert 24](#_Toc161365115)

[Figure 12 Co-interactive Module (Qin et al., 2020a) 25](#_Toc161365116)

[Figure 13 Decoder Component of CoinBert 27](#_Toc161365117)

[Figure 14 Sample Survey Form’s Question 30](#_Toc161365118)

List Of Tables

[Table 1 Reviewed Works’ Model Performance 21](#_Toc161365119)

[Table 2 Reviewed Works’ Experimental Settings 21](#_Toc161365120)

[Table 3 Hyperparameter Settings 33](#_Toc161365121)

[Table 4 Benchmarks Models 33](#_Toc161365122)

[Table 5 Result of Coinbert + CNN vs Coinbert 35](#_Toc161365123)

[Table 6 Result of our model vs state-of-the-art 36](#_Toc161365124)

**CoinBert: Co-interactive Bert Joint Intent Detection and Slot Filling for Task-Based Dialogue System**

**Abstract.** Intent detection and Slot filling are two of the most important tasks in natural language understanding especially in task-based dialogue systems. Recent works have focus on using joint learning methods for improving the performance of the models and have succeeding in obtaining significant results. However, despite various models have been proposed with excellent performances there are lack of practical implementation of such system on real-world dataset making it hard to determine the robustness of the model toward noises and smaller sample size. Consequently, the lack of practical implementation also causes very little people to understand the values of task-based dialogue systems to various closed domain applications. In this project, we propose a Co-interactive BERT model with convolutional layer for Joint learning intent detection and slot-filling tasks. We also collect data to build our custom dataset to simulate a real-world task-based application to test how our model would perform on such dataset and to showcase the business value of the system in task-based applications. We experimented our model with 2 variations, the first one uses distilbert without convolutional layer, the and the second one uses distilbert with convolutional layer. Both models were tested with two benchmark dataset and our collected custom dataset and shown good performance to all three of the datasets.

**Keywords** Natural Language Understanding • Task-based dialogue system • Intent Detection and Slot Filling • Co-interactive • Joint Learning

# **Introduction**

This chapter will be providing the background, problem statement, aims, objectives, and the significance of this project. This paper has a total of 9 chapters, including the current chapter. In chapter 2, the fundamental concepts that are important for this project will be discussed. Following the exploration of these fundamental concepts, Chapter 3 will present the reviewed Intent Detection and Slot Filling (IDSF) works previously done. Then, the proposed architecture inspired by the reviewed works will be discussed in Chapter 4. In Chapter 5, we will be discussing about the dataset collection and decisions. From Chapter 6 onwards, details of the implementation for the system of this project will be discussed. Chapter 6 discussed about the experimental settings and evaluation metrics used. Chapter 7 shows and discuss the results obtained for the proposed model. In Chapter 8, we will discuss about the findings and performance of the model in details. Lastly, Chapter 9 will be the conclusion of this project.

## **1.1 Background**

One of the important applications of Natural Language Generation (NLU) is the task-based dialogue system, which often known as virtual assistant. The main objective of a task-based dialogue system is to assist users to accomplish specific tasks through natural language to help saving the time of the users by reducing the steps required for performing a task. A task-based dialogue system has two main subtasks, which are intent detection and slot filling. Generally, intents detection is considered as a multiclass classification problem and slots filling involves extracting keywords from the given utterance for the system to conduct certain action. The challenges of IDSF problem lies within balancing the accuracies of the model on detecting intent and identifying slot tag because no matter how well the model perform on one of these tasks, if their corresponding counterpart does not perform well the system would not be useful in practical environment. Typically, for a multi-task problem there are two options for the learning objectives:

1. Separate the IDSF tasks as different learning objective, meaning the loss is computed separate.
2. Joint the learning task of IDSF so the loss for both of the tasks are computed together

Recent works have been focusing on joint learning approaches as it was evident that both of these tasks are dependent to each other (Chen et al., 2019; Haihong et al., 2019; Wang et al., 2020; Wu et al., 2020; Qin et al., 2020a; Hui et al., 2021; Huang et al., 2022; Rafiepour & Sartakhti, 2023). Most of the recent models were benchmarked on well-known datasets for IDSF such as ATIS (Hemphill et al., 1990) and SNIPS (Coucke et al., 2018). However, the potential of these model to be used in applications of business contexts remain uncertain as there are a lack of implementation on real-world dataset. Consequently, very little utilization of IDSF system have been done on software applications to enhance user experiences. Additionally, real-word dataset for IDSF often contains much noises and less samples, therefore, assessing the resilience of the models on such datasets is crucial as well.

## **1.2 Problem Statement**

Virtual assistants are getting more popular especially with the advent of large language model (LLM) such as ChatGPT that helps to answer the user’s queries allowing the users to save their time. Nevertheless, even though there are several popular virtual assistant available in the market (Alexa, Google Assistant, Apple Siri), there are very little utilization of such systems in a more specific domain application to helps their users to perform task. For instance, in a project management software the system could be utilized to create task through users commanding it via natural language (e.g. “Create a task title project development and add staff from the development group to participate”). Hence, utilization of task-based dialogue system in closed domain may have not been explored enough.

On the other hand, the development of a model able to perform intent detection task and slot filling task well is very challenging as the model need to balance the performance between these tasks for practical use. Additionally, real-world dataset often contains less sample size and noises so a novel method that not only perform well on benchmark dataset but also robust to real-world dataset is important.

The purpose of this project is to explore various novel solutions for IDSF and develop a model that not only perform well on benchmark dataset but also on the dataset collected to simulate real-work application for this project to ensure the robustness of the model and display the value of IDSF in various application. Moreover, in this experiment we also reviewed various state-of-the-art (SOTA) models in previous works to compare their performance with our proposed model to determine if our proposed model is comparable to the SOTA models.

## **1.3 Aim of Study & Objectives**

The aim of this study is to develop a robust deep learning architecture that perform well for IDSF tasks by fusing the intent detection learning and slot filling learning as a joint learning objective. Furthermore, to collect and annotate a custom dataset that is similar to a real-world dataset for showcasing the business value of IDSF in task-based applications and robustness of the model. To realize this project’s goal, the following objectives must be achieved:

1. To explore methodologies that facilitate joint learning strategy to improve the IDSF model performance
2. To explore pre-trained LLM and utilize them as transfer learning technique to improve model performance
3. To utilize a custom application that simulate real-world applications to collect data that are similar to real-world dataset to test model robustness
4. To explore various algorithms that can improve the model final performance

# **Basics**

This chapter explains the important concepts and algorithms that this work built upon on to help the readers to better understand the subsequent chapters.

## **2.1 Sequence Modelling Problem**

Artificial Intelligence (AI) is a broad field, one of the major areas of study in AI is NLP. NLP became more prominent recently with various interesting LLM being introduced by major technical companies producing significant results for various NLP problems, for instance, text to speech by OpenAI SORA (Liu et al., 2024), image to text Google Gemini (Perera & Lankathilake, 2023), and open domain conversational agent by OpenAI ChatGPT (Kashyap, 2023). NLP revolves around the study of human linguistics, in professional terminology it is known as the science of language (Diksha et al., 2020). NLP is a sequential modelling problem since the temporal information is important to properly understand the context of certain word within an utterance.

Intent Detection (ID) task is considered as a multi-class classification problem where the model needs to understand the features extracted from the utterance passing through a sequence model often known as encoder and determine the intent class. The whole process is called sequence classification. On the other hand, Slot Filling (SF) task is often known as token classification problem or sequence labelling problem. Compare to ID, the order and dependencies between tokens are more important in SF. SF can be seen as a similar problem as Named Entity Recognition where each word (token) is tagged, the differences lie within the tags used in SF is more diverse and specific depending on the problem domain. The tags assignable to the tokens are called Inside-Outside-Beginning (IOB) tags, tokens that is not important are marked with outside ‘O’ tag, the beginning tag is marked with ‘B-’ at the start of the tag text. For example, an utterance “Create event name Japan Trip” shown in **Figure 1** below, consider “Japan Trip” is entity of event for every beginning of the tag ‘B-’ will be added in front. The inside ‘I-’ tag is place in front of tags the happen consecutive after the entity beginning tag, hence called the inside tag. For SF the output and the input will need to in same length.

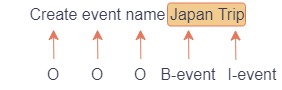


Figure 1 IOB Slot Tag Example

## **2.2 Sequential Architecture**

Over the years, various sequential architecture had been proposed to solve sequential data problem. The earliest sequential architecture is the RNN architecture proposed to learn temporal information which regular ANNs are not capable of doing so. However, RNN is prone to vanishing gradient problem (forgetting word dependencies), hence subsequent recurrent-based architectures such as GRU and LSTM were proposed to mitigate the problem. Both GRU and LSTM using RNN as backbone.

### *2.2.1 Artificial Neural Network (ANN)*

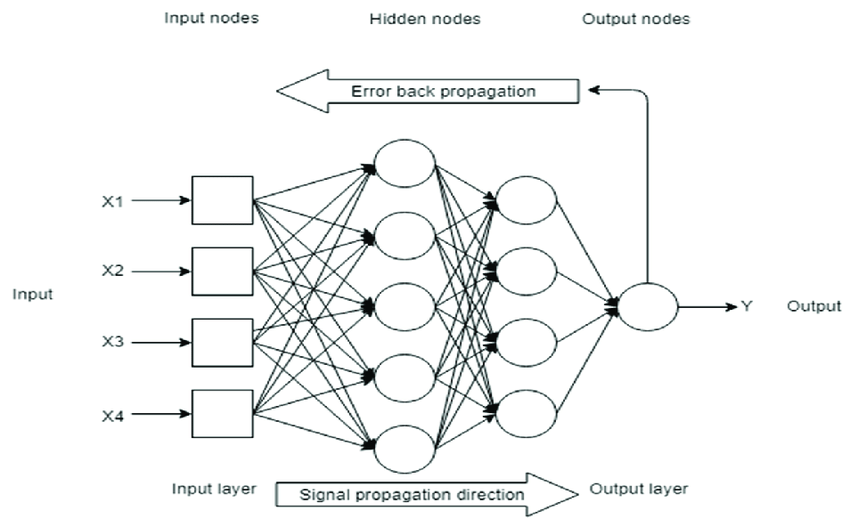


Figure 2 ANN Architecture (Azlah et al., 2019)

While ANN is not a sequential model it is important to understand how it works to understand the underlying structure of the recurrent models. **Figure 2** above (Azlah et al., 2019) illustrates the architecture of an ANN, which comprises input, output, and hidden layers. In each of the layer have a number of neurons which having their own respective weights, the product of the neurons input and weight of each neuron is then summed together including a bias term and the final value will be activated by the activation function to introduce non-linearity. The equation can be denoted as , where denotes the bias term, is the activation function, is the sum of product of output from previous neurons with their corresponding weight. After the output layer, a loss function will be used to compute the loss for the optimizer such as gradient descent to compute gradient for backpropagation. The equation for gradient descent is denoted as where is the old weight, denote learning rate, and denote the gradient obtain from computing the derivative of loss function with respect to previous weight.

### *2.2.2 Recurrent Neural Network (RNN)*

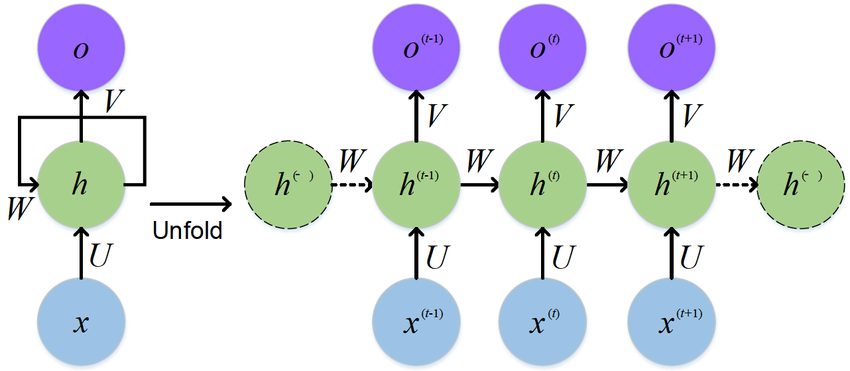


Figure 3 RNN Architecture (Feng et al., 2017)

RNN is the first sequential model introduced to solve sequential data problem. The motivation of RNNs were to capture temporal dependencies for data that exhibit sequence pattern such as speech, text and time series data (Schmidt, 2019). **Figure 3** above shows the RNN architecture, a RNN is basically a multiple ANN architecture which contain a hidden state mechanism that forward information of previous words to current time step to affect the output. Let consider a sentence with 3 words and the current time step is 2 which represent the second word denoted as . The computation of is represented as equation (Xiao & Zhou, 2020) where and is the weight for current timestep and hidden state weight for last hidden state and is the bias term for current timestep. Basically, each is the hidden layer where it connects the whole input sequence allowing previous features to be past to the future for prediction. There is also a variation of RNN that utilize future information as well, the architecture is call Bi-directional RNN. While RNN was capable of solving sequential problem it is prone to vanishing gradient problem when there are long dependencies between words as input sequence became long it is hard to conduct backpropagation.

### *2.2.3 Long Short-Term Memory (LSTM)*

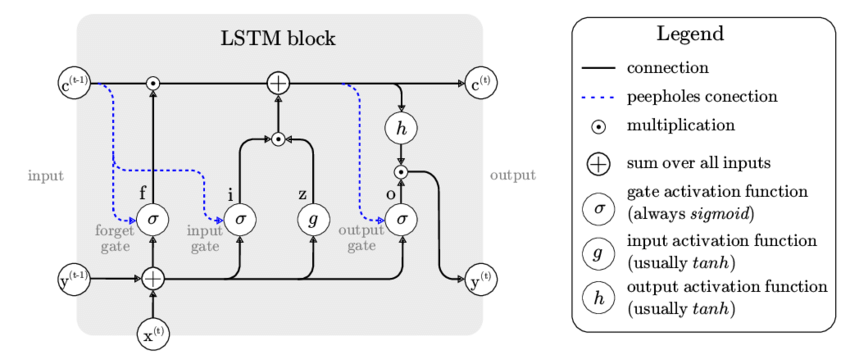


Figure 4 LSTM Architecture (Houdt et al., 2020)

LSTM is one of the recurrent-based architectures based upon the RNN, the development of LSTM architecture was motivated by resolving the diminishing gradient issue within the RNN architecture (Staudemeyer et al., 2019). **Figure 4** (Houdt et al., 2020) shows the architecture of a LSTM cell. The LSTM architecture have a memory cell that passes through the full input sequence to help the model to remember important information in each of the time step (Houdt et al., 2020). There are three gating mechanisms in LSTM architecture, namely, forget gate denoted as , input gate denoted as , and output gate denoted as . First of all, the forget gate is to help the architecture to decide the information in the cell state that should be forgotten . The forget gate is computed as followed , where is the last layer hidden state, is the current timestep’s input, denotes the concatenation of last layer hidden state and current input, is the forget gate weights for the last hidden state and current timestep’s input, denote the bias term and is the notation for the sigmoid activation function. The input gate and output gate also have similar notation and respectively. The input gate is utilized to determine the significant information requiring updates to the cell state. Whereas, the output gate will be used for computing the hidden state for the current input step. These gating mechanisms will be used to first compute the new cell state = \* + where is the current candidate for the cell state which can be computed by . After the new cell state is computed it will be used to compute the current hidden state denoted as . In some cases, peephole connection will also be introduced to the gating mechanism to provide additional information from the cell state where the notation of the input, output, and forget gate will then be denoted as , and .

## **2.3 Transformer Architecture**

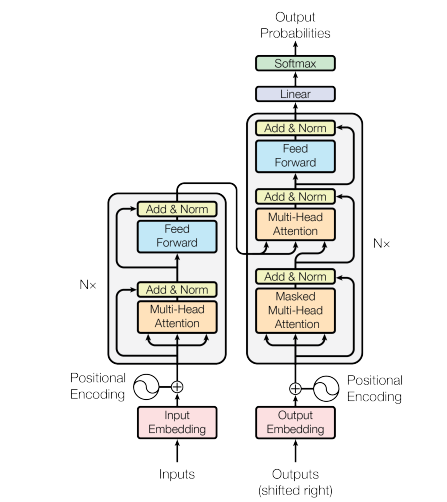


Figure 5 Transformer Architecture (Vaswani et al., 2017)

Recurrent-based architecture had dominated the NLP field for a long period of time, but the diminishing gradient problem still persist even though architecture like LSTM and GRU have mitigated the problem to some extent. Currently, the transformer (Vaswani et al., 2017) have shown to be excel in long term word dependency than recurrent-based architecture, allowing long text processing such as document summarization to be done more efficiently. Other than that, recurrent-based architecture is restricted from parallelism as the previous words need to complete processing before the current word can be processed. Transformer architecture on the other hand uses attention mechanism which process the words independently allowing parallelism more efficiently.

As of the time of writing this paper, transformer is currently the best model for solving sequential data especially for NLP problem. Various LLM have been using the transformer architecture as building block, such as BART (Lewis et al., 2019). **Figure 5** above (Vaswani et al., 2017) illustrate the framework for the transformer model. The transformer framework consists of two primary layers: the encoder and the decoder layer (Vaswani et al., 2017). The encoder layer is used for encoding the input sequence into contextual features representation to be used for downstream task. The decoder of transformer is flexible, in essence that layers (head) can be added on top of it for specific downstream task. For instance, **Figure 5** above(Vaswani et al., 2017) shows the decoder being used for next word prediction but for other task such as sequence classification a classification head can be added at the end of the decoder output.

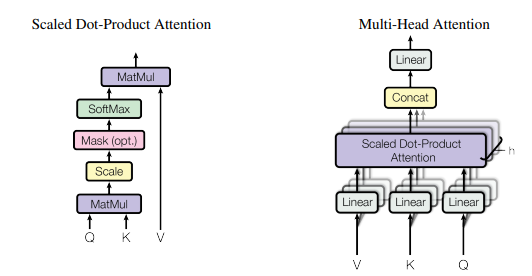


Figure 6 Self-attention Mechanism (Vaswani et al., 2017)

The self-attention stands as the primary mechanism that significantly contributes to the transformer’s ability to capture long-range dependencies within sequences. **Figure 6** (Vaswani et al., 2017) above shows the architecture of the self-attention. The self-attention layer comprises three components: Query (Q), Key (K), and Value (V), as proposed by Vaswani et al., (2017). These components are utilized to calculate the self-attention , which represents compute the features representation of a word, as described by Vaswani et al., (2017). As a loosely analogy, query can be thought as “what to attend to” to understand what should be focused on from other words in sentence to understand the current word better. The key acts as an identifier for each word in the sentence providing the information of how important the current word is to other words or more specifically to the context of the query. The value component contains the factual features of the present word, typically is the word embeddings. The multi-head attention consists of several layers of self-attention heads denoted as , so self-attention was computed multiple time in multi-head attention.

## **2.4 Word Embeddings**

There are two types of embeddings: non-contextual embeddings and contextual embeddings. Non-contextualize embeddings are static, meaning the meaning of the word does not change depending on the context. For instance, a word that can represent different meaning will have the same vector, for example a word “bank” can represent financial institution or edge of a river based on the context of the sentence. However, despite the limitation non-contextual embeddings are still valuable as they help to capture semantic similarities between words so that words that are similar tend to be close together in embedding space. These embeddings are also less computational expensive and they may be more efficient than contextualize embeddings in some cases.

Nevertheless, contextual embeddings are shown to be provide better representation of the semantic features of words (Miaschi & Dell'orletta, 2020). Example of contextual embeddings includes ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) . Contextual embeddings are dynamic in the sense that the word embeddings are generated on runtime. ELMo leverage a bi-direction LSTM (Bi-LSTM) architecture for its pre-trained model. Where the model was trained to generate contextual word representations (Peters et al., 2018). Similarly, BERT also leverage deep learning architecture, specifically BERT follows the encoder architecture of the transformer model as building block (Devlin et al., 2019). These models can be fine-tuned, however most of the time they are used for transfer learning, since they were trained on large amount of dataset and contains millions of parameters.

### 2.5 Sequence Labelling Algorithms

Sequence labelling algorithms are useful by modelling the probability of sequence of labels, they were applied in various named entity recognition system. Some of the popular sequence labelling algorithms includes conditional random field (CRF) and hidden arkov model (HMM). The HMMs are generative models that helps to identify the most likely sequence labels for words in any given sequence (Wallach, 2004). CRFs are discriminative models and are more complex that HMMs as they compute the probability of the labels not only by previous one, hence they usually perform better than HMMs by avoiding the label bias problem and weakness in maximum entropy present in HMMs models (Wallach, 2004). Overall, the CRF identifies the label sequence with highest score by considering the compatibility between the labels.

# **3.0 Related Works**

Motivated by the need to build a task-based dialog system that are robust to real-world dataset and to display the business values of task-based dialog system in closed domain applications, this work focuses on exploring different approaches for building an architecture that can solves IDSF tasks effectively on both benchmark dataset and custom dataset collected to simulate real-world use cases. In this chapter recent works and techniques used on IDSF that inspired the development of the proposed methodologies of this project will be discussed.

## **Joint Intent Detection and Slot Filling**

Recent works focused on development of joint learning architecture (Hui et al., 2021; Huang et al., 2022; Rafiepour & Sartakhti, 2023) as they perform better than independent learning methods. One of the earlier works proposed a JointBert model (Chen et al., 2019) which fine-tune pre-trained BERT model for joint learning IDSF tasks by including two classifier layers one for intent classification and another for slot filling which are just simple linear layer to transform the input (features representation output by Bert) into dimension compatible with the number of labels to compute loss for the intent classification and slot filling respectively (Chen et al., 2019). For the intent classification cross entropy loss function was used and conditional random field (CRF) was used to compute the slot loss (Chen et al., 2019). The loss of the intent detect and slot filling are then summed together as the total loss for optimization (Chen et al., 2019). Other than that, Bert uses word piece tokenization which may cause the sequence to have different length than the slot labels length, in order to align the sequence and labels they proposed to only use the first word of the sub words to compute the loss during inference time (Chen et al., 2019).

A work by Qin et al., (2020a) proposed a novel co-interactive (CI) module for joint intent detection and slot filling learning. The module contains 3 components: A label attention component, a CI-attention layer, and a feed-forward network (Qin et al., 2020a). The label attention mechanism first helps to extract the intent and slots semantic features, then it will be pass into the CI-attention layer. The CI-attention layer is the main component that compute the interaction for the IDSF tasks by generating an “intent-aware slot representation” and a “slot-aware intent representation” by feeding the intent representation obtained from the label attention component as query and the slot representation obtained from the label attention component as key and value to compute the intent-aware slot representation and similar process is done for computing the slot-aware intent representation (Qin et al., 2020a). Then the and will be passed to the feed-forward layer to produce the hidden states to be used for computing logits for the intent and slot filling tasks.

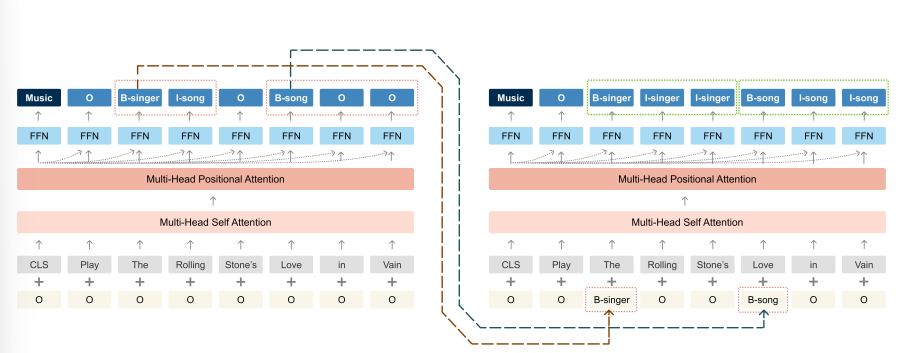


Figure 7 SlotRefine Architecture (Wu et al., 2020)

SlotRefine (Wu et al., 2020) was also one of the architectures proposed to solve joint learning IDSF task. The proposed SlotRefine followed the orginal Transformer encoder architecture (Vaswani et al., 2017) as shown in **Figure 7** (Wu et al., 2020) above. As the model produce the hidden state values for each of the words after passing through the attention mechanism, they use the to compute the logits of their intent detection task because the last hidden state of [cls] is typical use for sequence classification task but here they used it to represent the intent classification task instead of sequence classification. For the slot filling they joint the information of the intent representation before feeding-forward computation by concatenating the with where is the current word representation (hidden state) (Wu et al., 2020). Their method enabled parallel computation by utilizing the attention mechanism but can cause the uncoordinated slots problem (“B-singer” followed with “I-song” which is incorrect IOB format) (Wu et al., 2020) because their of difficulty in capturing dependency information between the slot labels, so they proposed a two-pass refine mechanism (Wu et al., 2020) which the first pass output slot label with beginning “B-” tag will be pass as input on the second pass to force the model to learn the corresponding “I-tags” (Wu et al., 2020).

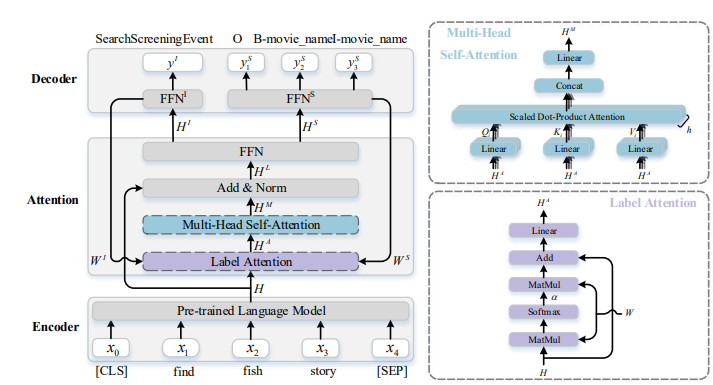


Figure 8 FAN Architecture (Huang et al., 2022)

A fast attention network (FAN) (Huang et al., 2022) was proposed for joint IDSF task. The motivation of the proposed architecture is to be able to run inference quickly on edge devices so that the model can be used in real time devices. The diagram presented in **Figure 8** (Huang et al., 2022) shows the comprehensive architecture of the proposed FAN framework. The FAN architecture has 3 main components: the encoder layer, attention, and the decoder layer. The encoder layer is just an interchangeable pre-trained LM such as BERT, DistilBERT, or TinyBERT used to generate contextualize word embeddings for the sequence. The attention layer contains two attention mechanism: the label attention and the multi-head self-attention (Huang et al., 2022). In the label attention, the tokens’ features representation was used to obtain the label representation (Huang et al., 2022). Basically, the label attention layer uses the respective weight from the decoder’s fully connect layer (FFN) which are then concatenated to form , where represent the weight obtain from the FFN of the intent and from the weight of FFN for slot (Huang et al., 2022). The is the multiply with the feature embedding generated from the encoder and proceed the step shown in the bottom right of **Figure 8** to obtain (Huang et al., 2022).Then the output from the label attention is fed into the multi-head self-attention layer which map the to the query, key, and value matrices (Huang et al., 2022). A two-layer FFN is used at the top of the multi-head attention after the add & norm block to represent the features into the intent features and slot features and finally the decoder module also contains two FFN to project the intent and slot features for softmax computation (Huang et al., 2022).

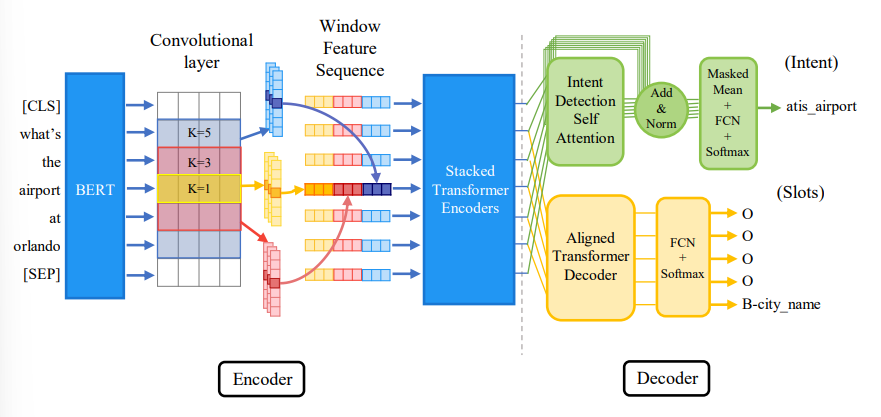


Figure 9 Ctran Architecture (Rafiepour & Sartakhti, 2023)

A convolutional transformer architecture proposed by Rafiepour & Sartakhti (2023) has achieved a strong performance in their experiment. **Figure 9** (Rafiepour & Sartakhti, 2023) above shows their model’s architecture. In their experiment they utilize a convolutional layer and a stacked transformer encoders layer as their encoder layer (Rafiepour & Sartakhti, 2023). They have two separate classifiers for ID and SF respectively in their decoder layer. The convolutional layer was set to have 4 kernels with size of 5, 3, 2, and 1 to capture the contextual features of context range of 5 words, 3 words, 2 words, and 1 word. Instead of max pooling they use a mechanism call window feature sequence which concatenate the contextual features produced by each of the kernel to retain the word orders (Rafiepour & Sartakhti, 2023). The stacked transformer encoders were used to generate a new representation that better emphasize on neighbouring words (Rafiepour & Sartakhti, 2023). For their decoder intent classifier, a self-attention component was used following a add & norm block, a linear FFN as projection for adjusting the dimension for softmax computation (Rafiepour & Sartakhti, 2023). The slot classifier on the other hand starts with an aligned transformer decoder that masked the sub words to align with the slot labels, after that the output also pass-through linear feed-forward network and computed with softmax (Rafiepour & Sartakhti, 2023).

## **3.2 Transfer Learning**

In the experiment by Rafiepour & Sartakhti (2023) pre-trained BERT model was use to extract word embeddings for the input sequence in the encoder layer of their proposed architecture shown in **Figure 9** (Rafiepour & Sartakhti, 2023) above. The BERT model was pre-trained on large amount of data and its’ parameters were freeze during the training process as a form of transfer learning (Rafiepour & Sartakhti, 2023). ELMo which is another popular pre-trained model for extracting contextual embedding was also used in their experiment (Rafiepour & Sartakhti, 2023). In the work by Wu et al., (2020), they also utilized BERT model to improve their proposed SlotRefine model, which had significantly improved their model performance. They also conclude that the reason that BERT could improve their model performance was because of the benchmark dataset contains a lot of out of vocabulary words where BERT excel in contextualize these words to provide meaningful representation (Wu et al., 2020).

There are several pre-trained language modes that can be utilized as transfer learning for IDSF tasks other than BERT as well. While the original BERT has brought significant improvement in various IDSF models (Wu et al., 2020, Qin et al., 2020a) the model is quite large in size with millions of parameters (Huang et al., 2022). Hence, various effort has been taken to compress the model while hoping that the performance can be retained. ALBERT, a variant of BERT, integrates factorization techniques into embedding and implements parameter sharing to dimmish the amounts of parameters in BERT. However, it doesn’t significantly lower the hidden size or the number of layers within the transformer block, thus still necessitating substantial computational resources (Huang et al., 2022). DistilBERT is another variation of BERT that smaller in size by 40% and 60% faster than BERT while retraining 97% performance of BERT (Huang et al., 2022). It was pre-trained with distillation technique. TinyBERT stands out as the most compact BERT variant, featuring merely four transformer encoder layers. This configuration achieves a smaller size (7.5x) and faster speed (9.4x) compared to BERT, all while preserving over 96.8% of its performance (Huang et al., 2022).

## **3.3 Evaluation Metrics**

Accuracy is the most popular evaluation metric for classification task, it was used for intent detection in previous works (Chen et al., 2019; Wang et al., 2020; Qin et al., 2020; Huang et al., 2022; Rafiepour & Sartakhti, 2023) as well since ID is considered as a classification task. On the other hand, F1 score metric was used for evaluating slot tagging task (Chen et al., 2019; Wang et al., 2020; Qin et al., 2020; Huang et al., 2022; Rafiepour & Sartakhti, 2023) as it provides balance mean between the precision and recall. The F1 score is denoted as . These metrics provide a good evaluation for an IDSF system’s performances.

## **3.4 Benchmark Datasets**

ATIS (Hemphill et al., 1990) and SNIPS (Coucke et al., 2018) datasets are popular benchmark datasets used for benchmarking IDSF model’s performance (Chen et al., 2019; Wang et al., 2020; Qin et al., 2020; Huang et al., 2022; Rafiepour & Sartakhti, 2023). ATIS have 3 sets of data the training, development and testing set containing 4,478, 500, and 893 records respectively (Hemphill et al., 1990). Similar to ATIS, the SNIPS dataset also has 3 sets of data the training set contain 13,084 records, the development set contain 700 records, and the test set contain 700 records (Coucke et al., 2018). In order to solve multi-intent classification problem two popular benchmark datasets were used widely (Qin et al., 2021; Tu et al., 2023). The MixATIS dataset contains 13,162, 759, and 828 records for training, validation, and testing set respectively (Tu et al., 2023). The MixSNIPS dataset on the other hand, have 39,776, 2,198, and 2,199 records training, validation and testing sets respectively (Tu et al., 2023). The MixATIS and MixSNIPS datasets are much challenging and have more records than the original single intent ATIS and SNIPS dataset.

### **3.5 Related Works Models Performance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Models** | **SNIPS** | | | **ATIS** | | |
| **Intent** | **Slot** | **Sent** | **Intent** | **Slot** | **Sent** |
| Joint BERT (Chen et al., 2019) | 98.6 | 97.0 | 92.8 | 97.5 | 96.1 | 88.2 |
| Co-int Transformer (Qin et al., 2020a) | 98.8 | 95.9 | 90.3 | 97.7 | 95.9 | 87.4 |
| SlotRefine (Wu et al., 2020) | 99.04 | 97.05 | 92.96 | 97.74 | 96.16 | 88.64 |
| BERT-FAN (Huang et al., 2022) | 98.3 | 97.1 | 93.0 | 97.8 | 96.1 | 88.7 |
| Ctran (Rafiepour & Sartakhti, 2023) | 99.1 | 98.3 | - | 98.1 | 98.5 | - |

Table 1 Reviewed Works’ Model Performance

### **3.6 Related Works Experiment Settings**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Models** | **Settings** | | | | | |
| **Batch** | **Max Len** | **Epochs** | **Optimizer** | **Hidden size** | **LR** |
| Joint BERT (Chen et al., 2019) | 128 | 50 | 40 | Adam | 768 | 5e-5 |
| Co-int Transformer (Qin et al., 2020a) | 32 | 32 | 100 | Radam | 128 | 0.001 |
| SlotRefine (Wu et al., 2020) | 32 | - | 100 | - | - | 0.001 |
| BERT-FAN (Huang et al., 2022) | 32 | 50 | - | Adam | 768 | 5e-5 |
| Ctran (Rafiepour & Sartakhti, 2023) | 16 | 60 | 50 | AdamW | 512 |  |

Table 2 Reviewed Works’ Experimental Settings

# **Methodology**

In this chapter the methodology used for this project will be discussed. The full model architecture will be discussed in details in section 4.1 then the subsequent section will discuss about the sub components that are used to build the proposed architecture in details.

## **4.1 Model Architecture**

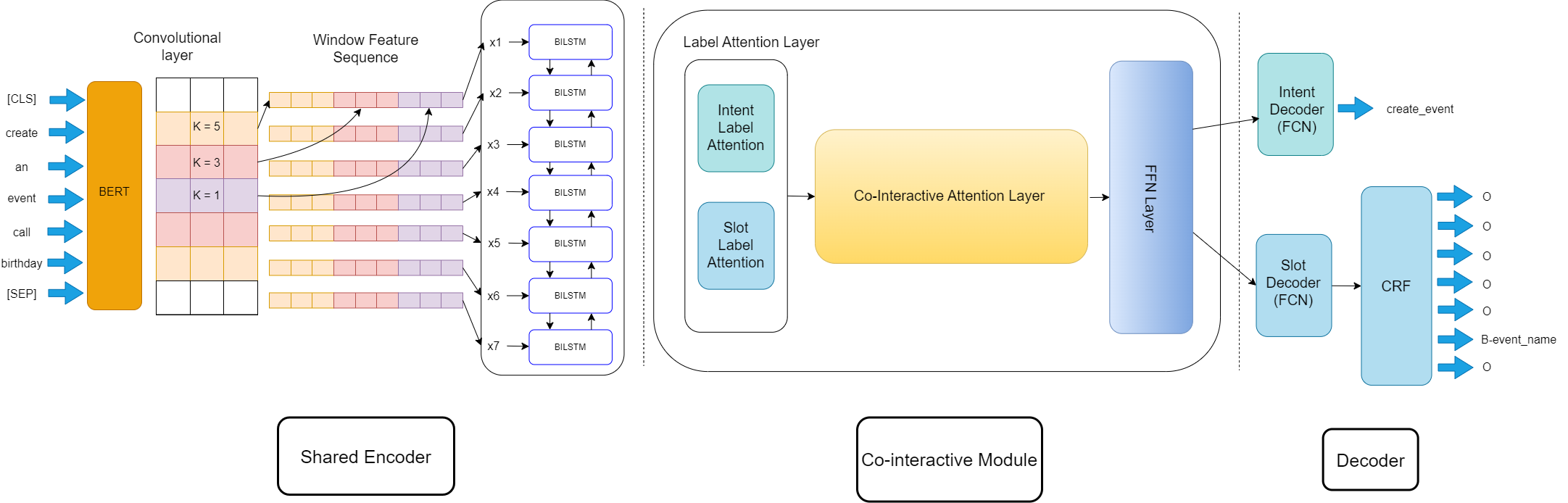


Figure 10 Proposed CoinBert Architecture

The proposed architecture of this project follows the transformer architecture as backbone containing encoder and decoder layer. **Figure 10** above shows the full proposed architecture for this project. The proposed architecture has 3 main components: shared encoder, co-interactive module (Qin et al., 2020a), and a decoder. The shared encoder is responsible for extracting useful feature representations from texts in the vector space then passed to the co-interactive module (Qin et al., 2020a) to utilize the feature representations to generate intent-aware slot representations and slot-aware intent representations to facilitate intent and slot information. Finally, the decoder will contain compute the logits for the IDSF tasks and compute their joint losses. Overall, we try to design the model to be able to run on less resource intensive environment taking into consideration the limited processing power available for running this experiment. More details regarding the setup of this experiment will be discussed in section 6.1 below.

## **4.2 Shared Encoder**

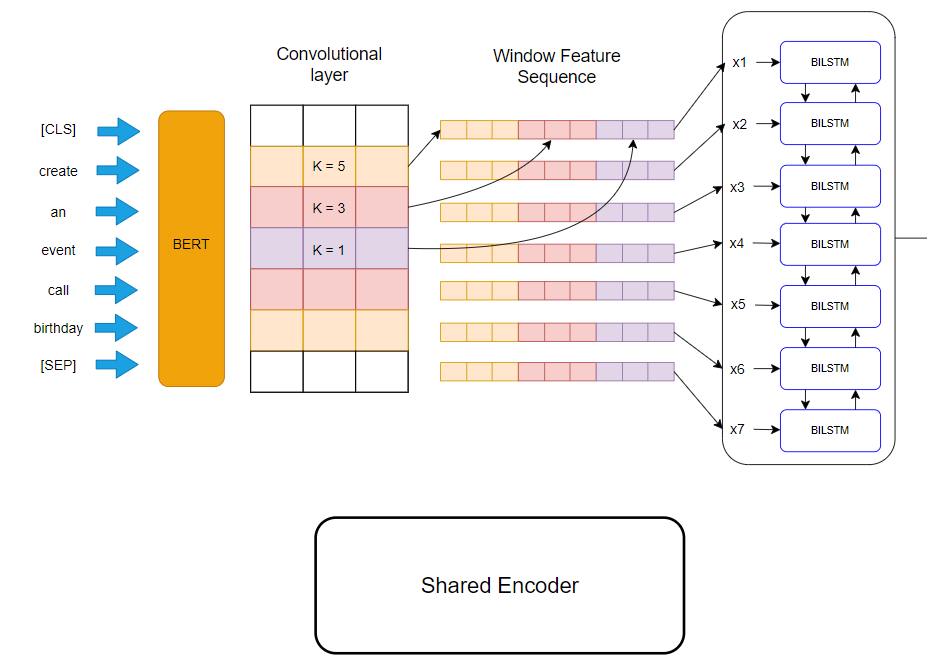


Figure 11 Shared Encoder component of CoinBert

**Figure 11** above shows the shared encoder component of our proposed model. Within the shared encoder component there are 3 main components, namely, the BERT model, the convolutional layer, and the Bi-LSTM model.

**BERT**. The BERT model (Devlin et al., 2019) parameters will be freeze as a form of transfer learning for extracting contextual embedding for the tokens. Note that BERT model requires the input sequence to include [CLS] token and [SEP] tokens to be added at the start and at the end of the input sequence, so in order to align with the slot labels, the slot labels for the input sequence are append with the respective slot labels that mark the beginning of sequence and end of sequence during the pre-processing stage.

**Convolution Layer**. Once the BERT generates the word embeddings for the input sequence, the embeddings will be sent to the convolutional layer (Rafiepour & Sartakhti, 2023) which contains 4 kernels with size of 5, 3, 2, 1 respectively. The main purpose of the convolutional layer is to emphasize the local dependency between the words in the context window of 5, 3, 2, and 1. Following Rafiepour & Sartakhti (2023) the window feature seqeunce will also be applied in our model to concatenate the features extracted by the kernels and retain word order (Rafiepour & Sartakhti, 2023).

**Bi-LSTM**. The Bi-LSTM model which consist of two LSTM layers uses the features from the convolution layer and further extract the features for the whole input sequence, this time emphasizing the global context to produce context-sensitive hidden states.

## **4.3 Cross Attention Module**

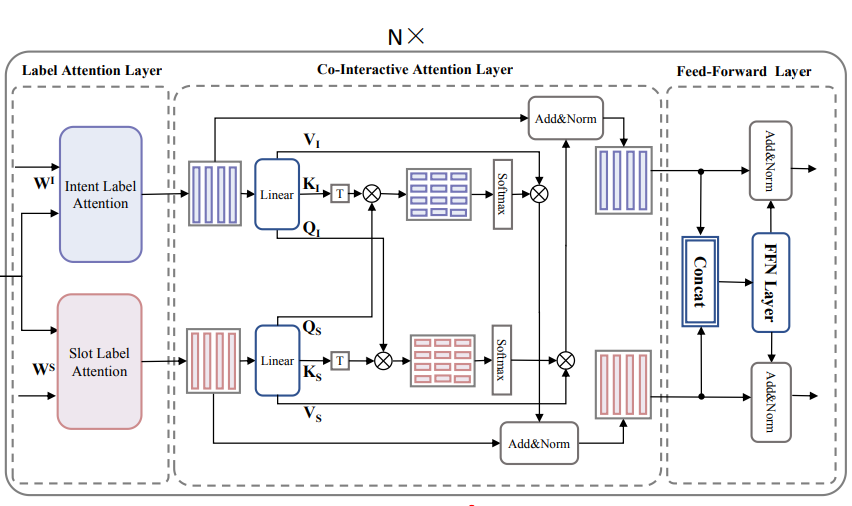


Figure 12 Co-interactive Module (Qin et al., 2020a)

The proposed model uses the co-interactive module mechanism proposed by Qin et al., (2020a) for the joint learning of IDSF. The co-interactive module is made up of 3 components: the label attention, co-interactive, and feed-forward layers. **Figure 12** above shows the co-interactive module borrowed from the work by Qin et al., (2020a) for our proposed model.

**Label Attention Layer**. The label attention layer was inspired by Cui & Zhang (2019) to the work by Qin et al., (2020a) which proposed a label attention network for sequence labelling task. The label attention layer by Qin et al., (2020a) is composed of two attention mechanism, namely, the intent-label attention and slot-label attention mechanism to obtain the **intent representation** and **slot representation** (Qin et al., 2020a). Particularly, a fully-connected classifiers parameters which can be thought as the distributions of the intent and slot labels were used as the intent and slot embedding. (Qin et al., 2020a). Then the embeddings were fed into the self-attention to get the representations. Particularly, the full formula for the label attention was defined as (Qin et al., 2020a), is computed as where is treated as the query component, is the intent or slot embedding and is treated as the key and value components. Remember the self-attention mechanism from **Figure 6** above the was considered as the output of the softmax of the Q and K component then is essentially the matrix multiplication of the softmax output of Q and K with the V, so the equation actually denote the full self-attention mechanism.

**Co-interactive (CI) attention layer**. The CI attention layer computes the **intent-aware slot representation** and **slot-aware intent representation** (Qin et al., 2020a) by using the labels representation produced from the label attention layer. Basically, this layer follows the multi-head attention computation (Vaswani et al., 2017) which the original formula was denoted as , here the equation was modified to , where and (Qin et al., 2020). Then the respective intent-aware slot representation and slot-aware intent representation will be pass to a add & norm block which denoted as , where is the layer normalization function.

**Feed-forward Network layer**. The feed-forward network layer will first concatenate the and to combine their information (Qin et al., 2020a). Then the concatenated representation will be fed into a FFN and the output of the FFN will be use by the respective add & norm blocks (Qin et al., 2020a).

## **4.4 Decoder**

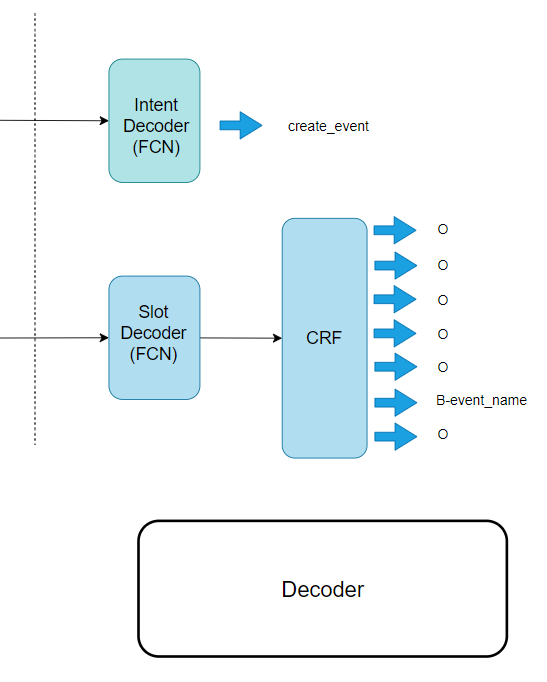


Figure 13 Decoder Component of CoinBert

**Figure 13** above shows the decoder component of our proposed model. The decoder component contains two classifiers: intent decoder and slot decoder which uses fully connected layer with output dimension compatible with the respective number of labels to compute the logits.

**Intent Decoder.** Within the intent decoder a max-pooling operation (Kim, 2014; Qin et al., 2020a) will be applied onto the intent hidden states produced from the co-interactive module to obtain sentence representation for computing the intent distribution using softmax function. The most probable intent obtained from the softmax distribution will be the predicted intent.

**Slot Decoder.** For the slot decoder, the slot hidden states produced from the co-interactive module will be applied with Conditional Random Field (CRF) layer to model the dependency between labels (Qin et al., 2020a). The output of the CRF will be used for computing the negative log likelihood as the loss of the slot (Qin et al., 2020a). Additionally, as this project uses BERT for the word embeddings there may be situation where sub words cause the length of input sequence to not align with the slot labels, because BERT uses WordPiece tokenizer (Devlin et al., 2019). Inspire by Chen et al., (2019) to align the sequence and the slot labels during inference we only uses the first sub word to do prediction by assigning subsequent sub words with label index of -100 where it will be ignored during loss computation. We create a subtoken mask that mark all the input indices with 1 and 0 for those having index of -100 as their slot label and use this mask to re-arrange the sequence by assigning those masked with 0 to have index 0 and moving them to the end of the sequence treated as padding. During inference time, the padding will be ignored hence only the sequence and the slot labels will then be aligned.

## **4.5 Data Processing**

In order for the ease of labelling slot tag, the benchmark dataset ATIS (Hemphill et al., 1990) and SNIPS (Coucke et al., 2018) had removed punctuations in their default dataset. Hence, the only processing steps that needed to be taken care of when using the benchmark dataset are only to tokenize the sequence, generate padding and generate mask for the Bert Model. We use the Bert tokenizer in the huggingface library to helps tokenizing the input sequence, this process also generates the attention mask which will be used by the Bert Model to identify which token need to be ignored such as the padding token. However, as previously mentioned the sub words may cause misalignment between the sequence and slot labels, hence during pre-processing the sub words are labelled with a temporal slot tag with index of -100 which will be ignored during inference time. After that, the processed data will be converted into tensor which is a data structure in pytorch for gradient computation and the tensors will be store into a tensordataset class to be loaded in DataLoader for batch processing.

For the custom dataset for this project, it uses similar processing procedure as the benchmark dataset, however the custom dataset consists of data like date and time which contains punctuations so an extra step of removing the punctuations is taken when processing the dataset. The punctuations of the date and time are removed, and the numbers are combined together as a single word matching the slot label.

# **Dataset**

In this chapter the justification of the selected benchmark datasets is discussed. Other than that, the data collection and annotation processes will also be discussed in details in this chapter.

## **Benchmark Dataset Selection**

There are several benchmark datasets available for IDSF tasks since the difficulty can varies based on the requirements, for instance, intent detection typically is done to identify one detect per utterance but in some cases, there may be multiple intent per utterance which is where multi-intent dataset such as MixSnips (Tu et al., 2023) and MixAtis (Qin et al., 2020b) would be used. For the purpose of this project a single intent dataset would be sufficient, hence the chosen datasets are ATIS (Hemphill et al., 1990) and SNIPS (Coucke et al., 2018). These datasets were selected because

1. They are easy to understand and apply to our experiment
2. Our custom dataset is also a single intent dataset
3. Previous works used these datasets so using them enable us to benchmark our model more easily

The ATIS dataset contains 4,478, 500, and 893 records for training, development and testing. Whereas, the SNIPS dataset contains 13,084, 700, and 700 records for training, development and testing. The ATIS dataset have 26 unique intent classes and 126 unique slot tags, where as SNIPS dataset have 6 unique intent classes and 72 unique slot tags

## **5.2 Custom Dataset Collection & Annotation**

### *5.2.1 Data Collection*

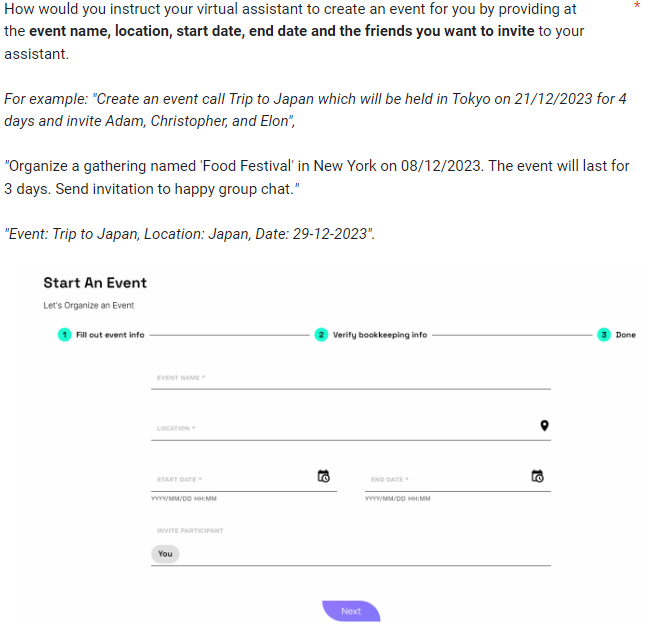


Figure 14 Sample Survey Form’s Question

The custom dataset data were collected by distributing survey form and interview. An application interfaces were designed using Figma software to simulate a real-world application that allow users to create and manage their events. The basic functionalities of the application include adding event, recording expenses, inviting friends to join the event, settle up expenses, accepting and rejecting the event invitations, displaying event details. The interfaces were then added into the survey form to provide respondents better understanding on the question. One of the survey questions samples is shown in **Figure 14** above. The goal is to collect data on how user would interact with their virtual assistant on performing the **12** **intent tasks**, namely, create event, create expense equally, create expense unequally, invite participant, accept invitation, reject invitation, settle expenses, show event information, simplify expenses count, show debtors and payers, show debtors only, and show payers only. In order to perform the intent tasks **75 slot tags** will be required to obtain the important keywords.

The data collection was done by distributing survey form and performing interviews for span of a month. For this project, our target population is but not limited to people in Malaysia and the respondents were not restricted in any manner (age, occupation, nationality) because the questions are very general where there is not right or wrong answer. The survey form was circulated across diverse social media platform, including WhatsApp, Facebook, Discord, and Reddit. Whereas, interview was conducted to whoever physically available. The data collection was done for a span of a month. However, despite the effort the data collected was very limited because we only receive 22 responses for the survey and 8 responses for the interview. The data obtained from the 30 responses are not sufficient so manual data augmentation was conducted to increase the sample size. With the help of ChatGPT and Google Gemini conversation agents, we first manually create some sample through using similar linguistic with the obtained response as well as feeding the obtained responses and manually augmented sample to the conversation agents and do modification manually if required for the generated samples by the conversation agents to increase sample size. Finally, the we were able to produce a sample size of 3132 records, where each intent have at least 100 samples. The sample was split into 2501 records for training, 317 records for testing, and 314 records for validation.

### *5.2.2 Data Annotation*

To ensure that the data is annotated properly manual annotation method was used. The data were first separated by intents then python coding is used to remove unwanted punctuation so that the annotation process can be done more smoothly. However, unlike the benchmark dataset, the custom dataset contains information such as date and time which the punctuation should not be removed for the initial slot annotation process, hence with the help of python library call dateutils (Parser — Dateutil 2.8.0 Documentation, 2012) we first identify whether the word is date or time and if it is date or time the punctuation removal will not be done on it. Once the punctuations were removed, we create another algorithm to tag all the token with “O” slot tag and write them to text file. After the initial tagging is done, we manually replace the “O” tag with the respective slot label if the word is a keyword. The data annotation process took about a month to complete.

# **Experiment & Evaluation**

In this chapter the details implementation of the proposed model for this project will be discussed. The evaluation metrics used will also be discussed in this section.

## **Setup**

For our experiment we did 2 experiments on 3 different datasets: ATIS, SNIPS, and Custom Dataset. The experiments done includes:

(1) CoinBERT without convolution layer

(2) CoinBERT with Convolution layer

With the time constraint and limited computational power the proposed model was designed to be small and effective. The full experiment was done on Google Colab using the T4 GPU 15GB, with SNIPS dataset having the largest sample size completed within 3 hours. To keep the experiment replicable and consistent, we set the seed so that everytime the experiment is run, it will produce the same result if the hyperparameter is not changed. All the experiments done on all the dataset used the same GPU device and same seed for a fair comparison. While both of the models can be run on CPU, it is recommended to use GPU as it is much faster. The settings and hyperparameter is the similar for all models and the datasets except the batch size due to Snips data having much larger sample size. The hyperparameters are shown in **Table 3** below. The CoinBERT with convolution layer and without convolution layer have around 2million and 3million trainable parameters respectively.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | Batch Size | Hidden Size | Max Length | Bert Dropout | Emb Dropout | Optimizer (Radam) | | | Attention Heads |
| LR | Gama | Step ep |
| Atis | 32 | 256 | 100 | 0.3 | 0.5 | 0.001 | 0.3 | 40 | 8 |
| Snips | 128 | 256 | 100 | 0.3 | 0.5 | 0.001 | 0.3 | 40 | 8 |
| Friendlink | 32 | 256 | 100 | 0.3 | 0.5 | 0.001 | 0.3 | 40 | 8 |

Table 3 Hyperparameter Settings

## **6.2 Benchmark Models**

We benchmark our model with the models in **Chapter 3** related works, in addition with some others popular IDSF models as shown in **Table 4** below. The intent was evaluated with accuracy, the slot was evaluated with F1 score and the sent is the overall accuracy.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Models** | **SNIPS** | | | **ATIS** | | |
| **Intent** | **Slot** | **Sent** | **Intent** | **Slot** | **Sent** |
| SF-ID Network (Haihong et al., 2019) | 97.0 | 90.5 | 78.4 | 96.6 | 95.6 | 86.0 |
| CM-Net (Liu et al., 2019) | 98.0 | 93.4 | 84.1 | 95.6 | 96.6 | 86.0 |
| Joint BERT (Chen et al., 2019) | 98.6 | 97.0 | 92.8 | 97.5 | 96.1 | 88.2 |
| Co-int Transformer (Qin et al., 2020a) | 98.8 | 95.9 | 90.3 | 97.7 | 95.9 | 87.4 |
| SlotRefine (Wu et al., 2020) | 99.04 | 97.05 | 92.96 | 97.74 | 96.16 | 88.64 |
| BERT-FAN (Huang et al., 2022) | 98.3 | 97.1 | 93.0 | 97.8 | 96.1 | 88.7 |
| CTran (Rafiepour & Sartakhti, 2023) | 99.1 | 98.3 | - | 98.1 | 98.5 | - |

Table 4 Benchmarks Models

## **6.3 Metrics**

For our experiment we will be using accuracy for evaluating our model intent detection performance and we will also be using F1 score for evaluating our model slot filling performance. First of all, accuracy was chosen for evaluating intent detection is because for our experiment ID is a single intent classification task, which accuracy can tell us how accurate the model is able to identify the correct intent class. On the other hand, F1 score provides the balances between precision and recall which is useful for imbalanced datasets which often is the case for slot tags. Other than that, previous works also used these metrics for evaluating their models, so by using the same metrics we can compare our model with theirs more easily. We also compute the sent accuracy, which measure how many time our model predict both the intent and all the slot tags correctly as the overall accuracy.

# **Results**

In this chapter the results obtained by our proposed model will be presented and be compared against the benchmark models.

## **Benchmark**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Models** | **SNIPS** | | | **ATIS** | | | **Friendlink (Custom)** | | |
| **Intent** | **Slot** | **Sent** | **Intent** | **Slot** | **Sent** | **Intent** | **Slot** | **Sent** |
| CoinBert without CNN | **98.6%** | 94.3% | 87.1% | **97.7%** | 95.1% | 85.4% | 96.5% | 93.9% | 77.6% |
| CoinBert with CNN | 98% | **94.8%** | **88%** | 97.5% | 95.1% | 85.4% | **97.8%** | **94%** | **80.5%** |

Table 5 Result of Coinbert + CNN vs Coinbert

**Table 5** above shows the results of the variations of our proposed model on the two benchmarking dataset and the custom dataset. Overall, both of our models were able to obtain a good performance. Based on the result it can be seen that the model with CNN layer was slightly better than the one without it. However, one thing to note is that the CNN version contains much lesser trainable parameters than the one without it, but perform better. Our model was able to obtain an accuracy of 97.8% for classifying intent, 94% f1 on slot filling and an overall accuracy of 80.5% on the custom dataset showing that it can learn on real-world dataset well. Other than that, the benchmark results also show good performance with both having overall accuracy above 80%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Models** | **SNIPS** | | | **ATIS** | | |
| **Intent** | **Slot** | **Sent** | **Intent** | **Slot** | **Sent** |
| SF-ID Network (Haihong et al., 2019) | 97.0 | 90.5 | 78.4 | 96.6 | 95.6 | 86.0 |
| CM-Net (Liu et al., 2019) | 98.0 | 93.4 | 84.1 | 95.6 | 96.6 | 86.0 |
| Joint BERT (Chen et al., 2019) | 98.6 | 97.0 | 92.8 | 97.5 | 96.1 | 88.2 |
| Co-int Transformer (Qin et al., 2020a) | 98.8 | 95.9 | 90.3 | 97.7 | 95.9 | 87.4 |
| SlotRefine (Wu et al., 2020) | 99.04 | 97.05 | 92.96 | 97.74 | 96.16 | 88.64 |
| BERT-FAN (Huang et al., 2022) | 98.3 | 97.1 | **93.0** | 97.8 | 96.1 | **88.7** |
| CTran (Rafiepour & Sartakhti, 2023) | **99.1** | **98.3** | - | **98.1** | **98.5** | - |
| **CoinBert – Distilbert (Our)** | 98.6% | 94.3% | 87.1% | 97.7% | 95.1% | 85.4% |
| **CoinBert – Distilbert + CNN (Our)** | 98% | 94.8% | 88% | 97.5% | 95.1% | 85.4% |

Table 6 Result of our model vs state-of-the-art

**Table 6** above shows the comparison of our model’s performance against some of the recent models, it can be seen that while our model perform well it is lacking in performance when compared with the high performance’s architectures. Our model placed 5th on the SNIPS dataset for its overall performance and placed last for the overall performance on ATIS dataset indicating that there are rooms for improvement especially on the Atis dataset. Despite our lack proposed model having lack of performance on the benchmark dataset against the benchmark models the performance our model exhibit is still satisfactory and is sufficient for achieving the objectives of this project.

# **Critical Analysis**

Based on the results obtains from the previous **Section 7**, there are several findings. First of all, both variations of our models exhibit good overall performance on the benchmark dataset, while also showing robust performance on the custom dataset which have less training sample (2501 samples) and imbalance distribution of slots labels. On the other hand, our proposed model with the CNN component which have lesser trainable parameters perform slightly better than the one without it, showing that the introduction of the CNN component not only improve only model performance but also reduce our model complexity significantly (<1M parameters).

On the flip side, our proposed model lack in performance when compared against the benchmarking models as shown in **Table 6** above. Indicating that there are much room for improvement. Nevertheless, the goal of this project was met which is to develop a IDSF model that can perform well on the benchmark dataset and on the custom dataset collected to simulate real world task-based application to display the capabilities of the system to be used in practical situation and our model has exhibit its potential by showing satisfactory results.

We have also tested our model on larger epochs size (100) to identify the limit of our model but the performance gain is often very small (< 0.05) and in cases where the validation dataset and testing dataset have different distribution (our custom dataset) due to having little sample size for splitting, the model overfit on the performance of the validation set resulting in drop of performance in the testing set. So, we conclude that the epochs size of 50 is much suitable for our experiment. Other than that, we have also tried a larger learning rate (0.01) but found that our model converges too quickly making it unable to learn properly, and lesser learning rate (0.0001) causes the model to be unable learn the representation. Our current learning rate setting (0.001) provides the best result.

Based on the results of our model, our model often achieve accuracy for ID but perform worst on SF with F1 score which is the main reason that cause our model to be worse than the benchmark models. Overall, our model is performing decently achieving overall accuracy on all three of the datasets.

# **Conclusion**

This project proposes a model that joint learning the IDSF tasks. The proposed model has exhibited a good performance that indicates that it could be used in actual applications. Our proposed model first uses a pre-trained language model by freezing its parameter as transfer learning for extracting contextual features representation, then we utilized a CNN layer containing 4 kernels with context of 5 words, 3 words, 2 words, and 1 word to extract meaningful local dependency features representation between the tokens in the sequence. The CNN have helps to reduce the parameters size of our model significant, while also slightly improved our model performances. Then we add a Bi-LSTM layer to emphasize the global dependency between the tokens, hence our model uses both the strength of CNN in capturing local features and the strength of Bi-LSTM in capturing global features. After that, our model uses the CI module for the interaction between the IDSF tasks. Then for our decoder layer we used max pooling for obtaining the sentence representation for the ID and used conditional random field to compute the loss for the SF task. Both the losses were then summed to perform joint learning.

We then benchmark our model against some of the strong performance models from recent works and found that our model is lacking performance against them. Despite that, our model still achieved a good result on the benchmarking dataset having 88% overall accuracy and 85.4% overall accuracy on the SNIPS and ATIS dataset respectively. Other than that, our model also exhibited good result on the custom dataset with an overall accuracy of 80.5%.

In conclusion, our proposed architecture had achieved the aims and objectives of this project defined in **Section 1.3** above. We have identified the co-interactive module mechanism to facilitate the joint learning strategy for improving IDSF model performance achieving the first objective. Then we also utilized the pre-trained language model as transfer learning technique to enhance our model NLU capability for the second objective. For the third objective, we have used Figma software to design interfaces to simulate a real-world task-based application for collecting dataset to test the robustness of our model in real world use cases. Lastly, we identified the convolutional methods to that have improved our model overall performances and reduced our model’s parameters significantly.

# **References**

Azlah, M. A. F., Chua, L. S., Rahmad, F. R., Abdullah, F. I., & Wan Alwi, S. R. (2019). Review on Techniques for Plant Leaf Classification and Recognition. *Computers*, *8*(4), 77. <https://doi.org/10.3390/computers8040077>

Chen, Q., Zhuo, Z., & Wang, W. (2019, February 28). *BERT for Joint Intent Classification and Slot Filling*. ArXiv.org. <https://doi.org/10.48550/arXiv.1902.10909>

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., Primet, M., & Dureau, J. (2018). Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1805.10190>

Cui, L., & Zhang, Y. (2019). Hierarchically-Refined Label Attention Network for Sequence Labeling. *ArXiv (Cornell University)*. <https://doi.org/10.18653/v1/d19-1422>

Chung, J., Gulcehre, C., & Cho, K. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Retrieved July 28, 2022, from <https://arxiv.org/pdf/1412.3555>)

Devlin, J., Chang, M.-W., Lee, K., Google, K., & Language, A. (n.d.). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. <https://arxiv.org/pdf/1810.04805.pdf&usg=ALkJrhhzxlCL6yTht2BRmH9atgvKFxHsxQ>

Dozat, T., & Manning, C. (2017). *DEEP BIAFFINE ATTENTION FOR NEURAL DEPENDENCY PARSING*. <https://arxiv.org/pdf/1611.01734.pdf>

Feng, W., Guan, N., Li, Y., Zhang, X., & Luo, Z. (2017). *Audio visual speech recognition with multimodal recurrent neural networks*. <https://doi.org/10.1109/ijcnn.2017.7965918>

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 82, 3713– 3744. <https://doi.org/10.1007/s11042-022-13428-4>

Haihong E, Niu, P., Chen, Z., & Song, M. (2019). A Novel Bi-directional Interrelated Model for Joint Intent Detection and Slot Filling. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1907.00390>

Hemphill, C., Godfrey, J., & Doddington, G. (n.d.). *The ATIS Spoken Language Systems Pilot Corpus*. Retrieved March 11, 2024, from <https://aclanthology.org/H90-1021.pdf>

Huang, L., Liang, S., Ye, F., & Gao, N. (2022). A Fast Attention Network for Joint Intent Detection and Slot Filling on Edge Devices. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2205.07646>

Hui, Y., Wang, J., Cheng, N., Yu, F., Wu, T., & Xiao, J. (2021). Joint Intent Detection And Slot Filling Based on Continual Learning Model. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2102.10905>

Kashyap, R. (2023). *A First Chat with ChatGPT: The First Step in the Road-Map for AI (Artificial Intelligence) ...* <https://doi.org/10.32388/dfe2xg>

Kim, Y. (2014). *Convolutional Neural Networks for Sentence Classification*. ArXiv.org. <https://arxiv.org/abs/1408.5882>

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L., & Ai, F. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. <https://arxiv.org/pdf/1910.13461.pdf>

Liu, J., Xiong, S., He, Y., Zhou, T., Wang, L., Li, X., & Xiao, B. (2023, June 1). *SIAST: A Slot Imbalance-Aware Self-Training Scheme for Semi-Supervised Slot Filling*. IEEE Xplore. <https://doi.org/10.1109/ICASSP49357.2023.10096302>

Liu, Y., Meng, F., Zhang, J., Zhou, J., Chen, Y., & Xu, J. (2019). CM-Net: A Novel Collaborative Memory Network for Spoken Language Understanding. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1909.06937>

Liu, Y., Zhang, K., Li, Y., Yan, Z., Gao, C., Chen, R., Yuan, Z., Huang, Y., Sun, H., Gao, J., He, L., & Sun, L. (2024). *Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models*. <https://arxiv.org/pdf/2402.17177.pdf>

Miaschi, A., & Dell'orletta, F. (2020). *Contextual and Non-Contextual Word Embeddings: an in-depth Linguistic Investigation* (pp. 110–119). <https://aclanthology.org/2020.repl4nlp-1.15.pdf>

*parser — dateutil 2.8.0 documentation*. (2012). Readthedocs.io. <https://dateutil.readthedocs.io/en/stable/parser.html>

Peters, M., Neumann, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations*. <https://arxiv.org/pdf/1802.05365.pdf>

Perera, P., & Madhushan Lankathilake. (2023). Preparing to Revolutionize Education with the Multi-Model GenAI Tool Google Gemini? A Journey towards Effective Policy Making. *Journal of Advances in Education and Philosophy*, *7*(08), 246–253. <https://doi.org/10.36348/jaep.2023.v07i08.001>

Qin, L., Liu, T., Che, W., Kang, B., Zhao, S., & Liu, T. (2020a). A Co-Interactive Transformer for Joint Slot Filling and Intent Detection. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2010.03880>

Qin, L., Xu, X., Che, W., & Liu, T. (2020b). *AGIF: An Adaptive Graph-Interactive Framework for Joint Multiple Intent Detection and Slot Filling*. Retrieved March 14, 2024, from <https://arxiv.org/pdf/2004.10087v4.pdf>

Qin, L., Wei, F., Xie, T., Xu, X., Che, W., & Liu, T. (2021). GL-GIN: Fast and Accurate Non-Autoregressive Model for Joint Multiple Intent Detection and Slot Filling. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2106.01925>

Rafiepour, M., & Sartakhti, J. S. (2023). CTRAN: CNN-Transformer-based Network for Natural Language Understanding. *Engineering Applications of Artificial Intelligence*, *126*, 107013. <https://doi.org/10.1016/j.engappai.2023.107013>

Schmidt, R. M. (2019). Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. ArXiv (Cornell University). <https://doi.org/10.48550/arXiv.1912.05911>

Staudemeyer, Ralf C, & Morris, E. R. (2019). Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1909.09586>

Tu, N. A., Uyen, H. T. T., Phuong, T. M., & Bach, N. X. (2023, August 28). *Joint Multiple Intent Detection and Slot Filling with Supervised Contrastive Learning and Self-Distillation*. ArXiv.org. <https://doi.org/10.48550/arXiv.2308.14654>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. <https://arxiv.org/pdf/1706.03762.pdf>

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. Artificial Intelligence Review, 53(8). [https://doi.org/10.1007/s10462- 020-09838-1](https://doi.org/10.1007/s10462-%20020-09838-1)

Wallach, H. (2004). *Conditional Random Fields: An Introduction \* 1 Labeling Sequential Data*. <https://nemenmanlab.org/~ilya/images/d/d4/Wallach-04.pdf>

Wang, J., Wei, K., Radfar, M., Zhang, W., & Chung, C. (2020). Encoding Syntactic Knowledge in Transformer Encoder for Intent Detection and Slot Filling. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2012.11689>

‌Wu, D., Ding, L., Lu, F., & Xie, J. (2020). SlotRefine: A Fast Non-Autoregressive Model for Joint Intent Detection and Slot Filling. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2010.02693>

Xiao, J., & Zhou, Z. (2020, June 1). Research Progress of RNN Language Model. IEEE Xplore. <https://doi.org/10.1109/ICAICA50127.2020.9182390>

‌