REVIEW

A joint learning classification for intent detection and slot filling from classical to deep learning: a review

Yusuf Idris Muhammad[1] · Naomie Salim[1] · Anazida Zainal[1] · Sinarwati Mohammad Suhaili[2]

Received: 4 June 2024 / Accepted: 4 May 2025

Abstract
In a dialogue system, the natural language understanding component plays a critical role in enabling effective communication. The two core tasks within this component are intent detection and slot filling. Intent detection identifies the user's goal, while slot filling extracts relevant information to fulfill that goal. Traditionally, these tasks were approached separately or in a pipeline-like manner. However, recent studies have emphasized the benefits of solving them jointly due to their natural interconnections. This study explores the evolution of joint learning models for intent detection and slot filling from 2008 to 2024, covering both classical and deep learning approaches. It discusses the limitations of classical models, which led to the rise of deep learning techniques, and introduces a new taxonomy for joint learning classifying joint learning architectures. Key benchmark datasets, evaluation metrics, and the challenges faced by joint models are also analyzed. Finally, the review identifies open research questions and proposes directions for future exploration in this field.

Keywords Intent detection · Slot filling · Dialogue system · Joint learning

1 Introduction

Dialogue systems have become essential tools for human–computer interaction, enabling users to communicate with machines using natural language [1]. Over the years, these systems have advanced significantly due to improvements in natural language processing (NLP), artificial intelligence, and user interface design [2]. In such systems, user input is first processed through an automatic speech recognition module before being interpreted by the natural language understanding (NLU) component [3].

The NLU component plays a critical role in understanding the semantics of user utterances [4]. It focuses on two key tasks: intent detection and slot filling [5]. Intent detection is a classification problem that identifies the user's goal by predicting the intent class while capturing semantic, syntactic, and contextual relationships between the words in the utterance [6, 7]. Slot filling, on the other hand, is a sequence labeling problem that extracts relevant entities required to fulfill the identified intent [8]. For example, in the query *'will it be warmer now in covenant life?*', the intent would be 'get-weather,' with slots such as 'condition_temperature,' 'time range,' and 'city' providing specific details. Semantic frames, such as the one shown in Table 1, represent this structured information, while the inside-outside-beginning (IOB) tagging format helps extract the slots efficiently.

Traditionally, intent detection and slot filling have been addressed as separate tasks, with models developed independently before being integrated into a complete system [9]. While this modular approach offers clarity, it introduces several limitations. For instance, treating the tasks independently overlooks the inherent interaction

https://doi.org/10.1007/s00521-025-11329-9
Published online: 05 June 2025

between intents and slots, which can otherwise enhance system performance when modeled jointly [10]. As demonstrated in Table 1, identifying slots like temperature and location can guide the system to detect the intent more accurately. Joint learning models capture these interdependencies, resulting in improved performance for both tasks [11]. Moreover, traditional pipeline systems often suffer from error propagation, where errors in one task affect the other, and fine-tuning them independently can be time-consuming [8, 12, 13]. Joint learning models address these challenges by sharing representations across tasks, improving both efficiency and performance [14].

The significance of joint learning is further underscored by its real-world applications. For instance, in conversational agents, voice-controlled systems, information retrieval tools, and domain-specific assistants in healthcare, finance and other sectors, joint learning of intent detection and slot filling, can efficiently handle complex queries, offering personalized responses with higher accuracy [13, 15, 16].

Over the years, significant advancements have been made in joint learning for intent detection and slot filling across several dimensions. Early models used classical machine learning techniques such as conditional random fields (CRFs) [17], support vector machines (SVMs) [18], maximum entropy models (MEMs) [19] and hidden Markov models (HMMs) [20]. These techniques have been succeeded by deep learning models like convolutional neural networks (CNNs) [21], recurrent neural networks (RNNs) [22], and Transformer architectures [23], which capture both local and global text dependencies [24]. Additionally, early models used lexical features for word representation, but the development of non-contextual embeddings like word2vec, Glove, and contextual embeddings such as ELMo (Embeddings from Language Models), and BERT (Bidirectional Encoder Representations from Transformers), has significantly enhanced semantic understanding by incorporating more information [25].

The learning strategies for joint learning models have also evolved, from implicit approaches that model interactions indirectly to explicit approaches that directly capture dependencies between intent detection and slot filling. More recently, fused approaches combine elements from both strategies to further improve performance [26–28]. Benchmark datasets such as ATIS (Air Travel Information Systems) and SNIPS have driven this progress by providing platforms for evaluation, while newer datasets like DSTC (Dialogue State Tracking Challenge) and Facebook NLU reflect the growing complexity of modern task-oriented dialogue systems.

This review provides an exploration of the literature, methodologies, and advancements in joint learning models for intent detection and slot filling from classical to deep learning models. Although several review studies have been published on natural language understanding tasks, such as [29] which describes the general overview of spoken language understanding in the era of neural networks, [1], outlines the fundamental framework of existing dialogue systems and provides a brief overview of the recent developments in several distinct subtasks of dialogue systems, such as natural language generation, dialogue management and natural language understanding, [30] which surveys the datasets for task-oriented dialogue systems, [31] surveyed the text representation used in NLP. A systematic review of the recent advances in deep learning-based dialogue system was reported in [32], while [33] provided a review of joint learning models for intent detection and slot filling, their survey focuses on neural network advancements within spoken language understanding.

This paper extends the review to include both classical and deep learning models, providing a historical perspective from 2008 to 2024. Additionally, this paper introduces a novel taxonomy for classifying joint learning

Table 1 An example of a semantic frame with query, slots, intent, and domain

Query	Will	it	be	Warmer	now	in	Covenant	Life
Slots	O	O	O	B-condition_temperature	B-timeRange	O	B-city	I-city
Intent	GetWeather							
Domain	Weather							

models, focusing on how these architectures evolve and interconnect over time to address intent detection and slot filling tasks. Furthermore, this review incorporates an analysis of benchmark datasets, their associated challenges and evaluation metrics. Finally, it highlights emerging challenges in the field and proposes new directions for future research.

The remainder of this paper is organized as follows: Sect. 2 discusses the technological advancements in joint learning models. Section 3 reviews classical models for joint learning classification, highlighting their performance and the limitations. Section 4 focuses on the deep learning models for joint learning. Section 5 introduces a new taxonomy for joint learning architecture. Section 6 reviews the datasets used for the joint learning classification of intent detection and slot filling. Section 7 discusses the challenges of joint learning classification datasets. Section 8 discusses evaluation metrics, while Sect. 9 explores the application areas of joint learning models for intent detection and slot filling. Section 10 identifies open challenges and future directions. Finally, Sect. 11 concludes the paper.

2 Overview of the joint learning technological advances

The development of joint learning models for intent detection and slot filling has evolved substantially, shifting from classical machine learning techniques to deep learning architectures. This section provides an overview of the technological advances in joint learning models, highlighting the transition from classical approaches to modern deep learning frameworks. Table 2 summarizes the key features, technologies, learning approaches, and datasets used in various studies.

2.1 Classical approaches (2008–2012)

Between 2008 and 2012, the foundation for joint learning models was built using classical machine learning techniques. Models such as CRFs [17], SVMs [18], MEMs [19] and HMMs [20] were extensively employed for joint learning intent detection and slot filing. These models relied heavily on manually engineered linguistic features, such as n-grams, prefixes, suffixes, part-of-speech (POS) tags, and semantic trees. The interaction between the two tasks was largely implicit, meaning the tasks interacted only at the encoder level. The ATIS dataset emerged as the benchmark for evaluating these models, providing a standardized dataset for spoken language understanding. In 2012, a dialog dataset derived from personal assistant systems was introduced, expanding the evaluation of these models to more practical scenarios.

2.2 Emergence of deep learning approaches (2013–2015)

The period from 2013 to 2015 marked a turning point with the adoption of deep learning techniques. These methods significantly improved performance by reducing reliance on manual feature engineering and allowing the automatic extraction of semantic and syntactic features. Models during this phase incorporated word embeddings, lexical items, and named entities, which enriched input representations. In 2013, CNNs were applied for joint learning to extract features directly from the text, coupled with CRFs for slot filling tasks [34]. By 2014, recursive neural networks (RecNNs), were employed to model the hierarchical dependencies within utterances, leveraging n-grams and named entities features [35]. By 2015, joint learning models had fully embraced neural network architectures, integrating CNNs for feature extraction with RNNs to capture both local and global dependencies. The ATIS dataset remained the primary benchmark for this period, while the Microsoft Cortana dataset was introduced to evaluate the robustness of models in real-world virtual assistant applications.

Table 2 Overview of the joint learning models literature

Year	Number of papers	Features	Technologies	Learning approach	Dataset	References
2008	1	n-grams, words	CRF	implicit	ATIS	[17]
2009	1	Semantic tree	SVM	implicit	ATIS	[18]
2010	1	n-grams, words, suffixes, prefixes, POS tags	CRF, MEMs	implicit	ATIS	[19]
2012	1	n-grams	CRF, HMM	implicit	Dialog dataset from personal assistant system	[20]
2013	1	Words	CNN, CRF	implicit	ATIS	[34]
2014	1	n-grams, lexical, Name entity	CNN, RecNN	implicit	ATIS	[35]
2015	1	Word embeddings	CNN, RNN	implicit	ATIS, Microsoft Cortana	[36]
2016	5	Glove, char embeddings, word embeddings, word2vec, 1-hot encoding, lexicon, NE	CNN, (Bi)RNN, (Bi)GRU, (Bi)LSTM, attention	implicit	ATIS, SNIPS, Microsoft Cortana	[37–41]
2017	4	Char embeddings, word embeddings	CNN, sparse attention, hiBiLSTM	implicit	ATIS, Microsoft Cortana	[42–45]
2018	14	Glove, word2vec, POS tags, lexical, rules	(Bi)LSTM, (Bi)GRU, CNN, Attention, self-attention	implicit, explicit-unidirectional	ATIS, TRAINS, SNIPS, ALEXA domains	[46–59]
2019	20	Word2vec, Glove, BERT embeddings, ELMo, gazetteer, word embeddings, char embeddings, dependency parse	(Bi)LSTM, (Bi)GRU, CNN, CRF, BERT, MLP, RCNN, Attention, self-attention, multi-head attention	implicit, explicit-unidirectional, explicit-bidirectional	ATIS, SNIPS, FRAMES, DSTC4, CAIS, NLPCC2018, AMIE-incabin	[14, 60–79]
2020	14	Word2vec, Glove, BERT embeddings, ELMo, word embeddings, char embeddings, dependency parse, POS, Regular Expressions, external knowledge	(Bi)LSTM, (Bi)GRU, CNN, CRF, BERT, MLP, Attention, self-attention, graph attention	implicit, explicit-unidirectional, explicit-bidirectional, fused	ATIS, SNIPS, TOP, DSTC4, CSID, TRAINS, Facebook NLU, KVRET	[27, 28, 79–89]
2021	13	NE, POS, ELMo, BERT, FastText, Glove, word2vec	(Bi)LSTM, (Bi)GRU, CNN, CRF, BERT, RoBERTa, Stacked LSTM/GRU, MLP, Attention, self-attention, graph attention	implicit, explicit-unidirectional, explicit-bidirectional, fused	ATIS, SNIPS, TRAINS, MIT Restaurant, ChatData, MIT movie	[6, 11, 27, 90–99]
2022	15	Word2vec, Glove, BERT embeddings	BERT, RoBERTa, (Bi)LSTM, attention, GCN, Graph attention	implicit, explicit-unidirectional, explicit-bidirectional, fused	ATIS, SNIPS, TOP, CAIS, PhoATIS, SMP-ECDT	[15, 100–113]
2023	14	BERT embedding, word2vec, Glove	BERT, RoBERTa, GCN, Graph attention, FFN	implicit, explicit-unidirectional, explicit-bidirectional, fused	ATIS, SNIPS, AGIS, CAIS, SMP-ECDT	[9, 10, 12, 114–124]
2024	3	BERT	BERT, GCN, (Bi)LSTM/GRU, hierarchical attention	implicit, explicit-unidirectional, Fused	ATIS, SNIPS	[26, 125–127]

2.3 Emergence of recurrent architectures and attention mechanisms (2016–2018)

Between 2016 and 2018, recurrent neural network encoder-decoder architectures gained prominence, particularly for sequence-to-sequence tasks. The introduction of attention mechanisms during this period allowed models to selectively focus on specific parts of the input sequence, improving both intent detection and slot filling. Models explored both unidirectional and bidirectional architectures, such as long short-term memory (LSTM) and gated recurrent unit (GRU), alongside embeddings like Glove and ELMo.

In this period, new datasets such as SNIPs and TRAINS provided a wider variety of benchmarks for training and evaluation. Furthermore, sparse and self-attention mechanisms emerged as powerful tools to manage language complexity, capturing global dependencies within input sequences. Additionally, explicit unidirectional learning approaches were introduced, enabling the model to capture directional dependencies between intent detection and slot filling.

2.4 Emergence of advanced architectures (2019–2024)

The advent of transformer-based models marked a significant advancement in joint learning. The BERT revolutionized the field by providing deep bidirectional context understanding, which improved both accuracy and generalization. Additionally, CRFs were reintroduced alongside deep learning architectures to further enhance sequence labeling performance, demonstrating synergy between classical and deep learning methods.

During this period, datasets expanded to reflect the growing complexity of task-oriented systems, including FRAMES, DSTC4, CAIS (Chinese AI Speaker), TOP (Task Oriented Parsing), Facebook NLU, etc. Furthermore, the use of graph attention networks and few-shot learning techniques were introduced to address challenges of data scarcity and model generalization. New variants of BERT, such as RoBERTa, alongside fused learning approach were introduced to enhance learning efficiency and model performance.

3 Classical models for joint learning classification

Classical models play an essential role in the early stages of developing joint learning models for intent detection and slot filling. These models, which are often based on statistical and machine learning principles, offer effective solutions for analyzing and processing natural language data. They employed basic nonparametric techniques to capture the crucial features embedded within the spoken language. These extracted features then acted as inputs for subsequent classification or regression models. Examples of such models are CRFs [128], MEMs [129], SVMs [130], HMMs [20] and logistic regression, which formed the foundation of early approaches to joint learning classification.

3.1 Conditional random fields

Conditional random fields are probabilistic graphical models widely used for sequence labeling tasks, such as part-of-speech tagging [131] and named entity recognition [132]. CRFs model the conditional probability of a label sequence y given an observation sequence x, $P(y|x)$. This allows them to capture dependencies between labels and optimize predictions locally, leading to more accurate results [131]. CRFs also offer flexibility by incorporating a variety of contextual features and using a discriminative training approach, often outperforming generative models such as HMMs [133]. The conditional probability in CRFs is defined as:

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{t} w.\emptyset(y_t, y_{t-1}, x)\right) \tag{1}$$

where w represents the learned weights, and $\emptyset(y_t, y_{t-1}, x)$ is the feature function that captures dependencies between adjacent labels y_t and y_{t-1}, along with observation sequence x, and $Z(x)$ is the partition function, which sums over all possible label sequences y' for a given observation sequence x.

$$\sum_{y'} \exp\left(\sum_{t} w.\emptyset(y_t\prime, y_{t-1}\prime, x)\right) \tag{2}$$

The partition function normalizes the probability distribution, ensuring that the sum of all possible label sequences equals 1. However, calculating *Z(x)* can be computationally expensive, as it requires summing over all possible label sequences, which becomes especially challenging for long sequences or large datasets [134]. This is a key limitation of CRFs, as computing the partition function grows exponentially with the sequence length.

The earliest study on joint-learning classification utilized tri-angular CRFs to capture the interdependence between sequence labeling and sequence classification [17]. The proposed model represents two tasks in a single graphical model consisting of three layers: token features, sequence labeling, and sequence classification label layers. The key innovation lies in the utilization of parameter sharing, which is a technique that maximizes the likelihood of labeled training data. A hybrid approach was adopted in [19] to integrate MEMs and CRFs. The MEMs functioned as generative models for obtaining the target domain, whereas CRF served as a discriminative model for extracting the slots.

CRF is used in joint models for slot filling tasks because of its effectiveness in sequence-labeling problems [135–137]. Despite the shift toward deep learning, there has been a recent resurgence of interest in CRFs within the context of deep learning models. Several contemporary joint learning approaches [6, 78, 86, 121, 136, 138] have highlighted the renewed significance and value of CRFs in addressing specific challenges and enhancing the performance of deep learning models for joint intent detection and slot filling tasks. This resurgence highlights the benefits of integrating classical and deep learning approaches to further advance the field.

3.2 Maximum entropy models

Maximum entropy models are discriminative models based on the principle of maximum entropy, which suggests that the most unbiased probability distribution, given a set of constraints, is the one with largest entropy. MEMs are used in NLP due to their flexibility in feature representation and discriminative training [139]. Consider a sequence of observations $O_1, O_2, \ldots O_n$ with the objective of assigning tags $S_1, S_2, \ldots, S_n$ to these observations in a manner that maximizes the conditional probability $P(S_1, S_2, \ldots, S_n|O_1, O_2, \ldots, O_n)$. In the MEM, this probability is decomposed into Markov transition probabilities. This implies that the likelihood of transitioning to a specific label at a given position depends solely on the observation at that position, and the label assigned to the previous position.

$$P(S_1, S_2, \ldots, S_n|O_1, O_2, \ldots, O_n) \prod_{t=1}^{n} P(S_t|S_{t-1}, O_t) \tag{3}$$

Each transition probability originates from a common distribution $P(s|s', O)$. For every potential label value of the preceding label s', the likelihood of a specific label s is structured like that of a maximum entropy classifier:

$$P(s|s', O) = P_{s'}(s|o) \frac{1}{Z(o, s')} \exp\left(\sum_a \lambda_a f_a(o, s)\right) \tag{4}$$

In this context, $f_a(o, s)$ represents feature functions that can be either real-valued or categorical, and $Z(o, s')$ denotes a normalization term that guarantees that the distribution sums to one. This distribution aligns with the maximum entropy probability distribution that adheres to the constraint that the empirical feature expectation matches the model expectation.

$$E_e[f_a(o, s)] = E_p[f_a(o, s)] \text{ for all } a \tag{5}$$

One of the key strengths of MEMs is their ability to incorporate a wide variety of features, including overlapping and non-independent ones. This flexibility allows for the integration of rich contextual information, which is crucial for accurately labeling sequences in NLP [129]. Moreover, MEMs do not assume independence between features, allowing them to capture more complex dependencies in the data. However, despite these advantages, the computational complexity of MEMs can be a challenge. Training these models, which involves iterative methods like gradient descent, can be time-consuming, especially for large datasets [140]. Another limitation of MEMs is the label bias problem [141]. In MEMs the probabilities of outgoing transitions from a given state are normalized locally. As a result, these transitions are only compared against one another, without accounting for competing transitions from other states. This local competition can limit the model's ability to properly handle long-range dependencies across states, leading to performance bottlenecks in sequence prediction tasks [141].

Despite these challenges, MEM was used for frame classification in [19]. It served as classifier where the label was predicted on the input utterance. MEM was applied in a two-pass strategy: first for domain classification, and then CRFs were used for slot filling. The two approaches, which employed MEM, resulted in significantly better accuracy with a limited amount of data.

3.3 Hidden Markov models

Hidden Markov models are statistical models designed to represent systems with hidden states [142]. In an HMM, the system is modeled as a Markov process, where each state is hidden, but the observable data provide insight into these states. This structure makes HMMs useful for tasks involving sequences, where understanding the order and the relationship between elements is essential [129].

It consists of a set of hidden states $S = \{s_1, s_2, \ldots, s_n\}$ and a set of observable symbols $O = \{o_1, o_2, \ldots, o_m\}$. The transitions between hidden states are governed by the transition probabilities $A = \{a_{ij}\}$, where $a_{ij} = P(s_j|s_i)$, and the emission probabilities $B = \{b_j(o_t)\}$, where $b_j(O_t) = P(o_t|s_j)$, which represents the likelihood of observing o_t given the hidden state s_j. This allows HMMs to model the joint probability of the hidden state sequence and the observed data, making it possible to infer the most likely sequence of hidden states using algorithms such as forward–backward for inference and Viterbi for decoding.

$$P(O, S) = P(s_1) \prod_{t=2}^{T} P(s_t|s_{t-1}) P(o_t|s_t) \tag{6}$$

HMMs are particularly useful in tasks where maintaining the context between sequential elements is important [143]. Their probabilistic framework allows them to incorporate uncertainty and variability, which is common in real-world data [144]. However, despite these advantages, HMMs also have several limitations [141]: the assumption that state transitions depend solely on the current state can be restrictive, especially for tasks that

require capturing long-range dependencies in sequences. Additionally, HMMs rely on limited feature sets, focusing on current observation and the previous state, which may reduce their accuracy when richer information is required.

To overcome some of these challenges, researchers have developed variations and enhancements to the basic HMM framework. For example, in [20], multilayer HMMs have been proposed to incorporate hidden intentions and prior knowledge, improving their performance in joint intent detection and slot filling.

3.4 Support vector machines

Support vector machines are supervised learning algorithms used for classification and regression tasks [145]. In NLP, SVMs have been successfully applied to tasks like text classification [146], sentiment analysis [147], and named entity recognition [148]. They are particularly useful in dealing with high-dimensional data by leveraging feature vectors such as the Bag-of-words, TF-IDF, and semantic tree. This makes them effective for scenarios where the vocabulary size is large, as commonly encountered in NLP tasks.

SVMs work by solving an optimization problem to maximize the margin between two classes, which can be formulated as:

$$\min \frac{1}{2}||w||^2 \text{ subject to } y_i(w.x_i + b) \geq 1 \forall_i \tag{7}$$

where w is the weight vector, x_i represents the input feature vectors, y_i denotes the labels, and b is the bias term. The goal is to find the optimal hyperplane $w.\ x + b = 0$ that maximizes the margin between the data points of different classes. By doing so, SVMs can classify data points with improved generalization performance.

SVMs have been used for intent detection and slot filling [18]. In this context, the input features are obtained using a semantic tree, which is then passed to SVMs to obtain the intent and slots. Despite their success, traditional SVMs rely on manual feature engineering, which fails to capture the nuanced, context-dependent relationships between words. Additionally, SVMs do not perform well on large-scale dataset [149].

Table 3 provides a summary of the classical models used for joint learning classification of intent detection and slot filling, with their features, datasets and performance.

While these models played a role in the early development of joint learning classification for intent detection and slot filing, they also possess several inherent limitations that impact their performance, scalability and adaptability, especially compared to modern deep learning approaches. These limitations can be summarized as follows:

Feature Engineering Dependency [34, 150]*:* Classical models heavily rely on manually crafted features to perform tasks. Feature engineering requires significant domain expertise and effort to extract relevant linguistic patterns from raw data. This process can be time-consuming, and the quality of features directly impacts model performance.

Limited Ability to Capture Complex Contextual Relationships [141, 151]: Classical models like HMMs and MEMs assume independence between input features or between current and previous states in sequences, which can restrict their ability to capture long-range dependencies or context. This assumption oversimplifies the relationships within data, leading to reduced accuracy in tasks that require understanding complex interactions between words. While CRFs can capture local dependencies between labels, they are still limited in their ability to fully represent entire sequences.

Scalability Issues [152, 153]: Classical models struggle to handle large datasets due to their computational complexity. For example, calculating the partition function in CRFs or training MEMs with iterative methods like gradient descent can be computationally expensive, especially as the dataset size grows. SVMs, which

Table 3 Classical joint model's performance

References	Model	Features	Dataset	Performance		
				Intent detection		Slot filling
				Accuracy (%)	Error rate (%)	(F1-score) (%)
[17]	CRFs	n-grams	ATIS	93.07	–	94.42
[18]	SVMs	Semantic tree	ATIS	92.63	–	94.50
[19]	MEM + CRF	Lexical/word suffices	ATIS	–	0.11	61.0
[20]	HMMs	n-grams	Dialog dataset from personal assistant system	84.52	–	58.18

perform well with high-dimensional datasets, also face difficulties scaling large-scale data due to their reliance on optimization techniques that become inefficient with more data points.

Inflexibility in Handling Unseen Data [154, 155]: Classical models are generally less flexible in generalizing to unseen data or handling variations in language. They typically perform poorly when faced with out-of-vocabulary words, unseen contexts, or domains that differ from the training data. This lack of adaptability limits their usefulness in real-world applications where data distributions frequently shift.

Complex Training Process [34]: The training of classical models can be challenging due to the computational complexity and the optimization methods they rely on. For example, CRFs require summing over all possible label sequences to normalize the probability distribution, which becomes computationally expensive for longer sequences. Similarly, MEMs rely on iterative optimization, which can be slow, especially for large complex datasets.

Poor Handling of Data Scarcity [133]: While classical models, like SVMs are more robust to data scarcity than others, they generally do not perform well with limited labeled data. These models rely heavily on the quality and quantity of the training data, and their performance reduces when faced with sparse annotations.

These limitations have led to the development of alternative approaches, particularly deep learning approaches.

4 Deep learning models for joint learning classification

The success of joint learning for intent detection and slot filling lies in the choice of architectures and models that can effectively capture complex relationships between tasks. This section discusses various neural network-based approaches, including RecNNs, RNNs, CNNs, and transformer-based models, as well as their application in joint learning models. Additionally, the essential role of attention mechanisms in enhancing the capabilities of joint learning models is explored.

4.1 Recursive neural networks

Recursive neural networks are types of neural network designed to process structured data, such as trees or graphs, by applying same weights recursively. In natural language processing, RecNNs breaks down sentences into phrases and words, progressively combining these lower representations into higher level ones [156]. For example, if two child node representations h_1, h_2 are combined to form a parent node representation h_p, the process is expressed as:

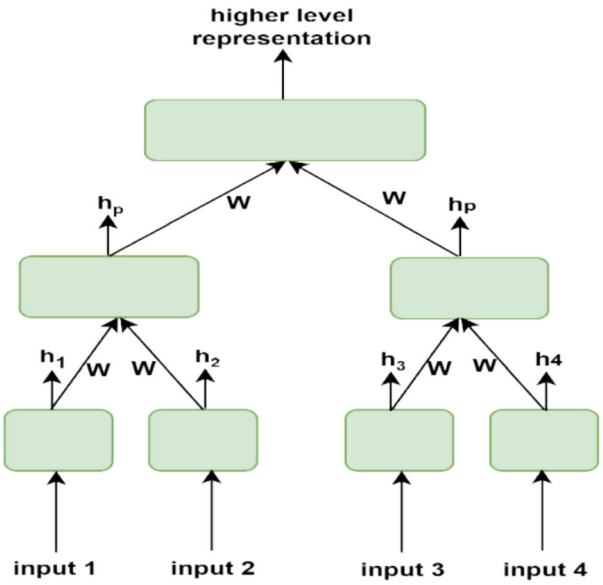


Fig. 1 RecNNs iteratively construct higher-level representations from lower-level representations

$$h_p = f(W.[h_1; h_2] + b) \tag{8}$$

where W represents the shared weights matrix, b is the bias term, and f is a nonlinear action function. The architecture consists of input layers representing basic units like words, hidden layers that combine child node representation into parent nodes, and an output layer that delivers a final representation for classification as shown in Fig. 1.

RecNN was proposed for joint intent detection and slot filling in [35], due to its ability to represent syntactic information in discrete and continuous space. To classify user intent, the dot product of the internal vectors from the leaf to the root is computed and passed to the softmax function to obtain the posterior over the intent labels. To perform the slot filling task, token-wise classification was applied to each word by using tree-derived features. This feature is computed by multiplying the output vector from the leaf to the root by a weight vector associated with the syntactic type of node. The resulting product was then summed up to generate a vector, which was subsequently passed to the maximum-entropy classifier to predict the IOB label for each word. After training the entire model, the Viterbi optimizer was applied to optimize the sentence-level tag sequence.

Although the model has achieved state-of-the-art performance, RecNNs face challenges in handling long-range dependencies and can be computationally expensive due to their recursive nature [157]. Moreover, their reliance on accurate syntactic parse trees means that errors in parsing can lead to suboptimal performance [158].

4.2 Recurrent neural networks

Recurrent neural networks are designed to process sequential data, making them ideal for tasks involving time series and natural language text [22]. What distinguishes RNNs from other architectures is their ability to maintain a memory of previous inputs through recurrent connections, allowing them to capture patterns and dependencies over time. This memory enables RNNs to perform well on tasks that require understanding the sequential nature of data such as language modeling [159], machine translation [160], sentiment analysis [161], and speech recognition [162].

This flexibility arises from their inherent structure, which enables them to operate on inputs of different lengths, using sequential processing. Moreover, RNNs possess a hidden state that evolves as the network sequentially processes each input. This hidden state serves as the context for capturing the relevant information

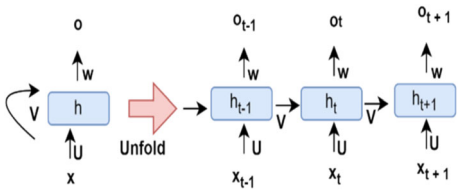


Fig. 2 Basic RNN [32]

from past inputs. Consequently, RNNs can leverage this contextual information to make predictions or to generate outputs at each step.

RNNs excel in intent detection and slot filling tasks because these tasks rely on understanding the context provided by a sequence of words [163]. In intent detection, the network's recurrent structure allows it to analyze a user's input in the context of prior words or phrases, making it easier to infer the user's intent [164]. Similarly, in slot filling, the sequential processing of RNNs helps in recognizing the relationships between words, thereby accurately identifying relevant slots in sentences [20, 135]. Furthermore, RNNs offer flexibility to incorporate additional linguistic features and contextual information into the model architecture [165], enhancing their performance in intent detection and slot filling tasks. RNNs have several variants:

4.2.1 Basic RNNs model

The fundamental RNN model, known as the Elman network [166], consist of three layers: input, hidden, and output. The hidden state h_t at any given time step t is computed as the function of the current input x_t and previous state h_{t-1}, making the process recurrent as shown in Fig. 2.

$$h_t = f(U * x_t + W * h_{t-1}) \tag{9}$$

where o_t, is current output, U and V are the weight matrices shared across time, and f is a nonlinear activation function, such as ReLU, tanh, or sigmoid.

The hidden state of the RNNs is represented as h_t in Eq. (9) serve as the memory of the network that gather information from the preceding time steps. This feature allows RNNs to effectively handle sequences, thereby addressing the limitations of feedforward neural networks.

While basic RNNs offer solutions for processing sequential data by maintaining the memory of past inputs, they also face challenges in effectively utilizing both past and future information, optimizing delay parameters, and integrating multiple network outputs to improve the prediction accuracy [24]. To address, these limitations, [167] proposed bidirectional RNNs (BiRNNs). This model processes the input in both forward and backward directions, allowing the network to consider past and future context in separate hidden layers, as shown in Fig. 3. The outputs of these two RNNs are merged at each time step to generate a richer representation of the sequence.

$$\overrightarrow{h_t} = f\left(V_f * \overrightarrow{h_{t-1}} + U_f * x_t + b_h\right) \tag{10}$$

$$\overleftarrow{h_t} = f\left(V_b * \overleftarrow{h_{t+1}} + U_b * x_t + b_h\right) \tag{11}$$

where $\overrightarrow{h_t}, V_f, U_f$ and $\overleftarrow{h_t}, V_b, U_b$. are the hidden states and weight matrices for forward and backward hidden states, respectively.

Both simple RNNs and BiRNNs have been observed to encounter a significant obstacle known as the vanishing gradient problem, which hinders their ability to effectively learn from long-range contextual information [168, 169]. To address this limitation, specialized RNN variants have emerged, including LSTMs and GRUs [170].

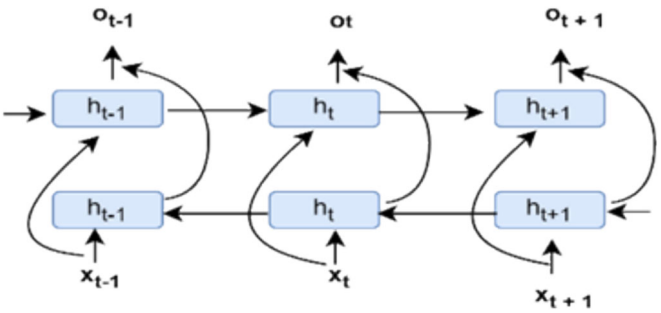


Fig. 3 BiRNN architecture

4.2.2 Long short-term memory

LSTM was proposed to overcome the vanishing gradient problem of a basic RNN [170]. This model consist of three gates: input, forget and output gate as shown in Fig. 4 [171]. The input gate (i_t), determines what new information to add to the long-term memory also known as cell state (c_t). The forget gate (f_t), back propagate errors through unlimited number of time steps by discarding parts of the cell states that are insignificant to the context and retaining the significant part, while the output gate (o_t), controls the output of the LSTM at each time step, which is based on the current cell state (c_t) and the hidden states (h_t) as per the equations below:

$$i_t = f(w_i * x_t + u_i * h_{t-1} + b_i) \tag{12}$$

$$f_t = f\left(w_f * x_t + u_f * h_{t-1} + b_f\right) \tag{13}$$

$$o_t = f(w_o * x_t + u_o * h_{t-1} + b_o) \tag{14}$$

$$c_t\prime = f(w_c * x_t + u_c * h_{t-1} + b_c) \tag{15}$$

$$c_t = f_t * c_{t-1} + i_t * c_t\prime \tag{16}$$

$$h_t = o_t * \sigma(c_t) \tag{17}$$

where f is the activation function, c_t, c_{t-1} are the long-term memory in the current and previous states, respectively, and h_t, h_{t-1} are the current and previous states of short-term memory, respectively. $w_i, w_f, w_o, w_c, u_i, u_f, u_o, u_c$ are the weight matrices, and b_i, b_f, b_o, b_c are the biases.

Despite the success of LSTM in addressing many challenges in recurrent neural networks, they also encountered certain limitations when processing sequential data. One key issue is the exponential decay of memory, where the forget gate may discard useful information too quickly when processing longer sequences [173]. Another challenge is saturation of memory cells [174]. LSTMs accumulate information in their internal states, and if this grows without control, it can saturate the output function, making learning inefficient.

4.2.3 Gated recurrent neural network

To provide a computationally efficient alternative to LSTM, [175] proposed a GRUs that utilizes two gates instead of three, and a single memory cell instead of two as shown in Fig. 5. GRUs regulate the flow of information over time by incorporating an update gate z_t to determine the portion of the new input to add to the previous hidden state and a reset gate r_t, thereby determining how much previous hidden state information is discarded as per the following equations.

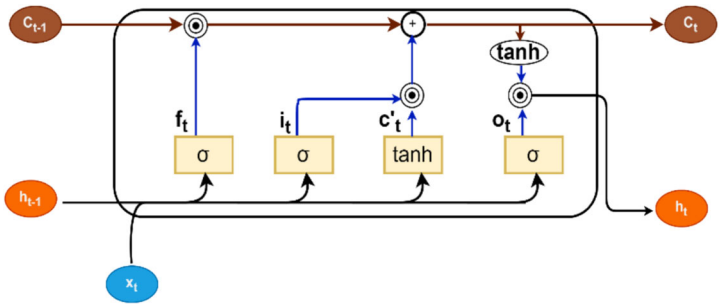


Fig. 4 LSTM architecture [172]

$$z_t = \sigma(w_z * x_t + u_z * h_{t-1}) \tag{18}$$

$$r_t = \sigma(w_r * x_t + u_r * h_{t-1}) \tag{19}$$

$$h_t\prime = \tanh(w * x_t + r_t \odot u * h_{t-1}) \tag{20}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t\prime \tag{21}$$

where z_t is the update gate, r_t is the reset gate, and h_t is the hidden state.

RNNs models are used in joint learning classification studies because of their suitability for capturing temporal dependencies. Researchers have applied RNNs using various methodologies [28, 37, 39–41, 54, 77, 82, 92, 98, 176]. In some studies, RNNs served as word-level classifiers, where intermediate hidden states are utilized for slot filling, and a weighted sum of these hidden states forms a context representation for intent detection. In other studies, RNNs act as sentence-level classifiers, where the last hidden state is used for intent detection.

In [41], the authors introduced a hierarchical LSTM model with two layers. The bottom-layer functions as a sentence-level classifier, using the last hidden state for intent detection, while the upper layer performed word-level classification for slot filling. In [39], an LSTM was used to detect both intent and slots at each time step, based on word-level information. Additionally, the overall intent was derived from the last hidden state of the LSTM, capturing sentence-level intent. In [40], the authors added a special token at the end of each utterance before passing it through BiLSTM. This approach generated a latent semantic representation of the entire input for intent detection, while intermediate hidden states were used for slot labeling. In [37], a context vector, computed from the weighted sum of the RNN hidden states, was utilized to obtain both intent and slots.

In [38], the authors proposed a joint model using a GRU and max pooling layer. The GRU captured the representation at each time step for slot labeling, while the max pooling layer was used for intent classification. The representations generated by the GRU were shared between the slot filling and intent detection tasks. Word

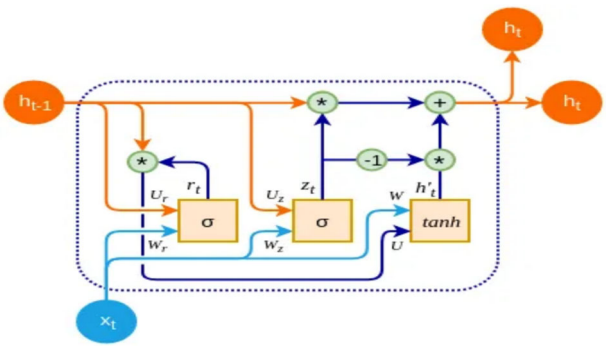


Fig. 5 GRUs architecture [22]

embedding with a context window served as input to the model. Additionally, the model incorporated the association of named entities with the embeddings. To predict slots, a bidirectional GRU was employed to learn representations of the input sequence in both forward and backward directions, with the final state computed by concatenating both forward and backward states. Furthermore, a max pooling layer was introduced to obtain representations of the entire sequence and convert them into a fixed-length vector to capture the entire text. Finally, the softmax function is applied to these representations to obtain slot filling and intent classifications.

Although RNNs are primarily used to capture sequential information, they can also be trained to consider other types of information such as local semantic information [177]. This is typically achieved by incorporating additional layers or architectures such as an attention mechanism [176, 178] or transformer layers [27, 178]. Additionally, the nature of the training data and tasks can influence how the RNN emphasizes certain types of information.

Despite achieving state-of-the-art performance, joint classification models based on recurrent architectures several challenges. The vanishing gradient problem in RNNs makes it difficult for the model to retain and use long-range dependencies effectively, which is critical for tasks like slot filling, where words earlier in a sentence influence the labeling of later words [179]. The issue of handling long-range dependencies also affects intent detection, as understanding the intent often requires considering the entire sentence context. Even though LSTMs and GRUs were designed to address some of these challenges, they also face the problem of slow training speed, which is attributed to internal recurrence [22]. Additionally, the fixed-length vector representation of the RNN architectures, constrains the joint model's ability to encode information in lengthy inputs [180].

4.3 Convolutional neural networks

Convolutional neural networks [181], originally developed for image processing [181], have proven to be highly effective in extracting higher-level features for NLP tasks [182]. Their ability to detect local patterns, such as key word sequences and n-grams, makes them well-suited for applications like sentiment analysis [183], text classification [184], question answering [185] and machine translation [186].

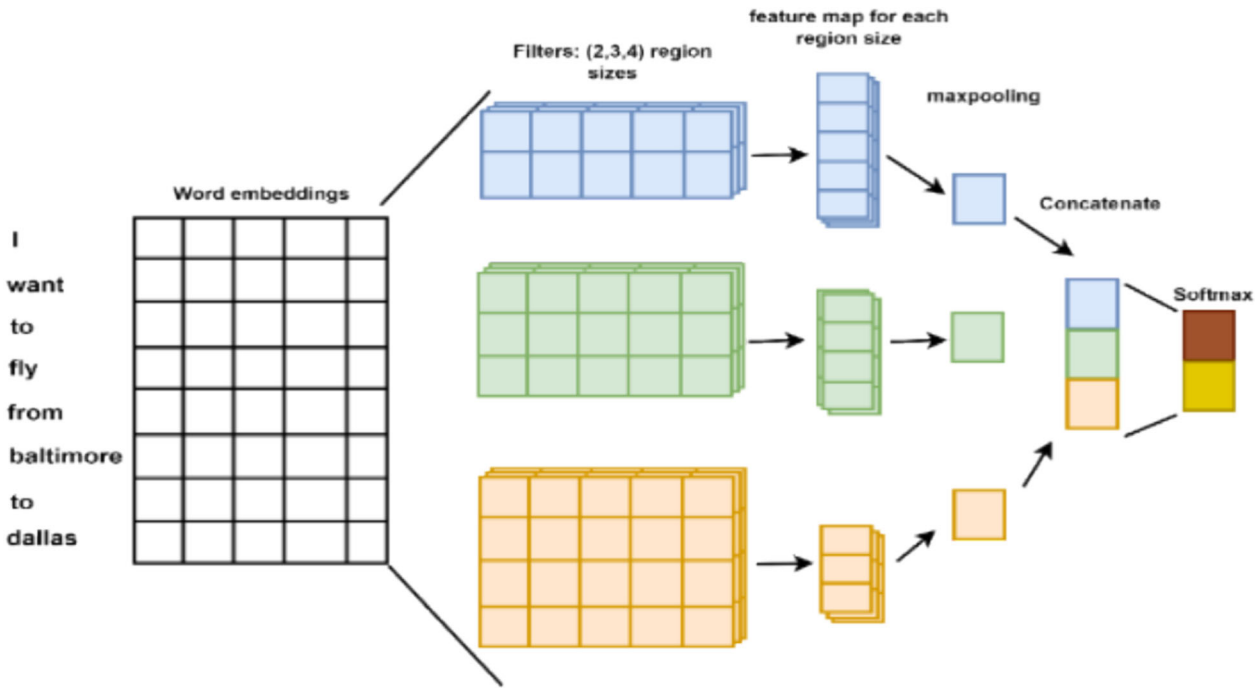


Fig. 6 CNN architecture for sentence classification [32]

The structure of a CNN is composed of several layers, each serving a distinct purpose in the feature extraction and classification process. The core components of a CNN include convolutional layers, pooling layers, and fully connected layers as shown in Fig. 6. Convolutional layers apply a set of filters to the input text, which is typically represented as a matrix of word embeddings. These filters slide over the text to capture local patterns, such as phrases or n-grams, that are important for understanding the context and meaning of the text. Following the convolutional layers are the pooling layers, which reduce the dimensionality of the feature maps generated by the convolutional layers. The pooling layers help to summarize the presence of important features across different parts of the of the text. This process makes the model more robust to variations in the text and helps to focus on the most relevant features for the task at hand [6, 187]. At the end of the network are the fully connected layers, which take the high-level features extracted by the convolutional and pooling layers and use them to make the final prediction.

In joint learning studies, CNNs have been applied using two approaches: the sentence-based approach [64, 188, 189] and the window-based approach [28, 34, 47]. The sentence-based approach transforms entire sentences into representations suitable for intent detection. Conversely, the window approach generates word-based predictions necessary for tasks such as slot filling [190], POS tagging [191], and NER [192].

To extract meaningful patterns, a convolution operation is performed on the embedding matrix $W \in \mathcal{R}^{n \times d}$, where d is the dimension of the word embeddings, and n is the number of words in a sentence. The convolution uses a filter $k \in \mathcal{R}^{hd}$, where h defines the number of words it spans. As the filter slides across the matrix, it captures localized features from overlapping word windows. For example, applying the filter to the word window $w_{i:i+h-1}$, generates a feature c_i, as illustrated below:

$$c_i = f\left(w_{i:i+h-1}.k^T + b\right) \tag{22}$$

where $b \epsilon \mathcal{R}$ represents the bias term, and f denotes a nonlinear activation function. The filter k is applied across all feasible windows with the same weights to generate the feature map, as given in Eq. (23).

$$c = [c_1, c_2, \ldots, c_{n-h+1}] \tag{23}$$

The max pooling operation is then applied to the output feature maps to extract the most salient features, as described below:

$$p = \max(c) \tag{24}$$

CNN was first employed in a joint model for intent detection and slot filling in [34], where the authors utilized a window-based approach to extract features directly from the embedded word sequences, which are then shared across both tasks. For intent detection, the aggregated feature vectors were max-pooled and passed through a softmax layer to predict the intent. For the slot filling, the features were passed to CRF, which predicted the slot labels considering the current word and the neighboring labels. In another study proposed in [64], the authors used a hierarchical CNN to extract features from the input, which were then passed to a multilayer perceptron for predicting both intent and slot labels.

A hybrid LSTM-CNN model was employed in [188]. This architecture used an LSTM to capture the contextual representation of input and a CNN to model sentence-level features in a parallel. The outputs of both networks merged to perform intent detection and slot filling. Similarly, in [47], a CNN-BiLSTM model was proposed, where CNN extracted the local features while the BiLSTM captured the contextual representations for intent detection and slot filling.

In [28], the authors used a parallel approach using a CNN combined with BiLSTM-CRF model. CNN was employed to extract higher-level n-grams and bigrams features, while BiLSTM-CRF captured the contextual information from the embedded text. These features were concatenated and passed through a fusion block for

explicit parameter sharing. The output from the fusion layer was then fed into dense layers to predict intent and slot label.

Despite achieving success in various NLP tasks, CNNs face notable limitations when applied to joint learning classification for intent detection and slot filling. One of the primary challenges is their inability to capture detailed sequential dependencies within sentences [193], which is crucial for slot filling. CNNs relies on identifying overall semantic patterns which are well suited for intent detection, but less effective for slot filling task. Consequently, CNNs often struggle to balance the demands of both tasks in joint learning models [194]. To address these limitations hybrid models like CNN-BiLSTM or CNN-LSTM have been proposed. These models use CNNs for local feature extraction and LSTMs or BiLSTMs to capture long-range dependencies, providing a more balanced approach to handling both intent detection and slot filling.

4.4 Attention mechanism

The attention mechanism has emerged as a critical component in revolutionizing the process of processing and understanding natural language. It enables models to focus selectively on various parts of the input sequence when generating the output, mimicking the human cognitive ability to prioritize information [195]. This dynamic allocation of attention enhances the contextual understanding and performance of NLP systems across a spectrum of tasks. Additionally, attention mechanisms, allow models to weigh the importance of various parts of an input sequence with respect to the current step in output generation [23]. The attention mechanisms used in joint model studies include:

4.4.1 The weighted sum of hidden states of RNNs

In this type of attention, an alignment function is used to compute the score for each element in the input sequence, as shown in Eq. 26. These scores indicate the extent to which each element is aligned during the decoding step. Various scoring methods such as dot product, additive, and multiplicative scoring are adopted to quantify this alignment. Computed alignment scores are transformed into attention weights using the softmax function, as shown in Eq. 27. These weights represent the relative importance of each input element during decoding. Higher attention weights indicated greater importance, whereas lower weights indicated lower relevance. Finally, the context vector is computed as the weighted sum of the hidden states based on their weights, as shown in Eq. 28. Figure 7 shows the work of the attention model.

Given an input $x_i = (x_1, x_2, \ldots, x_t)$ the encoder produces a hidden state h_i

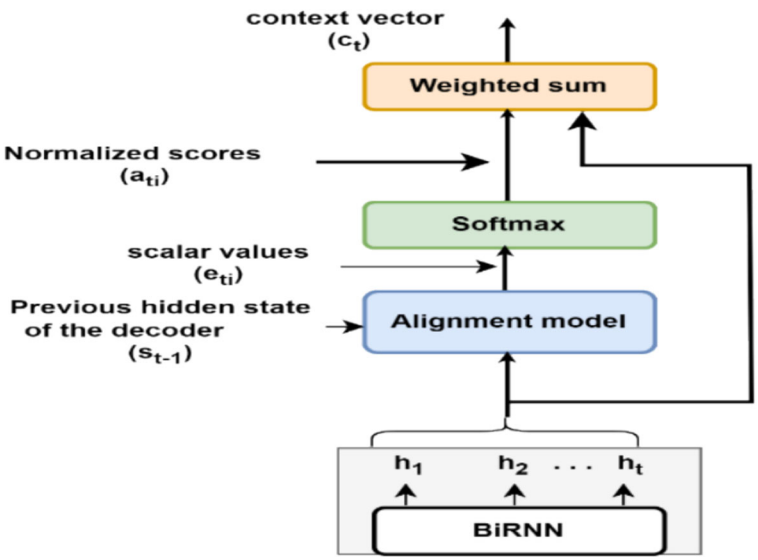


Fig. 7 Attention based on a weighted sum of hidden states of RNNs

$$h_i = (h_1, h_2, \ldots, h_t) \tag{25}$$

$$e_{ti} = f(s_{t-1}, h_i) \tag{26}$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{t} \exp(e_{ti})} \tag{27}$$

$$c_t = \sum_i a_{ti} h_i \tag{28}$$

This type of attention was first used in [37], employing an encoder-decoder architecture with weighted sum attention for a joint learning task. The proposed model utilizes a bidirectional RNN with LSTM recurrent units as an encoder, along with two unidirectional RNNs as decoders for intent and slot labeling. The encoder processed the input in both the forward and backward directions to generate hidden states. These hidden states were then concatenated and passed to the decoders to generate intent and slot labels. The intent decoder received input from the last state of the concatenated hidden states of the shared encoder, as well as a weighted sum of the encoder's hidden states, allowing it to leverage both global and local information for intent detection. Meanwhile, the slot filling decoder operated by calculating the slot label at each time step based on the last hidden state of the encoder, the weighted sum of the encoder's hidden states, and the previous decoder state. This combination of inputs enabled the slot filling model to capture contextual dependencies effectively.

This attention mechanism has been demonstrated to be computationally efficient compared to non-attention joint models, reducing the computational burden while improving performance in joint learning tasks [196]. Despite these advantages, the weighted sum of hidden states, commonly used in traditional encoder-decoder models, has limitations [197]. It relies heavily on the encoder's hidden states and processes sequentially, which can lead to inefficiencies with longer sequences. Additionally, it may struggle to capture long-range dependencies and global context, as it focuses on local alignment between encoder and decoder states. To address these issues, more advanced attention techniques have been introduced, allowing models to capture both local and global dependencies more effectively.

4.4.2 Self-attention

Self-attention, also known as intra-attention [23], is a mechanism that connects various positions within a single sequence to generate a representation of the sequence. It operates by mapping the input matrix X into the query matrix Q, the value matrix V and the key matrix K, then computes the attention weights through a scaled dot-product operation, as depicted in Fig. 8:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{29}$$

where $Q = W_Q X$, $K = W_K X$ and $V = W_V X$, d_k is the dimension of the key vectors and W_Q, W_K, and W_V are the weight matrix corresponding to Q, K, and V, respectively. This enables the model to assign varying levels of importance to different words in a sequence, facilitating the capture of long-range dependencies and context. This capability is particularly valuable for tasks such as intent detection and slot filling, where relationships between words can may span across the sequence [9].

In terms of complexity, self-attention outperforms recurrent layers when the sequence length (n) is shorter than the representation dimensionality (d). The complexity is $O(n^2 d)$ compared to $O(nd^2)$ for recurrent layers [198]. This makes self-attention more efficient for handling sequences common in dialogue systems, where processing

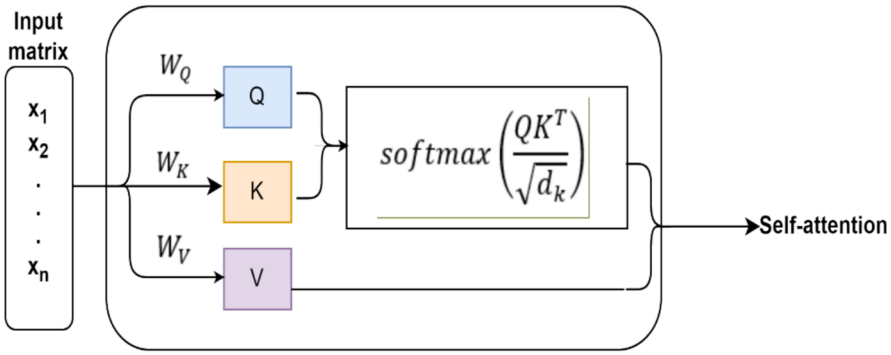


Fig. 8 Self-attention mechanism

speed is critical. Furthermore, self-attention can process all tokens simultaneously, reducing the need for sequential token processing required by RNN-based models.

Several studies have employed self-attention in joint model learning for intent detection and slot filling [58, 63, 72, 98, 176, 199]. However, despite its strengths, self-attention faces limitations when applied to extremely long sequences [200]. The mechanism can become constrained by the neighborhood around the corresponding output position, particularly when memory and computation resources are restricted [23]. This can be mitigated by techniques such as hierarchical or sparse attention.

4.4.3 Hierarchical attention

Hierarchical attention is a structured approach that applies the attention mechanism at multiple levels within a hierarchical data structure. It was first introduced in the context of document classification [201–203] and has since been employed in other tasks [204]. This model captures information at multiple granularities, typically at both the word and sentence levels, making it effective for processing long texts where context at different levels is critical.

In joint learning models, hierarchical attention proves useful in multi-intent dialogues [127]. For slot filling tasks, a word-level attention is employed to identify key words within a sentence, as different slots correspond to different words. The attention weight, denoted as α_i, is computed by projecting the hidden state h_i using a learned context vector u_w:

$$\alpha_i = \frac{\exp\left(u_w^T \tanh(W_w h_i + b_w)\right)}{\sum_j \exp(u_w^T \tanh\left(W_w h_j + b_w\right)} \tag{30}$$

where W_w, b_w are the learnable parameters. For the intent detection, the model aggregates the word representations to form a sentence-level representation, enabling it to capture the overall meaning of the sentence. The sentence-level attention weight, β_j, is computed as:

$$\beta_j = \frac{\exp\left(u_s^T \tanh\left(W_s s_j + b_s\right)\right)}{\sum_k \exp(u_s^T \tanh(W_s s_k + b_s)} \tag{31}$$

where s_j represents the sentence's representation.

In joint learning for intent detection and slot filling, hierarchical attention captures the importance of both individual words and sentences. This dual-level attention provides a more nuanced understanding of complex and lengthy inputs, significantly enhancing the model's ability to process multi-intent dialogues. However, one major challenge of hierarchical attention is its reliance on accurately defining the boundaries between different intents within an utterance [205]. This limitation may cause errors in distinguishing between the correct intents, leading to reduced performance in capturing the full meaning of an utterance.

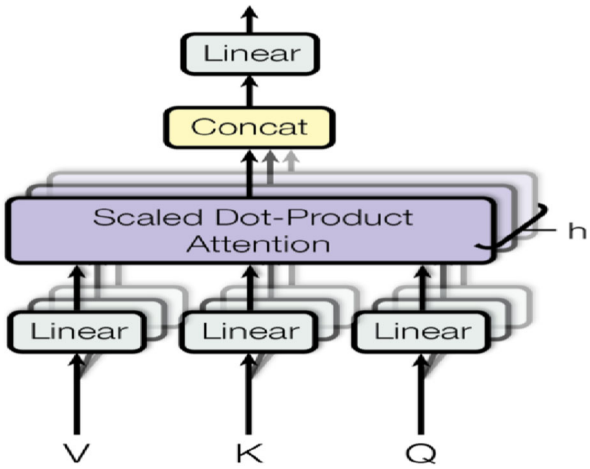


Fig. 9 Multi-head Attention mechanism [23]

4.4.4 Sparse attention

Sparse attention is designed to address the computational inefficiencies of traditional self-attention mechanisms, which scale quadratically with the sequence length n, leading to high memory and computational costs for long sequences [206]. In joint learning classification for intent detection and slot filling, sparse attention is applied to mitigate these inefficiencies by assigning higher attention weights to semantically important words, thus improving the model's performance by reducing noise from irrelevant words [44].

The sparse attention mechanism computes a sentence representation $\hat{h}$ by weighting each hidden state h_t based on its importance, as determined by an attention score α_i:

$$\hat{h} = \sum_{t=1}^{N} \alpha_t h_t \tag{32}$$

The attention score α_t is calculated using a context vector α, which is learned during training, and a nonlinear function ψ such as sigmoid or ReLU:

$$\alpha_t = \psi(h_t \cdot \alpha) \tag{33}$$

The function ψ induces sparsity by assigning higher weights to semantically important words while reducing the weights of less irrelevant words. This selective weighting not only reduces computational complexity but also enhances the model's ability to focus on relevant parts of the input. However, the efficacy of sparse attention diminishes in tasks involving shorter sequences, where most or all of the tokens tend carry meaningful information [200, 207].

4.4.5 Multi-head attention

The multi-head attention mechanism, as proposed in [23], introduces multiple attention heads rather than utilizing a single attention mechanism as seen in Sect. 4.4.2. Each head computes attention scores independently, and the results are then concatenated and passed through a final linear transformation W^o to produce a richer representation of the input data as shown in Fig. 9. The multi-head attention mechanism can be defined as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, \ldots, \text{Head}_n)W^o \tag{34}$$

$$\text{Head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{35}$$

where Q, K, and V are the query, key, and value matrices of the input sequence, and W_i^Q, W_i^K and W_i^V are learned projection matrices for i-th attention head.

This architecture enables the model to process multiple perspectives of the input sequence concurrently, leading to a more comprehensive understanding of dependencies and interactions across the sequence [208].

Multi-head attention has been combined with other deep learning models to enhance the performance of joint learning models. For example, the authors in [109] integrated multi-head attention with iterated dilated CNNs, allowing the model to capture richer semantic information from the multiple parts of the input text. In another study proposed in [68], multi-head attention was combined with CRF and prior knowledge. The multi-head attention helped capture long-range dependencies in the input text, while the CRF layer ensured consistency in the output labels. Additionally, a prior mask was used to incorporate knowledge into the model, further enhancing its performance.

Despite its advantages, multi-head attention presents certain limitations. The primary challenge lies in increased computational and memory demands, as each attention head requires its own set of parameters and projections [23]. This parallel computation increases resource consumption, particularly for long sequences or when many attention heads are used. Moreover, while multi-head attention is designed to capture diverse aspects of the input, it can lead to redundancy if the attention heads focus on irrelevant features, potentially hindering model performance [209].

4.5 Transformer architecture

Transformers have revolutionized the field of NLP, offering a paradigm shift in the processing and comprehension of textual data. Originally introduced in the groundbreaking paper "Attention is All You Need" [23], Transformers have become the backbone of many state-of-the-art NLU models [210]. Unlike recurrent neural network architectures, which process data sequentially, the Transformer processes the entire sequence in parallel, making it faster and more effective in capturing complex patterns [211].

The Transformer architecture consists of two main blocks: encoders and decoders as shown in Fig. 10. The encoder processes the input data in parallel, making it more efficient than traditional models that process data step by step. The input is first converted into a vector using an embedding technique. Because Transformers lack an inherent sense of sequence order, position encoding is added to the input vectors to provide information about the order of elements in the sequence.

The encoder is made up of multiple identical layers, each containing two components: multi-head attention mechanism and feed-forward neural network. The multi-head attention mechanism allows the model to focus on different parts of the input simultaneously, capturing multiple relationships within the data. After applying the attention mechanism, the encoder passes the data through a Feed-Forward Network, which helps capture more

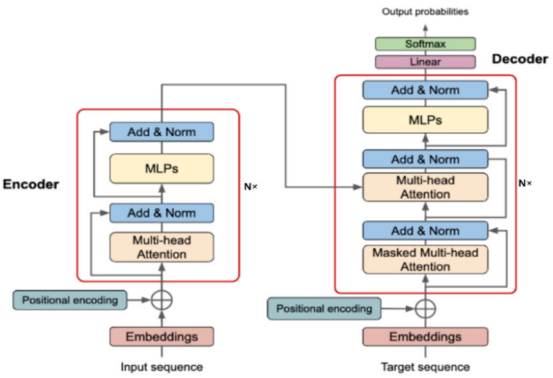


Fig. 10 Transformer architecture [23]

complex patterns and relationships. Add & Norm operations are then used to stabilize the learning process and ensure that the model converges effectively.

The decoder is responsible for generating output sequence. It includes an additional feature known as masked multi-head attention mechanism, which prevents the decoder from seeing future elements in the output sequence during training. Like the encoder, the decoder also uses a standard multi-head attention mechanism to focus on relevant parts of the encoder's output. The decoder's final output is then passed through a softmax layer, which converts the output into a probability distribution, allowing the model to generate predictions by selecting the most likely outcome.

The Transformer architecture has enabled the development of large-scale pre-trained language models such as Bidirectional Encoder Representations from Transformers (BERT) [212], Generative Pre-trained Transformers (GPTs) [213], and eXtreme Language Understanding Network (XLNet) [214].

BERT is one of the most frequently used Transformer architectures for joint learning classification of intent detection and slot filling [26, 60, 61, 63, 99, 100, 124, 126, 178, 215, 216]. In these models, the input sequence is tokenized into sub-word units, and each token is encoded into a dense vector representation. The Transformer captures the contextual information by considering the entire input sequence. For intent detection, a classification layer is added on top of the Transformer output, mapping contextualized representations to the intent labels. Additional layers are used to label slots, which can be either linear [216] or recurrent [86]. Although transformers excel at modeling long-range contextual dependencies, they may struggle to capture fine-grained and local patterns in data [33], which are often crucial in slot filling task [217].

4.6 Hybrid model

Hybrid-based approaches combine different models to perform joint learning classification for intent detection and slot filling. Several studies [44, 47, 54, 57, 64, 72, 218–220] combined RNNs and CNNs to capture the local and global dependencies within the input, leading to improvements in the performance of both intent classification and slot filling. In these models, the input is first encoded using an RNN, which captures the sequential nature of the input. The final hidden state is then passed to a fully connected layer for intent detection. A CNN is then applied to the task of slot filling by convolving the sequence of word embeddings with multiple filters of varying sizes. The resulting feature maps are combined and passed through a fully connected layer to predict slot labels.

In other studies [52, 66, 68, 76, 77, 86, 138, 221, 222], RNNs and CRFs are employed. The input is encoded using an RNN, and the hidden state is fed to a fully connected layer for intent detection. For slot filling task, the output of the RNN is fed into the CRF, which models the dependencies between adjacent slot labels. Other works [12, 78, 79, 82, 86] used Transformer models with LSTM or CRF to leverage the Transformer's contextualization power and the sequential modeling capacity of CRFs and LSTM.

Table 4 summarizes the performance of the deep learning models recorded on different datasets. The results suggest that joint learning approaches perform particularly well on the ATIS and SNIPS datasets, achieving up to 99% in both intent detection and slot filling tasks in almost all techniques.

The features used in the deep learning models are generated automatically. However, to capture more information, pre-trained embeddings are often used to initialize the embedding layer and serve as input features. These features include word embeddings such as word2vec [223], Glove [224], and ELMo [225], as well as character embeddings, such as fastText, which are beneficial for handling out-of-vocabulary (OOV) words, and BERT embeddings, which provide contextual embedding.

In some studies, multiple embeddings were combined to harness their strengths. For instance, [51] combined Glove, word2vec, and POS tags as embeddings to capture a broader range of word representations. To address the OOV issue, [64, 75, 226] used a combination of character and word embeddings. ln [74] one-hot encoding was used along with the key verb features of the input data. A recent study [227] proposed the use of contrastive learning to obtain embeddings from convolutional features and BERT encoding.

Table 4 Deep learning-based models

Technique	Features	Performance			References
		Intent Detection Accuracy (%)	Slot filling F1-score (%)	Dataset	
RNN-based	Word embeddings, character embeddings (FastText) Graph Embedding, bag-of-words, one-hot encoding, BERT embeddings	75–99.6	39–99.98	ATIS	[4, 11, 28, 37, 38, 40–42, 48, 54–56, 58, 71, 74, 92, 98, 99, 178, 228–232]
		73–98.8	41.1–98.74	SNIPS	
		83.11–84.88	96.89–98.78	TRAINS	
		94–95	93–95	Cortana	
		59	33.2	TOP	
		96.05	87.12	CQUD	
		99.40–99.48	98.01–98.84	DSTC2	
		53.35	45.51	DSTC5	
CNN-based	Convolutional n-grams	94.09	95.42	ATIS	[34]
Hybrid-based	Word embeddings, character embeddings, Graph Embedding, bag-of-word, Convolutional n-grams, BERT embeddings, syntactical features (POS)	94.58–99.29	74.56–99.54	ATIS	[14, 27, 44, 46, 47, 52, 57, 62, 66, 72, 75, 77, 82, 86, 138, 199, 215, 218–222, 233–235]
		95.80–99.7	91.78–98.44	SNIPS	
		94.56	86.16	CAIS	
Transformer-based	BERT embeddings	98.6–99.76	96.1–98.75	ATIS	[60, 61, 63, 99, 100, 178, 215, 216]
		98.96–99.25	94.8–98.78	SNIPS	

https://doi.org/10.1007/s00521-025-11329-9

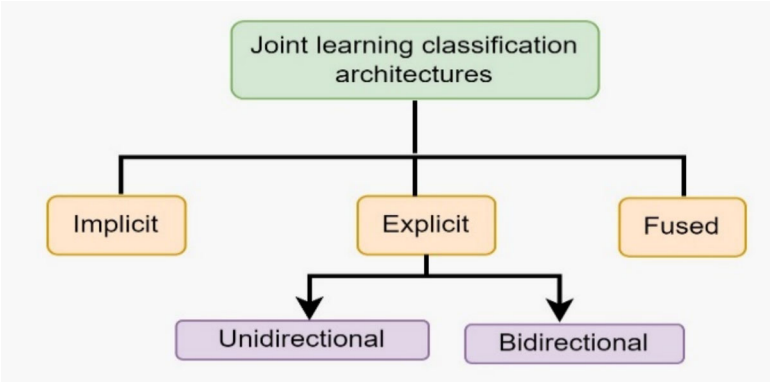


Fig. 11 A Taxonomy of the joint learning classification architectures for intent detection and slot filling

5 Joint learning classification architectures' taxonomy

The relationship between intent detection and slot filling has led to the development of various joint learning architectures designed to exploit this interdependence. These architectures can be classified into three broad categories as shown in Fig. 11: implicit, explicit, and fused, based on how they capture and leverage the interaction between the two tasks. Each of these categories introduces distinct design choices regarding key components, such as encoders, shared representations, interaction modules, and decoders.

At the heart of many joint learning models lies the shared encoder, a component responsible for processing the input data and generating a feature representation that serves both tasks. This allows the model to extract patterns relevant to both intent detection and slot filling, eliminating the need to design separate models for each task. Common examples of shared encoders include Bi(RNNs), Bi(LSTMs), Bi(GRUs) and transformers, all of which are widely used for their ability to process sequential data effectively. The intermediate output produced by the shared encoder is known as the shared representation, which contains the features extracted from the input sentence. This representation serves as a common knowledge base that feeds both the intent detection and slot filling modules, enabling both tasks to access relevant information from the same source. To further enhance the interdependence between the two tasks, some architectures incorporate interaction module. This component allows the predictions made by one task to influence the other. Finally, the outputs of the model are generated through decoders, which are task-specific layers that produce final predictions. This section introduces taxonomy, providing an overview of the architectural approaches used in joint learning models.

5.1 Implicit shared feature joint learning architecture

In implicit joint learning architectures, a shared encoder is employed to capture common features without explicitly modeling the interaction between intent detection and slot filling. The primary focus of this approach is to extract shared representations between the two tasks through a joint encoding mechanism. These models rely on shared features across tasks to drive predictions, without explicitly modeling task dependencies. A typical structure of an implicit shared feature joint model is shown in Fig. 12.

Several studies have explored this approach. For instance, Liu and Lane [37] proposed an RNN as a shared encoder to produce vector representations of the input data, followed by two decoders for intent detection and slot filling. Similarly, Chen, Hakanni-Tür [219] used a CNN as the shared encoder, with an RNN to label the slots while leveraging the last hidden state for intent detection. Firdaus, Bhatnagar [51] used stacked GRU and LSTM layers as shared encoders, passing the shared representations to multilayer perceptron whose outputs were ensembled for predictions. Daha and Hewavitharana [222] proposed BiLSTM as a shared encoder, using the last hidden state for intent detection and CRF for slot filling. In another study [233] proposed BiLSTM for contextualized representations, passing them to a Graphical Convolutional Network with multi-head attention to obtain the shared representations for intent detection and slot filling tasks.

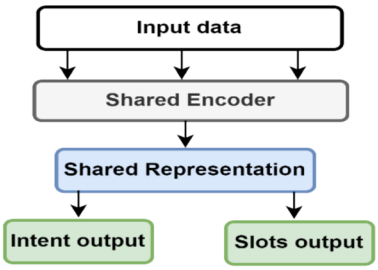
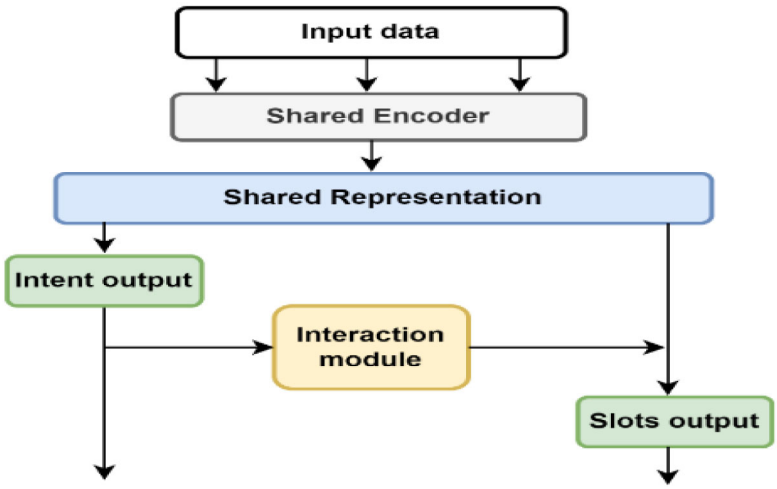


Fig. 12 Implicitly shared feature joint model architecture

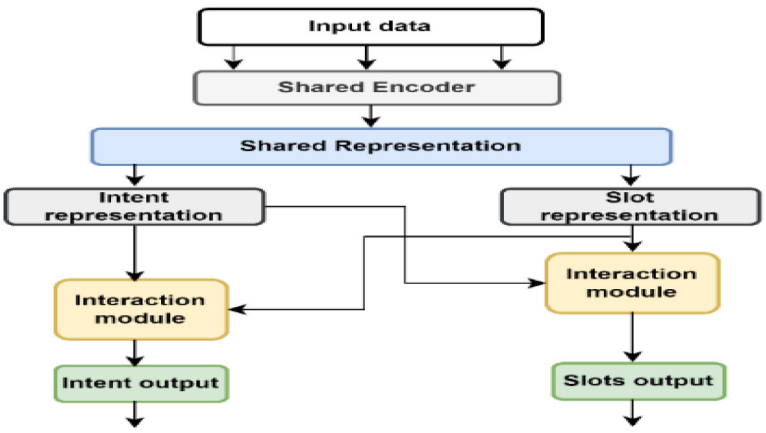
Although the implicitly shared feature joint models provide a direct approach for knowledge sharing, they often suffer from lower interpretability and performance due to the absence of explicit interaction between tasks [78].

5.2 Explicit shared feature joint learning architecture

To address the limitations of implicit joint learning architectures, explicit shared feature models have been developed [38]. These models capture the interaction between the intent and slots through architectural mechanisms such as attention mechanisms, gating structures, or cross-task dependencies. Explicit modeling has been



a) Unidirectional explicit shared feature joint learning architecture



b) Bidirectional explicit shared feature architecture

Fig. 13 Explicit shared feature joint model architecture

shown to improve accuracy, interpretability, and robustness compared to implicit joint model architectures, offering a promising direction for advancing dialogue understanding systems. Explicit joint model architectures can be categorized into two types: unidirectional and bidirectional, as shown in Fig. 13(a), (b), respectively.

In unidirectional explicit shared feature joint learning architectures, the intent information is passed to the slot filling module without feedback. For instance, Goo, Gao [232] and [58] proposed models that used intent vectors to improve the slot filling task via a gating mechanism. Despite the efficacy of the models with unidirectional explicit architecture, their reliance on unidirectional flow introduces the risk of error propagation, where incorrect intent detection can affect slot filling performance [71].

On the other hand, bidirectional explicit shared feature joint learning architecture allows mutual information exchange between the tasks. For example, [54] proposed bi-modal-based RNN structures to leverage the cross-impact between slots and intent using bi-directionality. Qin, Liu [178] proposed a co-interactive transformer model that establishes bidirectional connections between the tasks, and models like [98], used a self-attention to generate bidirectional representations that facilitate task interaction. Bidirectional architectures have demonstrated enhanced accuracy and interpretability by allowing tasks to refine each other's predictions.

5.3 Fused joint learning architecture

Fused joint learning architectures use separate encoders for intent detection and slot filling, integrating their outputs through a fusion layer before final prediction. Unlike explicit architectures, which facilitate interaction at the encoder level, fused architectures merge task representations at a later stage, as shown in Fig. 14. This approach allows for task-specific feature extraction while still benefiting from shared knowledge during prediction stage.

Bhasin, Natarajan [28] proposed a fused joint architecture utilizing distinct BiGRU encoders for intent detection and slot filling tasks, where feature combination for prediction is achieved through bilinear pooling. Another study [27] introduced a dual encoder that independently encodes input sequences using BERT and BiLSTM, and the outputs are merged via an interaction block, before being processed by intent and slot decoders. A recent study conducted by[26] employed BERT and BiLSTM as encoders for intent and slots, respectively, and combined their outputs to model explicit task interactions. To reduce error propagation, this model uses word-level information to fuse the two types of information. While the models based on fused architecture offer flexibility by handling each task independently, the lack of interaction between the tasks at the encoder level may limit their ability to fully capture complex dependencies between intent and slot tasks.

Table 5 presents the performance of various joint learning architectures for intent detection and slot filling on the ATIS dataset. Implicit joint learning architectures show a wide range of performance, with a few achieving competitive results. The best performing implicit models reach near perfect accuracy and F1-score. This

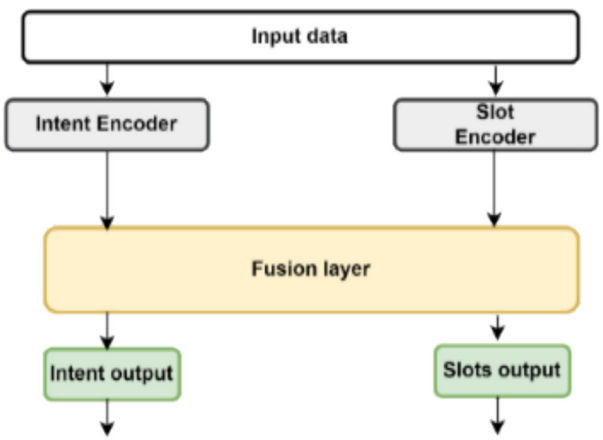


Fig. 14 Fused joint model architecture

Table 5 Performance comparison based on learning architecture on ATIS dataset

Joint learning architecture	Performance				References
	Intent detection Accuracy (%)	Slot filling F1-score (%)	Average Accuracy (%)	Average F1-score (%)	
Implicit	75–99.60	39.7–99.98	87.3	69.84	[4, 11, 12, 17, 35, 37–39, 44, 47, 51, 56, 64, 66, 72, 77, 78, 99, 138, 219, 221, 222, 228, 233, 236, 237]
Explicit-unidirectional	93.6–99.76	94.22–99.98	96.68	97.10	[34, 52, 57, 58, 63, 79, 100, 231, 232]
Explicit-bidirectional	88.8–98.99	95.2–99.6	93.90	97.40	[27, 46, 54, 61, 62, 75, 92, 98, 176, 178, 199, 220, 238, 239]
Fused Joint	97.54–98.43	95.8–99.54	97.99	97.67	[26–28]

highlights that, when tuned well, implicit models can handle the tasks effectively despite their architectural simplicity. The large variability in intent accuracy and F1-scores indicates significant inconsistency. The lowest values, especially the F1-score of 39.7%, suggest that these models often fail to capture task-specific dependencies leading to low performance. The average scores affirm that implicit joint learning architecture struggles to balance both tasks due to the lack of explicit task interaction.

Models with an explicit unidirectional architecture show improvement over implicit models. With an average intent accuracy of 96.68% and an F1-score of 97.10%, these architectures effectively capture task dependencies. Notably, the unidirectional flow of information from the intent detection module to the slot filling module results in a higher average performance for the slot filling task compared to intent detection, as the latter informs the former without feedback, enhancing the slot labeling task.

The models with bidirectional explicit architecture show improved F1-scores and stable intent accuracy. Although these models perform well, the slightly lower average accuracy compared to unidirectional models could be attributed to the increased complexity of maintaining mutual task interaction.

Fused joint model architecture consistently achieves high performance in both intent accuracy and F1-score. This strong performance can largely be attributed to their ability to balance task-specific feature extraction with shared learning.

Table 6 compares the performance of joint learning architectures on the SNIPS dataset. Similar to the ATIS dataset, implicit architecture shows a wide range of performance variability. This highlights the inherent

Table 6 Performance comparison based on learning architecture on SNIPS dataset

Joint learning architecture	Performance				References
	Intent detection Accuracy (%)	Slot filling F1-score (%)	Average Accuracy (%)	Average F1-score (%)	
Implicit	73.94–99.7	38.6–96.30	97.12	89.10	[4, 12, 64, 72, 77, 78, 82, 99, 138, 221, 222, 228, 233, 237]
Explicit-unidirectional	88.03–98.96	88.8–98.78	98.01	93.25	[63, 71, 79, 231, 232]
Explicit-bidirectional	93.2–99.3	91.8–98.74	98.62	95.56	[27, 46, 61, 62, 75, 92, 98, 176, 178, 199, 220, 238, 239]
Fused Joint	98.14–99.0	96.7–98.49	98.58	97.40	[26–28]

challenges of joint learning models without explicit interaction. However, the average performance suggests that implicit architectures perform better on dataset with a large number of training examples such as SNIPS.

For explicit unidirectional models, the results align with those observed on the ATIS dataset, where the unidirectional flow of information from intent detection to slot filling tends to enhance the performance of the slot filling task. Similarly, explicit bidirectional joint learning models achieve higher accuracy on the SNIPS dataset compared to ATIS. The fused joint learning models also demonstrate improved average performance on the SNIPS dataset relative to ATIS. These findings suggest that the balanced distribution of intents in the SNIPS dataset provides more conditions for these architectures, allowing them to fully leverage task interdependence.

Tables 5 and 6 show that the fused joint learning architecture consistently delivers the highest performance across both datasets, likely due to its more integrated handling of the two tasks. In both ATIS and SNIPS datasets, explicit bidirectional joint learning models also perform strongly, indicating that architectures enabling task interdependence outperform those with implicit or unidirectional connections. However, despite the high overall accuracy on both datasets, the ATIS dataset consistently shows higher F1-scores compared to SNIPS, likely due to the ATIS dataset having more slots and better task alignment [240]. On the other hand, the SNIPS dataset demonstrates consistently higher intent accuracy, which could be attributed to its more balanced nature and larger number of training examples.

6 Datasets for joint learning classification for intent detection and slot filling

Robust datasets and appropriate evaluation metrics are essential for evaluating of joint learning models in intent detection and slot filling tasks [30]. Many errors in these models stem from issues such as annotation mistakes or inherent ambiguities in the datasets [241]. Several datasets have been developed to facilitate research on joint learning, including widely recognized datasets like ATIS, SNIPS, and Microsoft Cortana, which offer varied user intents and slot types, enabling realistic model training and evaluation. The selection of the dataset depends on the application domain and the complexity of language understanding required. Table 7 summarizes the key characteristics of the most used datasets in joint learning classification.

6.1 ATIS (Airline travel information system) dataset

The ATIS dataset was initially introduced under the Defense Advanced Research Projects Agency's initiative to automate travel booking systems and has since become a key benchmark for intent detection and slot filling tasks [33]. The dataset primarily contains queries related to air travel, such as flight details, fare information, and associated services. The original ATIS-0 release included 740 training examples featuring transcriptions, SQL queries, and tokens adhering to Standard Normal Orthographic Representation rules, with an average utterance length of 11.3 tokens per utterance [244]. Over time, various variants were developed, with ATIS-3 being the most used in research.

However, a key challenge of the ATIS dataset is its class imbalance, with approximately 75% of the samples pertain to the to the "atis-flight" intent [179]. This disproportionate distribution causes models to become biased toward predicting frequent intents while underperforming on less common ones.

6.2 SNIPS dataset

The SNIPS dataset, developed by SNIPS [245], a company known for its expertise in natural language understanding technology for voice assistants, includes a diverse set of queries across seven domain such music, weather inquiries, home automation, etc. A key advantage of SNIPS dataset is its balanced distribution of different intents, which promotes evaluation models. This characteristic is essential for training robust models that generalize well to a wide range of real-world spoken queries without bias. As a result, SNIPS has become a

Table 7 Characteristics of datasets for joint intent detection and slot filling

Dataset	Intent	Slots	Train dataset	Test dataset	Domain	public	References
ATIS	21	128	4478	500	Air Travel	Yes	[4, 11, 12, 17, 35, 37–39, 44, 47, 51, 56, 64, 66, 72, 77, 78, 99, 138, 219, 221, 222, 228, 233, 236, 237] [34, 52, 57, 58, 63, 79, 100, 231, 232] [27, 46, 54, 61, 62, 75, 92, 98, 176, 178, 199, 220, 238, 239]
SNIPS	7	72	13,084	700	Personal voice assistant	Yes	[4, 12, 64, 72, 77, 78, 82, 99, 138, 221, 222, 228, 233, 237] [63, 71, 79, 231, 232] [27, 46, 61, 62, 75, 92, 98, 176, 178, 199, 220, 238, 239]
FRAMES	24	136	200,006	6598	Hotel booking	Yes	[11, 12, 64]
DSTC2	13	4	4790	4485	Restaurant search	Yes	[41, 48, 188]
DSTC5	84	533	27,528	3447	Dialogue with a social robot	Yes	[48]
Microsoft Cortana	112	71	67,818	6905	Personal Assistant	No	[35, 40, 42, 218, 219]
Facebook	12	11	30,521	8621	Multi-lingual	Yes	[63, 86, 242, 243]
CQUD	43	20	3286	-	Multi-domain and multilingual	No	[38]
CMRS	5	5	2901	967	Meeting Room Booking	No	[48]
TRAINS	12	32	5355	1336	Problem-solving dialogue	No	[11, 12, 51, 64]
Rokid Music	4	10	50,638	10,807	Music		[52]
TOP	25	36	31,279	4462	Multi-domain	Yes	[99, 237]
CAIS	11	75	7995	1012	AI Speakers	Yes	[75]

https://doi.org/10.1007/s00521-025-11329-9

valuable resource for development and evaluation of personal assistant systems, offering a realistic environment to test model performance.

6.3 FRAMES dataset

The Frames dataset consists of multi-turn dialogues that simulate interactions between users and agents, focusing on hotel and vacation package bookings [11, 246]. These conversations are designed to reflect real-world scenarios where users explore multiple options, change preferences, and revisit previous topics during decision-making. Each dialogue involves a series of exchanges between a user and a wizard (agent), with both parties able to introduce new frames. Every dialogue turn in the dataset is annotated with dialogue acts and slot-value pairs. These annotations make the Frames dataset well-suited for training models that need to capture evolving contexts across interactions, ensuring that systems can accurately manage multiple user goals. The dataset is particularly useful for developing context-dependent systems, such as advanced virtual assistants or booking agents, that must handle complex interactions. These systems benefit from the ability to switch between multiple goals and maintain continuity across long conversations. As a result, the Frames dataset supports research into memory-enhanced dialogue systems capable of managing nuanced, multi-step interactions.

6.4 DSTC dataset

The Dialogue State Tracking Challenge (DSTC) datasets are used to benchmark dialogue systems, with different versions catering to various conversational tasks. DSTC2 focuses on restaurant search dialogue, assessing model's abilities to track user states throughout interactions. This feature is essential for task-oriented dialogue systems that need to manage user goals and preferences over multiple turns.

In contrast, DSTC5 expand its scope to cover open domain dialogues, incorporating a broader range of topics, including emotionally sensitive conversations. This broader scope adds complexity, making it a valuable resource for systems that need to manage multi-topic and empathetic interaction effectively.

6.5 Microsoft Cortana dataset

The Microsoft Cortana [35] is a proprietary dataset that includes data from user interactions with Cortana voice assistant. It focuses on tasks such as alarm management, weather updates, and calendar scheduling. Although not publicly available, it plays a role in enhancing Cortana's natural language processing capabilities.

6.6 Facebook NLU dataset

Facebook's NLU dataset is designed for evaluating natural language understanding tasks such as setting alarms, creating reminders, and checking weather. The dataset includes multilingual corpus, making valuable resources for testing model performance across different languages.

6.7 CQUD (Chinese Question Answering User Dataset)

The CQUD [38], collected from Baidu's question answering platform, includes queries across domains, such as flights, weather, and express delivery, with a primary focus on Chinese language interactions.

6.8 CMRS (Chinese Meeting Room Scheduling) dataset

The CMRS dataset [48] is more domain-specific, focusing on dialogues related to meeting room reservations. Its structured, task-oriented nature makes it particularly relevant for dialogue systems.

6.9 TRAINS dataset

The TRAINS dataset [247] was developed at University of Rochester for natural language understanding and dialogue systems. It consists of dialogues where users plan and manage train routes and cargo schedules. The dataset includes dialogue transcripts, annotations for dialogue acts, and task descriptions, making it ideal for training models that must handle complex multi-turn interaction.

6.10 Rokid music dataset

Rokid Music [52] is a Chinese-language dataset containing voice commands for music-related tasks. It serves as a domain-specific resource for evaluating models that manage voice-activated music control, contributing to the development of more effective music recommendation systems.

6.11 TOP (Task Oriented Parsing) dataset

The TOP dataset [248] is designed for parsing complex, task-oriented queries in domains such as navigation and event search. It includes both flat and nested intent labels, offering a challenging environment for models tasked with managing multi-level intents and slot filling tasks. This structured dataset is particularly useful for evaluating models that need to handle complex user requests.

6.12 CAIS (Chinese AI Speaker) dataset

CAIS dataset [75] includes user queries aimed at Chinese AI-powered speakers, for home automation tasks. This dataset provides a valuable resource for training models tailored to smart home device control, enabling systems to manage a range of commands efficiently.

7 Challenges of joint learning classification datasets

The performance of joint learning models for intent detection and slot filling is closely tied to the quality, diversity, and structure of the datasets they are trained on. However, several key challenges persist that can hinder the development of robust and accurate models. These challenges arise from issues such as data quality, domain specificity, and class imbalance, all of which need to be carefully addressed to improve model performance across varied tasks. This section discusses the key challenges and their impact on the development of joint learning models.

7.1 Data quality and consistency

The quality and consistency of data play a role in the performance of joint learning models for intent detection and slot filling. Many datasets used for these tasks often rely on crowd-sourced annotations or user-contributed data. While this approach allows for the rapid collection of large datasets, it introduces inconsistencies, noise, and labeling errors [249]. Variability in how different annotators label intents and slots can lead to discrepancies that

negatively impact model training. Maintaining high-quality, consistent annotations across all categories is essential to ensure that models learn correct patterns and relationships between intents and slot values. Poor quality data can significantly degrade model performance and reliability [250].

7.2 Domain specificity

Many datasets used for joint learning tasks are domain-specific, focusing on areas such as travel booking, technical support, or weather enquiry. While this specialization allows models to excel in particular domains, it poses challenges for developing models that generalize multiple domains. Different domains exhibit distinct linguistic patterns, specialized terminologies, and unique types of intents and slots. These domain-specific characteristics make it difficult for models to perform effectively outside their training domains [250]. Achieving cross-domain generalization remains a key challenge, requiring models to be adaptable to varied datasets and tasks.

7.3 Imbalanced classes

Class imbalance is a common issue in many real-world datasets, where certain intents or slots dominate the distribution, while others are significantly underrepresented [251]. This disparity can lead to models to overfit on majority classes and struggle to accurately learn and predict minority classes, which are often critical for handling edge cases and providing a more robust user experience. For instance, in the ATIS dataset, approximately 75% of the samples correspond to the "atis-flight" intent, leaving other intents underrepresented [179]. Addressing this imbalance is essential for developing models that not only excel at predicting common intents but also perform effectively on rare ones.

8 Evaluation metrics for joint learning models

This section outlines the most used evaluation metrics for joint learning models in intent detection and slot filling. Typically, these models are assessed based on the performance of each subtask, with intent detection evaluated as a classification problem and slot filling as a labeling task.

8.1 Intent classification metrics

Intent classification involves classifying the user inputs into predefined classes, where each class represents a specific intent. Some of the key evaluation metrics for intent classification are accuracy, precision, recall, and F1.

i. Accuracy

Accuracy is the most used evaluation metric for intent classification. It is calculated as the ratio of correct intent predictions to total sentences.

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{number of sentences}} \tag{36}$$

However, accuracy can be interpreted differently in multi-intent utterances. Some researchers consider a prediction to be correct if at least one intent is accurately detected [37, 58], while others [62, 232] require all intent labels to be correctly detected for the prediction to be deemed correct.

ii. Error Rate

Error rate is used to measure the proportion of incorrect predictions. This metric offers a direct perspective on the model's failure rate, highlighting areas where the model's performance falls short. Several studies

[38, 52, 55, 58, 98, 218, 252, 253] have employed error rate to assess the accuracy of intent label predictions. It is typically calculated using the following equation:

$$\text{error rate} = \frac{\text{incorrect predictions}}{\text{total number of predictions}} \tag{37}$$

$$\text{error rate} = 1 - \text{accuracy} \tag{38}$$

iii. Intent Recall, Precision, and F1

Though less frequently used as intent detection evaluation metrics, recall, precision, and F1-score, offer more granular insights into model performance. These metrics are typically computed using a confusion matrix. The confusion matrix provides a view of the model's predictive performance by comparing the predicted labels with actual labels. The matrix is organized into four quadrants, each representing a different combination of predicted and actual labels.

True Positive (TP): The model correctly predicted the positive class.
False Positive (FP): The model incorrectly predicted the positive class.
True Negative (TN): The model correctly predicted the negative class.
False Negative (FN): The model incorrectly predicted the negative class.

For multi-intent problems, recall and precision can be averaged across all intent classes using micro and macro averaging techniques:

Micro averaging considers individual true positives and false positive across classes:

$$\text{micro\_averaging Recall} = \frac{\sum TP}{\sum TP + \sum FP} \tag{39}$$

$$\text{micro\_averaging Precision} = \frac{\sum TP}{\sum TP + \sum FN} \tag{40}$$

Macro averaging computes the precision and recall for each class, then averages across classes:

$$\text{macro\_averaging Precision} = \frac{1}{\text{number of intent classes}} \sum \frac{TP}{TP + FP} \tag{41}$$

$$\text{macro\_averaging Recall} = \frac{1}{\text{number of intent classes}} \sum \frac{TP}{TP + FN} \tag{42}$$

The F1-score, which combines precision and recall using harmonic means, is particularly valuable in datasets with class imbalance [49, 231, 254–256]:

$$F1 = 2\left(\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right) \tag{43}$$

8.2 Slot filling metrics

The performance of slot filling is commonly evaluated using micro-averaged precision, recall, and F1-scores [179]. However, a few studies have also used accuracy as a metric for slot filling [188, 257]. Slot filling tasks are typically assessed using two methods: span-based and token-based evaluation.

i. *Span-based evaluation* focuses on consecutive words that represent a specific slot. For instance, "New York City" in an utterance would be labeled as a single span using IOB tagging as B-New, I-York, I-City. The span is considered correct only if all the tokens within it are predicted accurately. This approach has been employed in various studies [12, 26, 115].

ii. *Token-based evaluation* focuses on evaluating each token individually, without requiring the entire span to be correctly predicted [58, 73].

8.3 Semantic accuracy

Semantic accuracy refers to the model's ability to correctly interpret and classify user intents while accurately identifying and extracting relevant entities (slots) from an utterance. It is typically measured as the ratio of correctly analyzed utterances to the total number of utterances. Semantic accuracy has been used in several studies [46, 100, 231, 232] to evaluate the performance of joint models.

9 Applications of joint learning models

Joint learning for intent detection and slot filling has transformed the landscape of natural language understanding, finding applications across various domains. This section highlights real-world applications where joint learning has demonstrated significant effectiveness.

9.1 Virtual assistants and chatbots

Virtual assistants and chatbots are widespread in applications ranging from customer support to personal productivity tools. Joint learning models excel in these scenarios by providing a unified approach for understanding the user's intent and extracting relevant information. This leads to more responsive and context-aware conversational agents, thereby improving user experience [13].

9.2 Voice-activated system

In voice-activated systems, where spoken language serves as primary input, joint learning models play a role in accurately interpreting user requests. These models are integral to applications such as smart home devices, in-car assistants, and hands-free technology. By efficiently recognizing user intents and extracting relevant slots, joint learning ensures smoother interactions, enhancing the responsiveness and usability of these systems [15].

9.3 Information retrieval and search engines

Enhancing search engine capabilities and information retrieval are important joint learning applications. These models can contribute to more precise and contextually aware search results by accurately discerning user

intentions and extracting relevant details from the search queries. This is particularly beneficial in scenarios in which users employ natural language queries rather than conventional keyword-based searches [16].

9.4 Health care and medical assistance

In the healthcare domain, joint learning models facilitate medical assistants and applications that require a clear understanding of patient queries and the accurate extraction of medical information. These models enhance communication between users and healthcare systems, supporting tasks such as appointment scheduling, symptom analysis, and medication management [82].

10 Open issues and future directions

10.1 Lack of explainability

Joint learning models for intent detection and slot filling are designed to enhance natural language understanding. However, the lack of explainability remains a significant challenge. This issue arises from the complex, nonlinear interactions between intent detection and slot filling components, which make it difficult to interpret how specific input features influence predictions [258]. Furthermore, the black-box nature of these models obscures the decision-making processes, hindering the understanding of why certain words or phrases are linked to specific intentions or slots [259].

Recent research suggests that leveraging information theory and explainable attention mechanisms offers promising approaches to address these challenges [258]. Improving the interpretability of joint models, are essential to fostering trust in NLU systems, thereby unlocking their full potential across diverse applications and domains.

10.2 Generalization to new domain

Ensuring that joint learning models generalize effectively to new domains remains a challenge. Domain-specific variations in language such as specialized terminology can lead to misinterpretation of intents and slots. Additionally, data scarcity in new domains hinders effective model training, limiting the ability to capture domain-specific linguistic structure. Although transfer learning offers a solution, it requires careful fine-tuning to avoid negative transfer, where knowledge from the source domain degrades performance in the target domain. Researchers are exploring techniques such as domain adaptation techniques [16], few-shot learning [80, 260], and meta-learning strategies [99] to address these challenges, enabling models to adapt efficiently to new domains while maintaining performance.

10.3 Real-time processing

Real-time systems require a low latency to provide immediate feedback. However, joint learning models, built on deep learning architectures, are often computationally intensive, leading to delays that impact user experience. Another issue arises from the need for models to adapt to changing contexts in real-time to maintain accuracy. This challenge is compounded by the limited computational resources available on many real-time devices, which restrict the deployment of complex models.

To overcome these limitations, researchers are exploring lightweight architectures and efficient algorithms that can deliver high performance while minimizing resource consumption [89, 261].

10.4 Datasets

The development of robust joint learning models also depends on the availability of diverse datasets and benchmark tasks that reflect real-world use cases. Most of the current datasets are often limited to single-turn utterances with only one intent per input, whereas real-world conversations frequently involve multiple intents and complex dialogues [30]. Furthermore, there is a need for datasets that support multilingual dialogue systems and domain adaptability to enhance the robustness of joint models. Expanding datasets to capture these dimensions will provide better evaluation frameworks and improve the real-world applicability of joint learning models.

10.5 Enhance contextualized understanding

Advancing contextual understanding requires the integration of external knowledge sources, semantic representations, and contextual-aware embedding. Although existing models demonstrate some degree of contextual awareness, developing methodologies to further enhance contextual modeling remains an open research challenge. Future work may focus on novel architectures that leverage world knowledge and context at deeper semantic levels to improve model accuracy and robustness in complex conversational scenarios.

11 Conclusion

Joint learning classification for intent detection and slot filling has become a significant research area in NLU due to its crucial role in improving human–computer interaction. This review examines joint learning models, ranging from classical machine learning approaches to advanced deep learning architectures, highlighting the impact of recent innovations. While classical models laid the groundwork, their limitations in capturing complex language dependencies paved the way for deep learning models. Notably, the introduction of transformer architectures, especially models like BERT, has enabled deep contextual understanding and fine-grained attention mechanisms, substantially improving accuracy and interpretability in joint learning tasks. A recent trend involves the resurgence of classical models, such as CRF, integrated within deep learning frameworks to enhance performance. Various joint learning architectures have been analyzed, revealing that fused and explicit approaches offer greater explainability compared to implicit architectures. However, the results are influenced by the datasets and methodologies employed. Challenges related to commonly used datasets are also highlighted, emphasizing the need for collaborative efforts to establish standardized, community-wide benchmarks. Regarding evaluation metrics, accuracy for intent detection and the F1-score for slot filling are the most widely used, enabling consistent model comparisons. Despite the progress achieved with deep learning and transformer-based models, challenges persist, such as addressing data scarcity, adapting models across domains, and ensuring efficiency for real-time applications. Future research directions should prioritize the development of lightweight transformer models and efficient architectures suitable for deployment in resource-constrained environments. Additionally, exploring multi-task learning frameworks, few-shot learning, and domain adaptation will be essential to broaden the applicability of these models. Sustained collaboration and innovation will continue to advance NLU, ensuring that joint learning models meet the demands of increasingly complex real-world applications.

Author contributions Yusuf Idris Muhammad conducted the research and prepared the manuscript under the supervision and review of Naomie Salim and Anazida Zainal. Sinarwati Mohammad Suhaili provided additional support through proofreading.

Funding This research was funded by the Ministry of Higher Education Malaysia and Universiti Teknologi Malaysia (UTM) under grant scheme FRGS/1/2022/ICT06/UTM/01/1 with grant number R.J130000.7851.5F568.

Data availability This study is a review of existing literature and does not involve the generation or analysis of new data. As such, there are no new data associated with this paper. All data referred to in this review are available from the original sources cited in the text.

Declarations

Conflict of interest The authors declared that they have no known competing financial interests or personal relationships that could appear to influence the work reported in this article. The authors also declared non-financial interests which may be considered as potential conflicts of interest.

Ethical approval It is not applicable.

References

1. Xiaojie Wang CY (2016) Recent advances on human-computer dialogue. CAAI Trans Intell Technol 1(4):303–312
2. Hasani MF, Pratama GD, Erna F (2023) Multimodal learning conversational dialogue system: methods and obstacles. J Theor Appl Inf Technol 101(22):7235
3. Razumovskaia E et al (2022) Crossing the conversational chasm: a primer on natural language processing for multilingual task-oriented dialogue systems. J Artif Intell Res 74:1351–1402
4. Gupta A et al. (2019) Casa-nlu: Context-aware self-attentive natural language understanding for task-oriented chatbots. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing(EMNLP-IJCNLP'19). 2019. Association for Computational Linguistics
5. Algherairy A, Ahmed M (2023) A review of dialogue systems: current trends and future directions. Neural Computing and Applications. p. 1–27
6. Kane B et al. (2021) Joint intent detection and slot filling via CNN-LSTM-CRF. In: 2020 6th IEEE congress on information science and technology (CiSt).. IEEE
7. Schuurmans J, Frasincar F (2019) Intent classification for dialogue utterances. IEEE Intell Syst 35(1):82–88
8. Louvan S, Magnini B (2020) Recent neural methods on slot filling and intent classification for task-oriented dialogue systems: a survey. In: Proceedings of the 28th international conference on computational linguistics
9. Huang L et al (2023) A fast attention network for joint intent detection and slot filling on edge devices. IEEE Trans Artif Intell. https://doi.org/10.1109/TAI.2023.3309272
10. Wu J et al. (2023) A graph-to-sequence model for joint intent detection and slot filling. In: 2023 IEEE 17th International Conference on Semantic Computing (ICSC). IEEE
11. Firdaus M et al (2021) A deep multi-task model for dialogue act classification, intent detection and slot filling. Cogn Comput 13(3):626–645
12. Firdaus M, Ekbal A, Cambria E (2023) Multitask learning for multilingual intent detection and slot filling in dialogue systems. Inf Fusion 91:299–315
13. Suhaili SM, Salim N, Jambli MN (2021) Service chatbots: a systematic review. Expert Syst Appl 184:115461
14. Chen S, Yu S (2019) Wais: Word attention for joint intent detection and slot filling. In: Proceedings of the AAAI conference on artificial intelligence
15. Lim J et al (2022) intent classification and slot filling model for in-vehicle services in Korean. Appl Sci 12(23):12438
16. Wang H et al. (2023) Joint modeling method of question intent detection and slot filling for domain-oriented question answering system. Data Technologies and Applications
17. Jeong M, Lee GG (2008) Triangular-chain conditional random fields. IEEE Trans Audio Speech Lang Process 16(7):1287–1302
18. Mairesse F et al. (2009) Spoken language understanding from unaligned data using discriminative classification models. In: 2009 IEEE international conference on acoustics, speech and signal processing. IEEE
19. Wang, Y.-Y (2010) Strategies for statistical spoken language understanding with small amount of data-an empirical study. In: Eleventh annual conference of the international speech communication association
20. Celikyilmaz A, Hakkani-Tur D (2012) A joint model for discovery of aspects in utterances. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)
21. Li Z et al (2021) A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst 33(12):6999–7019
22. Das S et al (2023) Recurrent neural networks (RNNs): architectures, training tricks, and introduction to influential research. Mach Learn Brain Disorders 23:117–138
23. Vaswani A et al. (2017) Attention is all you need. Adv Neural Inf Process Syst. 30

24. Patel R, Patel S (2021) Deep learning for natural language processing. In: Information and communication technology for competitive strategies (ICTCS 2020) intelligent strategies for ICT. Springer
25. ESezerer, Tekir S (2021) A survey on neural word embeddings. arXiv:2110.01804
26. Zhu M, Xu X (2024) ID-SF-Fusion: a cooperative model of intent detection and slot filling for natural language understanding. Data Technologies and Applications
27. Hui Y et al. (2021) Joint intent detection and slot filling based on continual learning model. In: ICASSP 2021–2021 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
28. Bhasin A et al. (2020) Parallel intent and slot prediction using mlb fusion. In: 2020 IEEE 14th international conference on semantic computing (ICSC). IEEE
29. Tur G, De Mori R (2011) Spoken language understanding: systems for extracting semantic information from speech. Wiley
30. Stefan Larson, Leach K (2022) A survey of intent classification and slot-filling datasets for task-oriented dialog. arXiv:2207.13211
31. Patil R et al. (2023) A survey of text representation and embedding techniques in nlp. IEEE Access
32. Ni, J., et al., Recent advances in deep learning based dialogue systems: A systematic survey. Artificial intelligence review. p. 1–101
33. Weld H et al. (2021) A survey of joint intent detection and slot filling models in natural language understanding. ACM Computing Surveys (CSUR)
34. Xu P, Sarikaya R (2013) Convolutional neural network based triangular crf for joint intent detection and slot filling. In: 2013 ieee workshop on automatic speech recognition and understanding. IEEE
35. Guo D et al. (2014) Joint semantic utterance classification and slot filling with recursive neural networks. In: 2014 IEEE spoken language technology workshop (SLT). 2014. IEEE
36. Liu C, Xu P, Sarikaya R (2015) Deep contextual language understanding in spoken dialogue systems. In: INTERSPEECH
37. Liu B, Lane I (2016) Attention-based recurrent neural network models for joint intent detection and slot filling. In: Proceedings of the annual meeting of the speech communication association (INTERSPEECH'16). p. 685–689
38. Zhang X, Wang H (2016) A joint model of intent determination and slot filling for spoken language understanding. In: IJCAI
39. Liu B, Lane I (2016) Joint online spoken language understanding and language modeling with recurrent neural networks. In Proceedings of the annual meeting of the special interest group on discourse and dialogue (SIGDIAL'16).p. 22–30
40. Hakkani-Tür D et al. (2016) Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. in Interspeech
41. Zhou Q et al (2016) A hierarchical lstm model for joint tasks. Chinese computational linguistics and natural language processing based on naturally annotated big data. Springer, pp 324–335
42. Kim YB, Lee S, Stratos K (2017) Onenet: Joint domain, intent, slot prediction for spoken language understanding. In: 2017 IEEE Automatic speech recognition and understanding workshop (ASRU). IEEE
43. Zheng Y, Liu Y, Hansen JH (2017) Intent detection and semantic parsing for navigation dialogue language processing. In: 2017 IEEE 20th international conference on intelligent transportation systems (ITSC)
44. Ma M et al. (2017) Jointly trained sequential labeling and classification by sparse attention neural networks. In: Proceedings of the conference of the international speech communication association(INTERSPEECH'17). p. ISCA, 3334–3338
45. Yang X et al. (2017) End-to-end joint learning of natural language understanding and dialogue manager. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
46. Zhang C et al. (2018) Joint slot filling and intent detection via capsule neural networks, in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics. p. 5259–5267
47. Wang Y, Tang L, He T (2018) Attention-based CNN-BLSTM networks for joint intent detection and slot filling. In: Chinese Computational linguistics and natural language processing based on naturally annotated big data: 17th China National Conference, CCL 2018, and 6th International Symposium, NLP-NABD 2018, Changsha, China, October 19–21, 2018, Proceedings 17. 2018. Springer
48. Wen L et al. (2018) Jointly modeling intent identification and slot filling with contextual and hierarchical information. In: Natural Language PROCESSING and Chinese Computing: 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8–12, 2017, Proceedings 6. Springer
49. Pan L et al. (2018) A multiple utterances based neural network model for joint intent detection and slot filling. In: CCKS Tasks
50. Ren S et al (2018) Joint intent detection and slot filling with rules. CCKS Tasks 2242:34–40
51. Firdaus M et al. (2018) A deep learning based multi-task ensemble model for intent detection and slot filling in spoken language understanding. In: Neural information processing: 25th international conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part IV 25. Springer

52. Yu S et al. (2018) : A novel attentive cross approach for joint intent detection and slot filling. In: 2018 International joint conference on neural networks (IJCNN). IEEE
53. Schumann R, Angkititrakul P (2018) Incorporating asr errors with attention-based, jointly trained rnn for intent detection and slot filling. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). 2018. IEEE
54. Wang Y, Shen Y, Jin H (2018) A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In" Proceedings of the North American chapter of the association for computational linguistics: human language technologies. Association for Computational Linguistics
55. Li C, Kong C, Zhao Y (2018) A joint multi-task learning framework for spoken language understanding. In: 2018 IEEE International conference on acoustics, speech and signal processing (ICASSP). IEEE
56. Zhang D et al. (2018) Attention-based RNN model for joint extraction of intent and word slot based on a tagging strategy. In: Artificial Neural networks and machine Learning–ICANN 2018: 27th international conference on artificial neural networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part III 27. Springer
57. Zhao X, Haihong E, Song M (2018) A joint model based on CNN-LSTMs in dialogue understanding. In: 2018 International conference on information systems and computer aided education (ICISCAE). 2018. IEEE
58. Li C, Li L, Qi J (2018) A self-attentive model with gate mechanism for spoken language understanding. In: Proceedings of the 2018 conference on empirical methods in natural language processing
59. Lee J et al. (2018) Coupled representation learning for domains, intents and slots in spoken language understanding. In 2018: IEEE Spoken Language technology workshop (SLT). IEEE
60. Castellucci G et al. (2019) Multi-lingual intent detection and slot filling in a joint bert-based model. arXiv:1907.02884
61. Chen Q, Zhuo Z, Wang W (2019) BERT for joint intent classification and slot filling. p. arXiv:1902.10909
62. Niu P, Chen Z, Song M (2019) A novel bi-directional interrelated model for joint intent detection and slot filling. In: The Proceedings of the 57th annual meeting of the association for computational linguistics. Association for computational linguistics
63. Zhang Z et al (2019) A joint learning framework with bert for spoken language understanding. Ieee Access 7:168849–168858
64. Firdaus M et al (2019) A multi-task hierarchical approach for intent detection and slot filling. Knowl-Based Syst 183:104846
65. Tingting C, Min L, Yanling L (2019) Joint intention detection and semantic slot filling based on blstm and attention. In: 2019 IEEE 4th international conference on cloud computing and big data analysis (ICCCBDA). IEEE
66. Dadas S, Protasiewicz J, Pedrycz W (2019) A deep learning model with data enrichment for intent detection and slot filling. In: 2019 IEEE International conference on systems, man and cybernetics (SMC). IEEE
67. Okur E et al. (2019) Natural language interactions in autonomous vehicles: Intent detection and slot filling from passenger utterances. In: International conference on computational linguistics and intelligent text processing. Springer
68. Chen M, Zeng J, Lou J (2019) A self-attention joint model for spoken language understanding in situational dialog applications. arXiv:1905.11393
69. Li Y et al. (2019) A joint model of clinical domain classification and slot filling based on RCNN and BiGRU-CRF. In: 2019 IEEE international conference on big data (Big Data). IEEE
70. Shan J et al. (2019) A neural framework for joint prediction on intent identification and slot filling. In: Cognitive Computing–ICCC 2019: third international conference, held as part of the services conference federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 3. 2019. Springer
71. Qin L et al. (2019) A stack-propagation framework with token-level intent detection for spoken language understanding. In: Proceedings of the conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing(EMNLP-IJCNLP'19), Association for Computational Linguistics,: p. 2078–2087
72. Gupta, A., J. Hewitt, and K. Kirchhoff. Simple, fast, accurate intent classification and slot labeling for goal-oriented dialogue systems. In: Proceedings of 20th annual sigdial meeting on discourse and dialogue. 2019. Association for computational linguistics
73. Li C, Zhao Y, Yu D (2019) Conditional joint model for spoken dialogue system. In: Cognitive Computing–ICCC 2019: third international conference, held as part of the services conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 3. Springer
74. Zhang S et al (2019) A novel slot-gated model combined with a key verb context feature for task request understanding by service robots. IEEE Access 7:105937–105947
75. Liu Y, et al. (2019) Cm-net: A novel collaborative memory network for spoken language understanding. In: Proceedings of the conference on empirical methods in natural language processing and 9th international joint conference on natural lnaguage processing(EMNLP-IJCNLP'19), Association for Computational Linguistics. p. 1051–1060
76. Do QNT, Gaspers J (2019) Cross-lingual transfer learning for spoken language understanding. In: ICASSP 2019–2019 IEEE international conference on acoustics, speech and signal Processing (ICASSP). IEEE

77. Pentyala S, Liu M, Dreyer M (2019) Multi-task networks with universe, group, and task feature learning. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Association for Computational Linguistics
78. Zhang L, Wang H (2019) Using bidirectional transformer-CRF for spoken language understanding. In:CCF International conference on natural language processing and Chinese Computing. Springer
79. Wang C, Huang Z, Hu M (2020) SASGBC: Improving sequence labeling performance for joint learning of slot filling and intent detection. In: Proceedings of 2020 the 6th international conference on computing and data engineering
80. Bhathiya HS, Thayasivam U (2020) Meta learning for few-shot joint intent detection and slot-filling. In: Proceedings of the 2020 5th international conference on machine learning technologies
81. Wu D et al. (2020) SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP). p. 1932–1937
82. Ni P et al (2020) Natural language understanding approaches based on joint task of intent detection and slot filling for IoT voice interaction. Neural Comput Appl 32:16149–16166
83. Hardalov M, Koychev NP (2020) Enriched pretrained transformers for joints lot filling and intent detection
84. Cao X et al. (2020) Balanced joint adversarial training for robust intent detection and slot filling. In: Proceedings of the 28th international conference on computational linguistics
85. Tang Y et al. (2020) D-GHNAS for joint intent classification and slot filling. In: Web and Big Data: 4th international joint conference, APWeb-WAIM 2020, Tianjin, China, September 18–20, 2020, Proceedings, Part I 4. Springer
86. Tang H, Ji D, Zhou Q (2020) End-to-end masked graph-based CRF for joint slot filling and intent detection. Neurocomputing 413:348–359
87. Chao W, Ke Y, Xiaofei W (2020) POS scaling attention model for joint slot filling and intent classification. In: 2020 IEEE 20th international conference on communication technology (ICCT). IEEE
88. Peng H et al. (2020) An interactive two-pass decoding network for joint intent detection and slot filling. In: Natural language processing and chinese computing: 9th ccf international conference, NLPCC 2020, Zhengzhou, China, October 14–18, 2020, Proceedings, Part II 9. Springer.
89. Louvan S, Magnini B (2020) Simple is Better! lightweight data augmentation for low resource slot filling and intent classification. In Proceedings of the 34th Pacific Asia conference on language, information and computation
90. Liu H et al. (2021) An explicit-joint and supervised-contrastive learning framework for few-shot intent classification and slot filling. arXiv:2110.13691
91. Priya N, Tiwari A, Saha S (2021) Context aware joint modeling of domain classification, intent detection and slot filling with zero-shot intent detection approach. In: Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part III 28. Springer
92. Sun R, Rao L, Zhou X (2021) Bidirectional information transfer scheme for joint intent detection and slot filling. In: 2021 17th international conference on computational intelligence and security (CIS). IEEE
93. Saha T et al. (2021) A transformer based multi-task model for domain classification, intent detection and slot-filling. In: 2021 International joint conference on neural networks (IJCNN). IEEE
94. Basu S et al. (2021) Semi-supervised few-shot intent classification and slot filling. arXiv:2109.08754
95. Gao W et al (2021) Dirichlet variational autoencoder for joint slot filling and intent detection. J Comput 32(2):61–73
96. Wang J et al. (2021) Encoding syntactic knowledge in transformer encoder for intent detection and slot filling. In: Proceedings of the AAAI conference on artificial intelligence
97. Yang P et al (2021) AISE: Attending to Intent and slots explicitly for better spoken language understanding. Knowl-Based Syst 211:106537
98. Zhou P et al. (2021) PIN: A novel parallel interactive network for spoken language understanding. In: 2020 25th international conference on pattern recognition (ICPR). IEEE
99. Li SW et al. (2021) Meta learning to classify intent and slot labels with noisy few shot examples. In: 2021 IEEE spoken language technology workshop (SLT). IEEE
100. Han SC et al. (2022) Bi-directional joint neural networks for intent classification and slot filling. In: The proceedings of the conference of the international speech communication association (INTERSPEECH'21). ISCA
101. Ma Z, Sun B, Li S (2022) A two-stage selective fusion framework for joint intent detection and slot filling. IEEE Trans Neural Netw Learn Syst 35:3874
102. Chen D et al. (2022) Towards joint intent detection and slot filling via higher-order attention. In: IJCAI
103. Wei P, Zeng B, Liao W (2022) Joint intent detection and slot filling with wheel-graph attention networks. J Intell Fuzzy Syst 42(3):2409–2420
104. Nguyen HH, Nguyen ND, Bui KHN (2022) Intent detection and slot filling with low resource and domain. In: Proceedings of the 11th international symposium on information and communication technology
105. Zhou B et al. (2022) Multi-grained label refinement network with dependency structures for joint intent detection and slot filling. arXiv:2209.04156
106. Abro WA et al (2022) Joint intent detection and slot filling using weighted finite state transducer and BERT. Appl Intell 52(15):17356–17370

107. Wei J (2022) A bert-based joint model of intent recognition and slot filling cross-correlation. In: 2022 5th International conference on data science and information technology (DSIT). IEEE
108. Li B, Wang W, Bao F (2022) Joint training model of intent detection and slot filling for multi granularity implicit guidance. In: 2022 International conference on asian language processing (IALP). IEEE
109. Zhao J, Yin S, Xu W (2022) Attention-based iterated dilated convolutional neural networks for joint intent classification and slot filling. In: CCF conference on big data. Springer
110. Fan J-F et al (2022) Intent-slot correlation modeling for joint intent prediction and slot filling. J Comput Sci Technol 37(2):309–319
111. Zhu Z et al. (2022) A graph attention interactive refine framework with contextual regularization for jointing intent detection and slot filling. In: ICASSP 2022–2022 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
112. Amoake M et al (2022) Strategies to improve few-shot learning for intent classification and slot-filling. SUKI 2022:17
113. Zhang W et al. (2022) A bert based joint learning model with feature gated mechanism for spoken language understanding. In: ICASSP 2022–2022 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
114. Hou C et al. (2023) Prior knowledge modeling for joint intent detection and slot filling. In: Machine learning, multi agent and cyber physical systems: proceedings of the 15th international FLINS Conference (FLINS 2022). World Scientific
115. Tavares, D., et al. (2023) Task conditioned BERT for joint intent detection and slot-filling. In: EPIA conference on artificial intelligence. Springer
116. Li Q (2023) Joint modelling of slot filling and intent detection in constrained resource scenarios. Front Comput Intell Syst 5(2):111–115
117. Lu Y et al. (2023) Research on the joint learning method of intent detection and slot filling by fusing tag semantic information. In: 2023 IEEE international conference on control, electronics and computer Technology (ICCECT). IEEE
118. Xu, H., H. Zhang, and T.E. Lin, A Dual RNN Semantic Analysis Framework for Intent Classification and Slot, in SpringerBriefs in Computer Science. 2023. p. 45–54
119. Wang Y, Yang Z, Zhang X (2023) Improving NLP accuracy with stack-propagation and knowledge distillation: a joint model for intent detection and slot filling. Front Comput Intell Syst 3(2):106–109
120. Liu G et al. (2023) A joint intent classification and slot filling method based on knowledge-distillation. In: 2023 8th IEEE international conference on network intelligence and digital content (IC-NIDC). IEEE
121. Hao X et al (2023) Joint agricultural intent detection and slot filling based on enhanced heterogeneous attention mechanism. Comput Electron Agric 207:107756
122. Wu Y et al (2023) Joint intent detection model for task-oriented human-computer dialogue system using asynchronous training. ACM Trans Asian Low-Resource Lang Inf Process 22(5):1–17
123. Gore S et al. (2023) Leveraging BERT for next-generation spoken language understanding with joint intent classification and slot filling. In: 2023 International conference on advanced computing technologies and applications (ICACTA). IEEE
124. Shafi N, Chachoo MA (2023) Fine-Tuned BERT with Attention-Based Bi-GRU-CapsNet framework for joint intent recognition and slot filing. In: 2023 International conference on advancement in computation and computer technologies, InCACCT 2023
125. Huang J, Tang H (2024) A joint model of multiple intent recognition and slot filling based on graph neural network
126. Park S, Menassa CC, Kamat VR (2024) Joint BERT model for intent classification and slot filling analysis of natural language instructions in co-robotic field construction work. In: Computing in Civil Engineering. p. 453–460
127. Cheng X et al. (2024) Towards multi-intent spoken language understanding via hierarchical attention and optimal transport. In: Proceedings of the AAAI conference on artificial intelligence
128. Raymond C, Riccardi G (2007) Generative and discriminative algorithms for spoken language understanding. In: Interspeech 2007–8th Annual conference of the international speech communication association
129. McCallum A, Freitag D, Pereira FC (2000) Maximum entropy Markov models for information extraction and segmentation. In: Icml
130. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
131. Warjri S et al (2021) Part-of-speech (pos) tagging using conditional random field (crf) model for khasi corpora. Int J Speech Technol 24(4):853–864
132. Patil N, Patil A, Pawar B (2020) Named entity recognition using conditional random fields. Proc Comput Sci 167:1181–1188
133. Yu B, Fan Z (2020) A comprehensive review of conditional random fields: variants, hybrids and applications. Artif Intell Rev 53(6):4289–4333
134. Liu T, Huang X, Ma J (2016) Conditional random fields for image labeling. Math Probl Eng 2016(1):3846125
135. Jbene M et al. (2022) A robust slot filling model based on lstm and crf for iot voice interaction. In: 2022 IEEE Globecom Workshops (GC Wkshps). IEEE

136. Zhang S et al (2022) Bi-LSTM-CRF network for clinical event extraction with medical knowledge features. IEEE Access 10:110100–110109
137. Xu Z et al. (2008) CRF-based hybrid model for word segmentation, NER and even POS tagging. In: Proceedings of the sixth SIGHAN workshop on Chinese language processing
138. Chen Y, Luo Z (2023) Pre-trained joint model for intent classification and slot filling with semantic feature fusion. Sensors 23(5):2848
139. Patel D, Saxena S, Verma T (2016) Sentiment analysis using maximum entropy algorithm in big data. Int J Innov Res Sci Eng Technol 5(5):8355
140. Bridges RA et al. (2013) Automatic labeling for entity extraction in cyber security. arXiv:1308.4941
141. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Icml. Williamstown, MA
142. Mor B, Garhwal S, Kumar A (2021) A systematic review of hidden Markov models and their applications. Arch Comput Methods Eng 28:1429–1448
143. Alva P, Hegde V (2016) Hidden Markov model for POS tagging in word sense disambiguation. In: 2016 International conference on computation system and information technology for sustainable solutions (CSITSS). IEEE
144. Cohen SN (2020) Uncertainty and filtering of hidden Markov models in discrete time. Probab Uncertain Quant Risk 5:1–34
145. Shmilovici A (2023) Support vector machines. In: Machine learning for data science handbook. L. Rokach, O. Maimon, and E. Shmueli, Editors. Springer, Cham
146. Puri S, Singh SP (2019) An efficient hindi text classification model using svm. In: Computing and network sustainability: proceedings of IRSCNS 2018. Springer
147. Sharma D, Sabharwal M (2019) Sentiment analysis for social media using SVM classifier of machine learning. Int J Innov Technol Exploring Eng (IJITEE) 8(9):39–47
148. Ramachandran R, Arutchelvan K (2020) Optimized version of tree based support vector machine for named entity recognition in medical literature. In: 2020 3rd International conference on intelligent sustainable systems (ICISS). IEEE
149. Piccialli V, Sciandrone M (2022) Nonlinear optimization and support vector machines. Ann Oper Res 314(1):15–47
150. Ferrario A, Nägelin M (2020) The art of natural language processing: classical, modern and contemporary approaches to text document classification. Modern and contemporary approaches to text document classification
151. Hadi MU et al. (2024) Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. Authorea Preprints
152. Cohn TA (2007) Scaling conditional random fields for natural language processing. University of Melbourne, Department of Computer Science and Software
153. Cumani S, Laface P (2012) Analysis of large-scale SVM training algorithms for language and speaker recognition. IEEE Trans Audio Speech Lang Process 20(5):1585–1596
154. You J et al (2020) Handling missing data with graph representation learning. Adv Neural Inf Process Syst 33:19075–19087
155. Merritt T (2017) Overcoming the limitations of statistical parametric speech synthesis
156. Socher R et al. (2011) Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of the 28th international conference on machine learning (ICML-11)
157. Chinea A (2009) Understanding the principles of recursive neural networks: A generative approach to tackle model complexity. In: Artificial Neural Networks–ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14–17, 2009, Proceedings, Part I 19. 2009. Springer
158. Sahay A et al. (2021) Unsupervised learning of explainable parse trees for improved generalisation. In: 2021 international joint conference on neural networks (IJCNN). 2021. IEEE
159. Wei C et al. (2023) An overview on language models: recent developments and outlook. arXiv:2303.05759
160. Farooq U, Mohd Rahim MS, Abid A (2023) A multi-stack RNN-based neural machine translation model for English to Pakistan sign language translation. Neural Comput Appl 35(18):13225–13238
161. Patel P, Patel D, Naik C (2020) Sentiment analysis on movie review using deep learning RNN method. Intelligent Data Engineering and analytics: Frontiers in intelligent computing: theory and applications (FICTA 2020), vol 2. Springer, pp 155–163
162. Shewalkar A, Nyavanandi D, Ludwig SA (2019) Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. J Artif Intell Soft Comput Res 9(4):235–245
163. Abro WA et al (2020) Multi-turn intent determination and slot filling with neural networks and regular expressions. Knowl-Based Syst 208:106428
164. Mensio M, Rizzo G, Morisio M (2018) Multi-turn qa: A rnn contextual approach to intent classification for goal-oriented systems. In: Companion proceedings of the the web conference
165. Shi Y, Wiggers P (2012) Towards recurrent neural networks language models with linguistic and contextual features
166. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
167. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

168. Bas E, Egrioglu E, Kolemen E (2022) Training simple recurrent deep artificial neural network for forecasting using particle swarm optimization. Granular Comput 7(2):411–420
169. Noh S-H (2021) Analysis of gradient vanishing of RNNs and performance comparison. Information 12(11):442
170. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
171. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. Neural Comput 12(10):2451–2471
172. Zhou H (2022) Research of text classification based on TF-IDF and CNN-LSTM. J Phys Conf Ser
173. Pulver A, Lyu S (2017) LSTM with working memory. In: 2017 international joint conference on neural networks (IJCNN). IEEE
174. Beck M et al. (2024) xLSTM: Extended long short-term memory. arXiv:2405.04517
175. Cho K et al. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP. Association for Computational Linguistics
176. Huang Z et al. (2021) Sentiment injected iteratively co-interactive network for spoken language understanding. In: ICASSP 2021–2021 IEEE international conference on acoustics, speech and signal processing (ICASSP). 2021. IEEE
177. Hassan A, Mahmood A (2018) Convolutional recurrent deep learning model for sentence classification. Ieee Access 6:13949–13957
178. Qin L et al. (2021) A co-interactive transformer for joint slot filling and intent detection. In: ICASSP 2021–2021 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
179. Weld H et al (2022) A survey of joint intent detection and slot filling models in natural language understanding. ACM Comput Surveys 55(8):1
180. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv:1409.0473
181. LeCun Y et al (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
182. Rehman AU et al (2019) A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis. Multimedia Tools Appl 78:26597–26613
183. Aslan S, Kızıloluk S, Sert E (2023) TSA-CNN-AOA: Twitter sentiment analysis using CNN optimized via arithmetic optimization algorithm. Neural Computing and Applications. p. 1–18
184. Wang H et al (2020) A short text classification method based on N-gram and CNN. Chin J Electron 29(2):248–254
185. Kia MA et al (2022) Adaptable closed-domain question answering using contextualized CNN-attention models and question expansion. IEEE Access 10:45080–45092
186. Meng F et al. (2015) Encoding source language with convolutional neural network for machine translation. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (Volume 1: Long Papers)
187. Chiu JP, Nichols E (2016) Named entity recognition with bidirectional LSTM-CNNs. Trans Assoc Comput Linguistics 4:357–370
188. Barahona LMR et al. (2016) Exploiting sentence and context representations in deep neural models for spoken language understanding. arXiv:1610.04120
189. Kim Y (2014) Convolutional neural networks for sentence classification
190. Vu NT (2016) Sequential convolutional neural networks for slot filling in spoken language understanding. arXiv preprint arXiv:1606.07783
191. Rowtula V, Krishnan P (2018) Pos tagging and named entity recognition on handwritten documents. In: Proceedings of the 15th international conference on natural language processing
192. Gui T et al. (2019) CNN-Based Chinese NER with lexicon rethinking. In: ijcai
193. Young T et al (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13(3):55–75
194. Knigge DM et al. (2022) Modelling long range dependencies in nd: from task-specific to a general purpose CNN. In: The eleventh international conference on learning representations
195. Niu Z, Zhong G, Yu H (2021) A review on the attention mechanism of deep learning. Neurocomputing 452:48–62
196. Wang B, Liu K, Zhao J (2016) Inner attention based recurrent neural networks for answer selection. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers)
197. Chaudhari S et al (2021) An attentive survey of attention models. ACM Trans Intell Syst Technol 12(5):1–32
198. Wang S et al. (2020) Linformer: Self-attention with linear complexity. arXiv:2006.04768
199. Sun C et al (2022) A joint model based on interactive gate mechanism for spoken language understanding. Appl Intell 52(6):6057–6064
200. Child R et al. (2019) Generating long sequences with sparse transformers. SarXiv:1904.10509
201. Pappas N, Popescu-Belis A (2017) Multilingual hierarchical attention networks for document classification. In: Proceedings of the eighth international joint conference on natural language processing (Volume 1: Long Papers)
202. Tian B et al. (2019) Hierarchical inter-attention network for document classification with multi-task learning. In: IJCAI
203. Yang Z et al. (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies

204. Li Z et al. (2018) Hierarchical attention transfer network for cross-domain sentiment classification. In: Proceedings of the AAAI conference on artificial intelligence
205. Cheng L, Yang W, Jia W (2023) A scope sensitive and result attentive model for multi-intent spoken language understanding. In: Proceedings of the AAAI conference on artificial intelligence
206. Meng X et al. (2023) Rethink the top-u attention in sparse self-attention for long sequence time-series forecasting. In: International conference on artificial neural networks
207. Correia GM, Niculae V, Martins AF (2019) Adaptively sparse transformers. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)
208. Liu Y et al. (2024) Understanding llms: a comprehensive overview from training to inference. arXiv:2401.02038
209. Michel P, Levy O, Neubig G (2019) Are sixteen heads really better than one? In: Proceedings of the 33rd international conference on neural information processing systems
210. Devlin J et al. (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805S
211. Kalyan KS, Rajasekharan A, Sangeetha S (2021) Ammus: A survey of transformer-based pretrained models in natural language processing. arXiv:2108.05542
212. Kenton JDMWC, Toutanova LK (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT
213. Radford A et al. Improving language understanding by generative pre-training
214. Yang Z et al. (2019) Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems. 32
215. Wang Y et al (2020) A new concept of multiple neural networks structure using convex combination. IEEE Trans Neural Netw Learn Syst 31(11):4968–4979
216. Tao S et al. (2021) Incorporating complete syntactical knowledge for spoken language understanding. In: Knowledge graph and semantic computing: knowledge graph empowers new infrastructure construction: 6th china conference, CCKS 2021, Guangzhou, China, November 4–7, 2021, Proceedings 6. Springer
217. Pham T, Tran C, Nguyen DQ (2023) MISCA: a joint model for multiple intent detection and slot filling with intent-slot co-Attention. arXiv:2312.05741
218. Shi Y et al. (2015) Contextual spoken language understanding using recurrent neural networks. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
219. Chen YN et al. (2016) Syntax or semantics? Knowledge-guided joint semantic frame parsing. In: 2016 IEEE spoken language technology workshop (SLT). 2016. IEEE
220. He T et al (2021) Multitask learning with knowledge base for joint intent detection and slot filling. Appl Sci 11(11):4887
221. Siddhant A, Goyal A, Metallinou A (2019) Unsupervised transfer learning for spoken language understanding in intelligent agents. In: Proceedings of the AAAI conference on artificial intelligence
222. Daha FZ, Hewavitharana S (2019) Deep neural architecture with character embedding for semantic frame detection. In: 2019 IEEE 13th International Conference on Semantic Computing (ICSC). IEEE
223. Mikolov T, Yih W, Zweig G (2013) Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies
224. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)
225. PetersME, Mohit lyyer MN, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: human language technologies
226. Pentyala S, Liu M, Dreyer M (2019) Multi-task networks with universe, group, and task feature learning. arXiv:1907.01791
227. Ren F, Xue S (2020) Intention detection based on siamese neural network with triplet loss. IEEE Access 8:82242–82254
228. Zhang L et al. (2020) Graph lstm with context-gated mechanism for spoken language understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence
229. Wang S et al. Sequential recommender systems: challenges, progress and prospects
230. Wang S et al. (2019) Sequential recommender systems: challenges, progress and prospects. In: 28th International joint conference on artificial intelligence, IJCAI 2019. p. 6332–6338
231. Gangadharaiah R, Narayanaswamy B (2019) Joint multiple intent detection and slot labeling for goal-oriented dialog. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers)

232. Goo CW et al. (2018) Slot-gated modeling for joint slot filling and intent prediction. In: Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: human language technologies, Volume 2 (Short Papers)
233. He K et al. (2020) Syntactic graph convolutional network for spoken language understanding. In: Proceedings of the 28th international conference on computational linguistics
234. Zhang L, Wang H (2019) Using bidirectional transformer-CRF for spoken language understanding. In: Natural language processing and chinese computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part I 8. Springer
235. Bhasin A et al. (2019) Unified parallel intent and slot prediction with cross fusion and slot masking. In: International conference on applications of natural language to information systems. Springer
236. Bhathiya HS, Thayasivam U (2020) Meta learning for few-shot joint intent detection and slot-filling. In: ACM international conference proceeding series
237. Krone J, Zhang Y, Diab M (2020) Learning to classify intents and slot labels given a handful of examples. In: In Proceedings of the 2nd workshop on natural language processing for conversational AI. Association for Computational Linguistics
238. Cheng L, Jia W, Yang W (2021) An effective non-autoregressive model for spoken language understanding. In: Proceedings of the 30th ACM international conference on information & knowledge management
239. Tu NA et al. (2023) A bidirectional joint model for spoken language understanding. In: ICASSP 2023–2023 IEEE international conference on acoustics, speech and signal processing (ICASSP). 2023. IEEE
240. Béchet F, Raymond C (2018) Is ATIS too shallow to go deeper for benchmarking Spoken Language Understanding models? In: InterSpeech
241. Zarcone A, Lehmann J, Habets EA (2021) Small data in nlu: Proposals towards a data-centric approach. In: 35th Conference on neural information processing systems (NeurIPS 2021)
242. Ray A, Shen Y, Jin H (2019) Iterative delexicalization for improved spoken language understanding. In: Proceedings of the conference of the international speech communication association(INTERSPEECH'19)., ISCA. p. 1183–1187
243. Wang Y et al. (2019) Effective utilization of external knowledge and history context in multi-turn spoken language understanding model. In: 2019 IEEE international conference on big data (big data)
244. Hemphill CT, Godfrey JJ, Doddington GR (1990) The ATIS spoken language systems pilot corpus. In: Speech and natural language: proceedings of a workshop held at hidden valley, Pennsylvania, June 24–27
245. Coucke A et al. (2018) Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. arXiv preprint arXiv:1805.10190
246. Schulz H et al. (2017) A Frame Tracking model for memory-enhanced dialogue systems. In: Proceedings of the 2nd workshop on representation learning for NLP
247. Sikorski T, Allen JF (1996) TRAINS-95 system evaluation. University of Rochester, Department of Computer Science
248. Zhao W et al. (2021) kNN-ICL: Compositional task-oriented parsing generalization with nearest neighbor in-context Learning. In: Proceedings of the 2024 conference of the North American chapter of the association for computational linguistics: human language technologies. 1
249. Yaghoub-Zadeh-Fard M-A et al (2020) User utterance acquisition for training task-oriented bots: a review of challenges, techniques and opportunities. IEEE Internet Comput 24(3):30–38
250. Casanueva I et al. (2022) NLU++: a multi-label, slot-rich, generalisable dataset for natural language understanding in task-oriented dialogue. In: Findings of the Association for Computational Linguistics: NAACL
251. Kraiem MS, Sánchez-Hernández F, Moreno-García MN (2021) Selecting the suitable resampling strategy for imbalanced data classification regarding dataset properties: an approach based on association models. Appl Sci 11(18):8546
252. Mohasseb A, Bader-El-Den M, Cocea M (2018) Classification of factoid questions intent using grammatical features. ICT Express 4(4):239–242
253. Ray SN et al. (2021) Listen with intent: improving speech recognition with audio-to-intent front-end. arXiv preprint arXiv:2105.07071
254. Liu H et al (2020) A hybrid neural network bert-cap based on pre-trained language model and capsule network for user intent classification. Complexity 2020:1–11
255. Liu Y et al. (2020) A hybrid neural network RBERT-C based on pre-trained RoBERTa and CNN for user intent classification. In: Neural Computing for advanced applications: first international conference. NCAA 2020, Shenzhen, China, July 3–5, 2020, Proceedings 1. Springer
256. Khattak A et al (2021) Applying deep neural networks for user intention identification. Soft Comput 25:2191–2220
257. Yu D, Wang S, Deng L (2010) Sequential labeling using deep-structured conditional random fields. IEEE J Sel Top Signal Process 4(6):965–973
258. Zhuang X, Cheng X, Zou Y (2024) Towards explainable joint models via information theory for multiple intent detection and slot filling. In: Proceedings of the AAAI conference on artificial intelligence
259. Gunaratna K et al. (2022) Explainable slot type attentions to improve joint intent detection and slot filling. In: Findings of the Association for Computational Linguistics: EMNLP 2022

https://doi.org/10.1007/s00521-025-11329-9

260. Hou YL, Yongkui, Chen, Cheng, Che, Wanxiang, Liu, Ting (2021) Learning to bridge metric spaces: few-shot joint learning of intent detection and slot filling. In: Findings of the Association for computational linguistics: ACL-IJCNLP 2021
261. Wu Y, Mao W, Feng J (2021) AI for online customer service: intent recognition and slot filling based on deep learning technology. Mobile Netw Appl 27:2305

Authors and Affiliations

Yusuf Idris Muhammad[1] · Naomie Salim[1] · Anazida Zainal[1] · Sinarwati Mohammad Suhaili[2]

✉ Yusuf Idris Muhammad
muhammadidris@graduate.utm.my

Naomie Salim
naomie@utm.my

Anazida Zainal
anazida@utm.my

Sinarwati Mohammad Suhaili
mssinarwati@unimas.my

[1] Faculty of Computing, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

[2] Pre-University, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia