2.1. Evolution of Intent Detection Approaches

The field of intention recognition has evolved significantly since its inception. Three main paradigms can be distinguished, starting with simple dictionary-based and rule-based methods [41]. A second major approach focuses on statistically based classification methods [24]. In the current landscape, intent detection is experiencing a remarkable moment of dynamism, adopting the most advanced techniques in deep learning and natural language processing. Different neural network architectures, from RNNs to transformer-based models, have been applied, including proposals based on models such as BERT [2, 59, 61].

2.2. Current Research Challenges

Despite significant progress, intent detection remains a challenging task due to the complexity of human interactions. Given the foundational nature of intent detection, as a core component of different language technology systems, research must address different challenges.

- **Multi-intent detection**: User utterance in conversational interactions often includes multiple intents simultaneously [30]. For example, in the sentence: *'Hello good morning, I would like to cancel the medical appointment I had and ask for a new one'* the user expresses three different intents: greeting, cancelation and rescheduling. Moreover, temporal dependencies between intents can arise, as in: *'I want to make a transfer but first I would like to know my balance'*. In this case, the execution of the second intent depends on the completion of the first. In addition, operational or interpretative dependencies may exist, as in: *'Turn off all the lights that are on and turn on all the lights that are off'*. In this case, determining which lights are affected by the second intent must be resolved before executing the action associated with the first intent.
- **Integration of intent detection and entity recognition**: The tasks of intent detection and entity recognition are deeply interconnected but have traditionally been treated as separate problems within the language understanding pipeline [2, 51, 58]. Entities may be linked to different intents according to semantic or pragmatic relationships. Hence, a correct identification of entities may depend to a large extent on the recognition of intents. For example, in the sentence *'I would like to book the flight departing at 9'*, the value *'9'* should be classified as a time, while in *'I would like to order 9'*, the same value *'9'* represents a quantity. Furthermore, entity recognition can clarify ambiguous intents, as in *'From Paris to New York next Monday'*.
- **Out-of-domain intent management**: Effective handling of out-of-domain intents is essential for real-world applications, particularly in domains like banking, public administration, and customer service [7, 32]. Users often express intents that fall outside the operational scope of a system, posing a challenge for intent detection. This complexity is amplified by a high level of overlap in the way intents can be expressed. For example, a citizen may request the renewal of a document from the wrong administration, but linguistically, the request resembles a legitimate query. Detecting and accurately classifying these cases remains a significant challenge.
- **Model explainability**: In many critical applications, it is not sufficient for a conversational system to simply classify the intent but also to provide a clear and understandable explanation associated with the decision [62]. Moreover, explainability contributes to ensure the robustness and reliability of conversational systems and the identification of potential biases.

2.3. Suitability for Dynamical Systems Analysis

The analysis of the intention detection task, together with its current research directions and the associated methodological and technological challenges, suggests that it is a highly suitable domain for the application of dynamical systems analysis. As we show in this paper, intent detection models typically operate on low-dimensional manifolds, even though they are implemented using high-dimensional neural architectures. This feature reflects the intrinsic nature of intent detection, which at a high level of abstraction can be described as a mapping operation from complex linguistic expressions to a relatively small catalog of semantic intents. The objective is to maintain strong robustness and flexibility across with respect to variability of expressions at all linguistic levels, including lexical-morphological, syntactic, and semantic variability. On the other hand, modeling based on specific fixed-point topologies with attractors and saddle points aligns well with the inherent structure of intent classification problems.

The targets in the intent catalog can ultimately be interpreted as convergence points of the entire set of linguistic variants that express or represent the same meaning (semantic-level cataloging) or are associated with the execution of the same operation (pragmatic-level cataloging). This mathematical and computational framework provides a suitable approach for addressing one of the key characteristics of human language, such as ambiguity and the transition between different interpretations.

A key intuition underlying this paper is that the process of understanding, which underpins intent classification, can be viewed as a trajectory in a multidimensional space defined by word embeddings as they are processed and transformed by the network in the hidden layer for each token of an input expression. Analyzing these trajectories through space offers valuable insights into the comprehension process, which can be seen as a nonlinear accumulation of information received from the input signal in sequential nature. Within this framework, attractor points can be interpreted as the centers of regions that represent the intents defined in the catalog of the working domain. This approach integrates motivations of a computational nature (rooted in deep learning models with architectures ranging from RNNs to Transformers), mathematical (through the use of a topological model that enables operations such as comparison, trajectory analysis, proximity assessment, sudden changes in direction and speed), and linguistic (addressing issues like ambiguity, synonymy, polysemy, and dependencies between linguistic levels). This work also offers a starting point for tackling the challenges in the field of intent detection. For instance, the state space topology provides a new perspective for addressing multi-intent detection and out-of-domain intent identification. Additionally, the low-dimensional manifold structure serves as a foundation for analyzing the interaction between intent detection and entity recognition within a unified semantic space. Crucially, this framework's ability to provide a mathematical foundation for understanding the internal workings of intent detection systems enhances their explanatory power, addressing the limitations of black-box models and promoting greater transparency.

3. Background

3.1. Recurrent Neural Networks Computations

Feedforward networks (FFNs) analyze inputs under the assumption that the order in which samples are processed does not carry meaningful information. However, in many real-world scenarios, the order of the components, or their temporal or spatial relationships, is crucial. For example, NLP problems are studied as token sequences, and weather forecasting relies on time series data [18]. In such cases, a mechanism for retaining and learning temporal dependencies is required. To address this limitation, FNNs can be enhanced with feedback connections, as illustrated in Figure 1. This enhancement gives rise to the architecture known as recurrent neural networks (RNNs) [15].

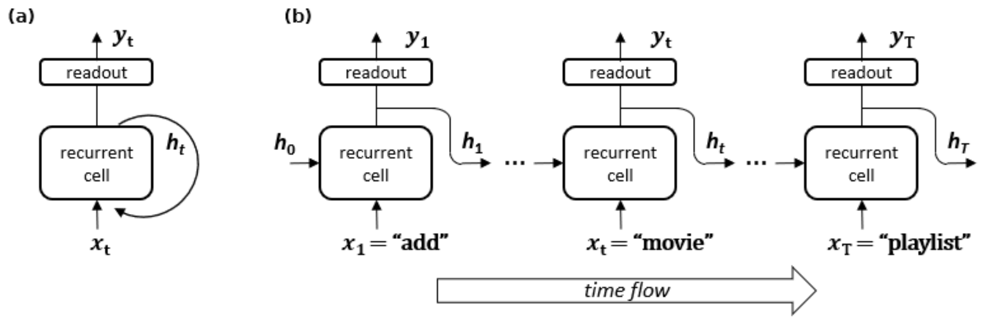

Fig. 1. **(a)** Folded representation of a Recurrent Neural Network (RNN), highlighting the recurrent connection within the architecture. **(b)** Unfolded representation of an RNN, explicitly showing the flow of time. At each time step t, the RNN processes an input token $\mathbf{x}_t$ (e.g. words in a sentence like "add", "movie", or "playlist"), updates its hidden state $\mathbf{h}_t$ based on the previous state $\mathbf{h}_{t-1}$, and generates an output $\mathbf{y}_t$. The initial hidden state $\mathbf{h}_0$ represents the starting point of the RNN's before processing any input tokens.

In general, the computations performed by RNNs can be summarized using the following pair of equations in difference:

$$\mathbf{h}_t = \mathbf{F}(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{1}$$

$$\mathbf{y}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b} \tag{2}$$

where t represents an integer index (commonly interpreted as time), $\mathbf{h}_t \in \mathbb{R}^n$ is the n-dimensional *hidden state* of the network, and $\mathbf{x}_t \in \mathbb{R}^m$ is the m-dimensional external input at step t. The nonlinear update function $\mathbf{F}$ defines how the hidden state evolves, depending on the specific recurrent cell architecture used (e.g. vanilla RNN, LSTM, GRU). Given an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the network updates its hidden state $\mathbf{h}_t$ at each step t based on the previous hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$. The predictions $\mathbf{y}_t$ are computed by passing the hidden states through a linear *readout layer*, where $\mathbf{W}$ is a $n \times n$ readout weight matrix and $\mathbf{b}$ is a bias vector. Each row $\mathbf{r}_i$ of $\mathbf{W}$ is referred to as a readout vector. In classification tasks, the output $\mathbf{y}_t$ consists of N logits, one for each class label. The way these outputs are generated depends on the specific problem. In the so-called many-to-many situations (e.g. named entity recognition), the entire sequence of hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_T$ is projected to produce a sequence of predictions $\mathbf{y}_1, \ldots, \mathbf{y}_T$. In contrast, in many-to-one contexts (e.g. intent detection or sentiment analysis), only the last hidden state $\mathbf{h}_T$ is used to generate a single prediction $\mathbf{y}_T$.

3.2. Fixed points

Systems governed by difference equations as in Equation 1 are called *discrete-time dynamical systems*, with their state represented by $\mathbf{h}_t$ at time t. In the context of RNNs, the update function $\mathbf{F}$ is inherently nonlinear, which classifies these systems as nonlinear discrete-time dynamical systems (NLDS) designed and tuned to perform specific tasks. Therefore, they can be analyzed using a wide variety of tools from the dynamical system theory. The vector state $\mathbf{h}_t \in \mathbb{R}^n$ can be visualized in a n-dimensional space known as the *state space* or the *phase space* of the system, where each axis corresponds to a component of $\mathbf{h}_t$. For a given initial state, the system evolves over time according to $\mathbf{F}$, producing a sequence of states that form a *trajectory* or *orbit* in the state space. These trajectories can exhibit highly diverse qualitative behaviors depending on the region of the phase space. As a result, a common approach is to partition the state space into regions and study the properties and interactions of these areas separately. A natural starting point for this analysis is the study of fixed points.

A *fixed point* or *equilibrium point*, denoted as $\mathbf{h}^*$, is defined as a zero-motion state (or an invariant point). By definition, if the system reaches $\mathbf{h}^*$, it will remain in it. In practical scenarios, noise or external perturbations may shift the system from a fixed point. Therefore, it is important to study the behavior of the system in the neighborhood of $\mathbf{h}^*$. If the system converges back to the fixed point after a small perturbation, $\mathbf{h}^*$ is classified as a *stable* fixed point. In contrast, if the system diverges from the fixed point, it is called *unstable*. An additional type of fixed point, known as a *saddle point* exhibits a mixed behavior. For certain trajectories, a saddle point behaves like a stable equilibrium, while for others, it behaves as an unstable point.

3.3. Linearization

Fixed points have a key property; the Hartman-Grobman theorem [19] states that the behavior of an NLDS near a fixed point $\mathbf{h}^*$ is qualitatively the same as the behavior of its linearization, provided that the fixed point is hyperbolic.[2] This equivalence means that the complex dynamics of an NLDS can be approximated locally by a linear system, greatly simplifying the analysis. The analysis of NLDS around fixed points is therefore performed in two steps: a) identifying the fixed points of the system, and b) analyzing the linearized behavior near these points. Linearization is obtained by calculating the *Jacobian matrix* of the system $\mathbf{JF}$ around the fixed point. This Jacobian provides a first-order approximation of the dynamics of the system in the vicinity of $\mathbf{h}^*$. Linearized system analysis involves decomposing the Jacobian matrix as $\mathbf{J} = \mathbf{R}\,\Lambda\,\mathbf{L}$, where Λ is a diagonal matrix that contains the eigenvalues λ_i of the

[2]A hyperbolic fixed point is one where none of the eigenvalues of the Jacobian have a magnitude equal to 1. For further details, see [19].