ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240928

Interpretation of the Intent Detection Problem as Dynamics in a Low-Dimensional Space

Eduardo Sanchez-Karhunen[a,*], Jose F. Quesada-Moreno[a] and Miguel A. Gutiérrez-Naranjo[a]

[a]University of Seville, Department of Computer Science and Artificial Intelligence. Seville, Spain
ORCID (Eduardo Sanchez-Karhunen): https://orcid.org/0000-0003-0136-3332, ORCID (Jose F. Quesada-Moreno): https://orcid.org/0000-0001-7458-5855, ORCID (Miguel A. Gutiérrez-Naranjo): https://orcid.org/0000-0002-3624-6139

Abstract. Intent detection is a text classification task whose aim is to recognize and label the semantics behind a users' query. It plays a critical role in various business applications. The output of the intent detection module strongly conditions the behavior of the whole system. This sequence analysis task is mainly tackled using deep learning techniques. Despite the widespread use of these techniques, the internal mechanisms used by networks to solve the problem are poorly understood. Recent lines of work have analyzed the computational mechanisms learned by RNNs from a dynamical systems perspective. In this work, we investigate how different RNN architectures solve the SNIPS intent detection problem. Sentences injected into trained networks can be interpreted as trajectories traversing a hidden state space. This space is constrained to a low-dimensional manifold whose dimensionality is related to the embedding and hidden layer sizes. To generate predictions, RNN steers the trajectories towards concrete regions, spatially aligned with the output layer matrix rows directions. Underlying the system dynamics, an unexpected fixed point topology has been identified with a limited number of attractors. Our results provide new insights into the inner workings of networks that solve the intent detection task.

1 Introduction

Modern recurrent neural networks (RNNs) are widely used to solve problems involving data sequences. Strong performance is obtained in tasks of natural language processing (NLP) such as sentiment analysis [27], intent detection and slot filling [13], and machine translation [40]. Unfortunately, this success has not been accompanied by a deep understanding of the internal mechanisms learned by RNNs to solve concrete tasks. The exact nature of their inner workings remains an open question. The nonlinear nature of RNNs combined with high-dimensional hidden layers poses constraints on our understanding of network behavior. In addition to that, practical solutions are evolving toward increasingly complex structures [3, 40]. This level of complexity makes it even more challenging to understand what is happening inside the nets. These neural networks are used in an ever-increasing number of areas, bringing significant advances in multiple domains. Their inclusion in automated reasoning systems with a high impact on society is becoming more common. These models find regularities in the data and give predictions without explicit rules governing their behavior, the known idea of neural networks as black boxes. Therefore, it is crucial to improve our understanding of how these models solve problems and make decisions in different situations. There is an increasing societal need to develop the interpretability of neural network decision making [14].

Several studies have tried to understand the behavior of RNNs by visualizing the activity of individual parts of the network (e.g. memory gates) during NLP tasks [22, 37]. However, the analysis at this unit level does not provide clear interpretations. The presence of feedback connections between RNN neurons allows their interpretation as nonlinear dynamical systems [31], opening the door to the use of a large set of well-known mathematical tools related to the theory of dynamical systems [38]. Based on this idea, several works have obtained complex analytic expressions for different aspects of network dynamics [17, 42], such as bifurcations in the parameter space of small networks and convergence analysis. In recent years, a new reverse engineering approach has emerged for the analysis of RNNs. Instead of paying attention to the microdetails of the trained RNN behavior, a higher-level approach is considered. The state space of the trained RNN is analyzed: fixed points are located, and the dynamics of the system is linearized around them. This line of research has revealed fundamental aspects of how RNNs implement their computations [39]. These techniques have been applied to text classification problems with promising results [2, 30]. A general idea that arises is that trained networks converge to highly interpretable, low-dimensional representations associated with attractors in the RNN state space. The geometry and dimensionality of these attractors manifolds depend on the task to be solved and the internal structure of the data set.

1.1 Our contributions

The main contribution of this paper is the pioneer study of the dynamics of the state space of trained RNNs for the SNIPS intent detection problem. We show that the state space is located in a manifold embedded in a low-dimensional space. The intrinsic dimensionality of this manifold is related to the size of the embedding layer and the number of neurons in the hidden layer. We also show that sentences fed into the network describe discrete trajectories through the state space toward its outer regions. A key point is the existence of distant regions from the initial states, where the trajectories end. We explain how predictions are possible due to the alignment between these peripheral areas and the directions determined by the readout matrix

* Corresponding Author. Email: fesanchez@us.es.

rows. The underlying fixed point topology of the system is obtained. Unlike sentiment analysis and document classification, we show that RNNs trained on the intent detection problem present an unexpected fixed point structure [2]. The number of attractors and saddle points learned by the network depends on the network parameters and the type of cell. We expect to generalize these promising results from the SNIPS dataset to generic intent detection problems.

2 Background

2.1 Recurrent Neural Networks computations

Feedforward networks (FFNs) perform a limited analysis of inputs based on the assumption of independence among the samples. However, in many situations, input samples are related in time or spatially; e.g. NLP problems are commonly studied as token sequences or the use of time series in weather forecast [15]. Some kind of memory is needed to learn the temporal information contained in the sequences. An option is to enrich the architecture of FFNs by including feedback or recurrent connections, as shown in Figure 1. This approach leads to the architecture called RNNs [12].

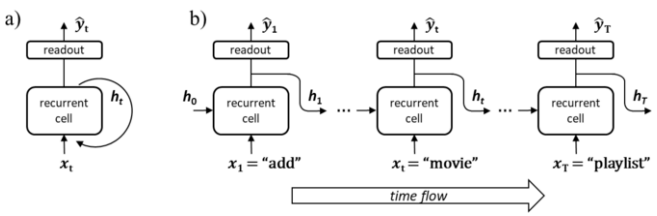

Figure 1. **a)** Folded representation of RNNs emphasizing the idea of recurrence. **b)** Unfolded RNN with explicit reference to time flow.

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

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

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

where t is an integer index (usually representing time), $\mathbf{h}_t \in \mathbb{R}^n$ is the n-dimensional *hidden state* of the network, $\mathbf{x}_t \in \mathbb{R}^m$ is the m-dimensional external input to be processed, and $\mathbf{F}$ is a nonlinear update function. The structure of $\mathbf{F}$ depends on the architecture selected to implement the recurrent cell. Given an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, in each computation step t, the network updates its state $\mathbf{h}_t$ as determined by $\mathbf{F}$, according to its previous state $\mathbf{h}_{t-1}$ and external input $\mathbf{x}_t$. To obtain the predictions of the model $\mathbf{y}_t$, these hidden states are passed through a linear *readout layer* that performs an affine projection, 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 called a readout vector. In classification problems, $\mathbf{y}_t$ consists of N output logits, one for each class label. In the so-called many-to-many situations (e.g. named entity recognition), the whole stream of hidden states, $\mathbf{h}_1, \ldots, \mathbf{h}_T$, is projected, obtaining 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), a single prediction $\mathbf{y}_T$ is obtained from the last hidden state $\mathbf{h}_T$.

2.2 Fixed points

Systems governed by difference equations as in Equation 1 are called *discrete-time dynamical systems* with state $\mathbf{h}_t$ at time t. In RNNs, the update function $\mathbf{F}$ is always nonlinear. Hence, RNNs are nonlinear discrete-time dynamical systems (NLDS) tuned to perform specific tasks. Therefore, they can be analyzed using a wide variety of well-known tools from dynamical system theory. The vector state $\mathbf{h}_t \in \mathbb{R}^n$ can be represented in a n-dimensional space called the *state space* or the *phase space* of the system, where each axis corresponds to a component of the state vector. For each initial state, the evolution of the system (governed by $\mathbf{F}$) is a sequence of states that describe a *trajectory* or *orbit* in the state space. The qualitative behavior of these trajectories can vary greatly between different parts of the phase space. Thus, a common line of work is to study separately different areas of the state space and the interactions between these regions. Fixed points are common points to start this analysis. A *fixed point* or *equilibrium point* $\mathbf{h}^*$ is a zero-motion point (or an invariant point) in the phase space. By definition, a system situated at a fixed point will remain in it. In real situations, noise or perturbations can shift the system from an equilibrium point. Therefore, it is important to understand the behavior of the system around $\mathbf{h}^*$. If the evolution of the system, when started in the neighborhood of $\mathbf{h}^*$, converges to the fixed point, $\mathbf{h}^*$ is called a *stable* fixed point. On the contrary, if the system diverges from it, the fixed point is called *unstable*. Finally, a *saddle point* behaves as a stable equilibrium point for some trajectories and as an unstable point for others.

2.3 Linearization

Fixed points have an important property; the Hartman-Grobman theorem [16] asserts that the behavior of an NLDS around a fixed point $\mathbf{h}^*$ has the same qualitative behavior[1] as its linearization $\mathbf{JF}(\mathbf{h}^*)$ (that is, the Jacobian of $\mathbf{F}$ around $\mathbf{h}^*$). Therefore, NLDS analysis is performed in two steps: a) to identify the fixed points of the system and b) to analyze their linearized behavior. Linearized systems analysis is much easier and involves the decomposition of the Jacobian $\mathbf{J} = \mathbf{R}\mathbf{\Lambda}\mathbf{L}$ where $\mathbf{\Lambda}$ is a diagonal matrix such that the elements λ_i are the n eigenvalues of an n-dimensional linear system and $\mathbf{L} = \mathbf{R}^{-1}$. Each λ_i has an associated eigenvector $\mathbf{r}_i$, row of the matrix $\mathbf{R}$. From this matrix decomposition, the state of a n-dimensional linear dynamical system can be interpreted as the linear composition of n independent one-dimensional exponential dynamics (also called *modes* or patterns of activity). Each of them takes place along the direction of the state space given by the eigenvector $\mathbf{v}_i$. These directions are invariant lines in the phase space. Therefore, the motion near $\mathbf{h}^*$ can be obtained as the linear composition of these n one-dimensional systems. The behavior of a mode along the direction $\mathbf{v}_i$ is controlled by its associated eigenvalue λ_i given by $\lambda_i^t b_i$. If $|\lambda_i| > 1$, the component $\mathbf{h}_t$ in direction $\mathbf{v}_i$ grows exponentially. On the contrary, if $|\lambda_i| < 1$, it shrinks exponentially. Therefore, the stability of $\mathbf{h}^*$ is determined by the spectral radius of $\mathbf{JF}(\mathbf{h}^*)$. If all λ_i are within the unit circle, then $\mathbf{h}^*$ is a stable fixed point. In a saddle point, some eigenvalue has a norm greater than 1. On the other hand, if every λ_i is beyond the unit circle, then $\mathbf{h}^*$ is a totally unstable equilibrium point.

2.4 Basins of attraction and saddle points

In many common situations, systems without external input evolve toward certain regions of the state space; such a converging point or region is called an *attractor*. Stable fixed points are the simplest attractors. The *basin of attraction* of an attractor is the region of the

[1] Only valid for hyperbolic fixed points. An interested reader can find more information at [16].

phase space (i.e. the set of all initial states) from which the system evolves towards the attractor. Any initial condition in that region will be iterated in the attractor. The whole state space is divided into basins of attraction associated with the attractors. In this partition, saddle points play a key role in controlling the interaction between attractors. Saddle points mostly have stable modes (i.e. modes associated to $\lambda_i < 1$), also called *stable manifolds* with only a small set of unstable modes (associated to $\lambda_i > 1$), also known as *unstable manifolds*. A region of state space can be funneled through its many stable modes and then sent to different attractors forced by an unstable mode. As a result of these state-space management operations, a stable saddle point manifold becomes a frontier between the basins of attraction [6]. The *index* of a saddle point is defined as the number of unstable manifolds of the fixed point.

2.5 Reverse engineering RNNs of classification tasks

RNNs, as NLDS, can be analyzed with tools of dynamical system theory. Modern RNN architectures are made up of hidden layers with hundreds of neurons, which implies high-dimensional hidden states. Traditional dynamical system analysis is performed considering individual neurons as parameters of the system. This high dimensionality of RNNs makes a standard analysis of state spaces difficult. A recent line of work considers the computational mechanisms learned by RNNs from a higher-level perspective [39]. These reverse engineering techniques are applied to analyze the dynamics of networks trained for specific tasks. The behavior of an RNN can be inferred from the structure of its state space: the fixed points, their linearized dynamics, and the interactions between these equilibrium points. For example, binary sentiment analysis and text classification problems share a common underlying mechanism [2]. In solving the task, their hidden state trajectories lie largely in a low-dimensional subspace of the full state space. An attractor manifold lies in this subspace that accumulates evidence (that is, keeps track) for each class as they process tokens of the text. The concrete dimensionality and geometry of this attractor manifold are determined by the structure of the dataset. In binary sentiment classification tasks, hidden states move along a line of stable fixed points [30]. In general, the attractors of a categorical classification of N classes form a $(N-1)$ dimensional simplex [2]. This dimensionality reflects the number of scalar quantities that the network remembers to classify.

2.6 Intent detection problem

Spoken Language Understanding (SLU) [33] is an important element of many natural language tools, such as dialogue systems. Its role is to capture the semantics of user utterances for use in other processes (i.e. question-answering or dialogue management). Three key tasks are involved in building this semantic frame: domain classification, intent detection, and slot filling, as shown in Table 1 [41]. The intent is the speaker's desired outcome from the utterance. Only if the user's intent is clearly identified can the query be routed to the correct subsystem.

Table 1. Example of utterance, its domain, intent and slots.

query	find	recent	comedies	by	James	Cameron
slots	O	B-date	B-genre	O	B-dir	I-dir
intent	find_movie					
domain	movies					

Two datasets have been widely used as benchmarks for intent detection models. First, SNIPS [8] is a balanced dataset with 7 intents, designed in the context of English voice assistants. On the contrary, ATIS [18] is a heavily imbalanced 26-intent dataset, which contains real conversations with English-speaking customers who request information about flights. Some of its utterances are labeled with more than one intent. Recently, a new multilingual MASSIVE dataset [10] was released. In this case, the original 60-intent SLURP dataset [4] has been localized in 51 different languages. The number of domains, intents, and slots in each of these datasets is shown in Table 2.

Table 2. Intent detection datasets comparison.

Name	Langs	Utters per lang	Domains	Intents	Slots
SNIPS	1	14484	-	7	53
ATIS	1	5871	1	26	129
MASSIVE	51	19521	18	60	55

The degree of imbalance of each dataset is shown in Table 3. The ATIS dataset is extremely imbalanced, with almost 74% of the samples belonging to a single intent. The MASSIVE dataset presents a large number of intents combined with a certain degree of imbalance. In this paper, we focus on the SNIPS dataset, trying to avoid possible side effects due to imbalance or a high number of intents.

Table 3. Degree of imbalance of the different datasets.

Name	#samples in larger intent	larger intent (%)	#samples in smaller intent	smaller intent (%)
SNIPS	2100	14.5	2042	14.1
ATIS	4298	73.7	1	0.02
MASSIVE	1190	6.9	6	0.04

Intent detection is usually approached as a supervised classification task that associates the entire input sentence with a label (or intent) of a finite set of classes [28]. Given the ability of RNNs to capture temporal dependencies, RNNs have been widely used to solve intent detection problems [34].

3 Experiments

We have divided our analysis into four steps: a) train different RNN architectures to solve the intent detection problem for the SNIPS dataset; b) obtain the state space learned by the RNN; c) analyze the manifold structure in which the state space is embedded; and d) obtain the structure of fixed points that underlies these trained RNNs.

We use TensorFlow 2 [1] to train basic RNNs with a trainable embedding layer with *embed_dim* neurons (i.e. no pretrained embeddings have been considered), a unidirectional RNN layer with *hidden_dim* neurons, and a final dense layer. Tokenization was implemented using a TextVectorization layer with a vocabulary truncated to 1K words (from a total of 10.5K words). Three different implementations of recurrent cells were considered: standard (or vanilla), LSTM [19], and GRU [7]. RNNs have been trained on the SNIPS [8] intent detection datasets. The optimized loss function is the usual one for classification problems (i.e. multiclass cross-entropy). Network training was carried out using an Adam optimizer [25] with a batch size of 32, and a learning rate $\eta =$ 5e-4. The number of training epochs was determined using early stopping [32] with a patience of 2 epochs. No additional hyperparameters were tuned, using the default value for the rest of the parameters. During training, no dropout [36]

or other regularization techniques have been used. For each architecture, we selected the best performing network based on a validation dataset. These validation subsets were obtained as a random sample 20% from the training dataset. Given the balanced distribution of the SNIPS target class, we evaluated the performance of the model by computing the accuracy in a test dataset not used during the training process. Dataset was randomly divided 80/20 to obtain the train and test subsets, respectively.

4 Results

4.1 Intent detection low-dimensional dynamics

In this section we show that the state space learned by an RNN is constraint to a low-dimensional hypersurface (or manifold). Before sentences are injected into RNNs, a tokenization mechanism is needed to transform each natural language phrase into a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, where $\mathbf{x}_i \in \mathbb{R}^m$ is a m-dimensional vector and T is the number of tokens in the sentence [21]. As shown in Figure 2, the sequential injection of these tokens generates a sequence of activations $\mathbf{h}_1, \ldots, \mathbf{h}_T$ in the output of the hidden layer. Each hidden state $\mathbf{h}_i \in \mathbb{R}^n$ is a n-dimensional vector (with $n = hidden\_dim$) given by Equation 1. The set of hidden states visited by sentences injected into a network is called the state space learned by the trained RNN.

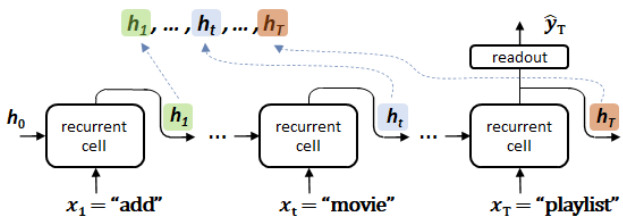

Figure 2. Sequence of hidden states associated to a tokenized input sentence fed into a RNN.

Under the assumption that the discrete points of the state space can be summarized by a manifold embedded in a higher-dimensional space, the next natural question is to determine its intrinsic dimensionality. Different measures of this dimensionality have been proposed [26, 5]. However, related works suggest that the variance explained threshold is the one that best fits the empirical data in classification tasks [2]. This measure considers the number of dimensions of the principal component analysis (PCA) [20] needed to reach a certain percentage of the variance explained. This threshold is typically set at a fixed value 95%.

Following this procedure, all sentences from the SNIPS test dataset were injected into a trained RNN. The state space points were concatenated, and by performing a principal component analysis, the variance captured by each principal component was obtained. In Figure 3 the accumulated explained variance versus the number of principal components is shown for two combinations of $hidden\_dim$ and $embed\_dim$. Variances for the vanilla, LSTM, and GRU cell types are represented in green, yellow, and blue, respectively. The horizontal dashed red line indicates the variance explained threshold at 95%. The vertical red line shows the number of principal components needed to surpass this threshold, i.e. the (intrinsic) dimensionality d of the state space. For simplicity, we denote by *cell_type(e:x,h:y)* an RNN with *cell_type* recurrent unit, *x* neurons in the embedding layer and a hidden layer of size *y*

From related work, the state space dimensionality of RNNs solving categorical text classification problems is $N - 1 \ll hidden\_dim$, with N the number of classes [2]. According to these proposals, the expected dimensionality d_e of the 7-class SNIPS dataset must be $N - 1 = 6$. We analyze whether this assertion also holds for the intent detection problem. A comparative analysis of the dimensionality of the state space and the accuracy of RNNs trained on the SNIPS dataset is shown in Figure 4.

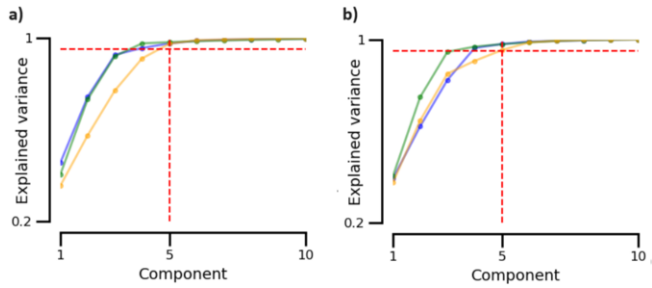

Figure 3. Variance explained of visited states vs principal components of a RNN trained on the SNIPS dataset. **a)** GRU(e:16,h:16). **b)** GRU(e:10,h:10).

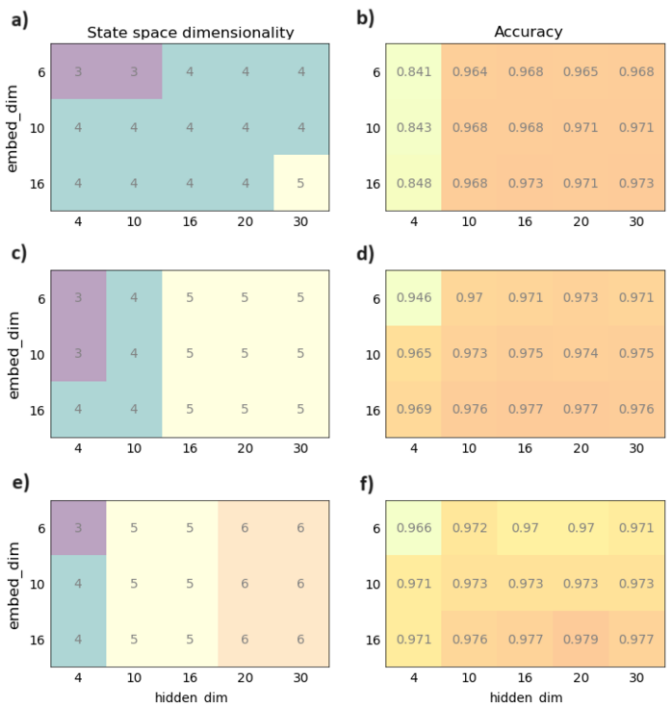

Figure 4. State space dimensionality and accuracy of RNNs trained on the SNIPS dataset for different embed_dim and hidden_dim combinations. **(a,b)**: Vanilla cell. **(c,d)**: GRU cell. **(e,f)**: LSTM cell

Different combinations of embedding and hidden layer sizes have been tested. The values presented correspond to the average accuracy and median state space dimensionality of ten trains with different seeds for each pair *(hidden_dim, embed_dim)*. Contrary to our initial expectations, the dimensionality of the state space depends on both network design parameters. Furthermore, at least in the range of tested parameters, for any *embed_dim*, a *hidden_dim* could be found such that it solves the problem with a state space manifold of dimensionality $d < d_e \ll n\_hidden$ (without accuracy degradation).

4.2 Intent detection state space projection

In the following sections, we analyze the spatial arrangement of hidden states in trained RNNs, taking advantage of the low dimensionality ($d << n\_hidden$) of their state spaces. For that, given a state space, it can be projected in the linear subspace given by the k-top principal components of Section 4.1. For this, a linear transformation $\mathbf{U}$ is applied to each hidden state:

$$\mathbf{p}_i = \mathbf{h}_i \mathbf{U} \tag{3}$$

where $\mathbf{h}_i \in \mathbb{R}^n$ is a n-dimensional hidden vector, $\mathbf{p}_i \in R^k$ is the k-dimensional projected hidden state and $\mathbf{U}$ is a $n \times k$ projection matrix. Top-2 and top-3 projections are generally considered for visualization purposes; meanwhile, higher-dimensional projections are useful for dimensionality analysis (as in Section 4.1). In Figures 5a) and b), 2D and 3D projections of the state space learned by a GRU(e:16,h:16) are shown. Each state is colored according to the intent of its source sentence. In both projections, hidden states appear seemingly grouped on the basis of their associated intent.

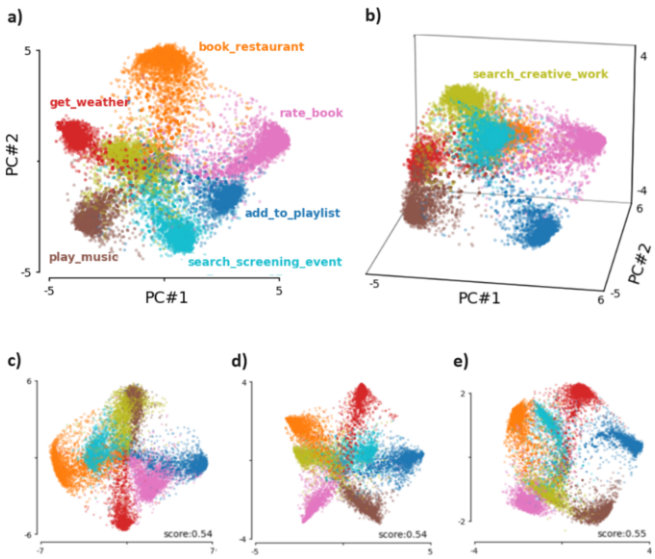

Figure 5. State space top-2 and top-3 PCA projections of RNNs trained on the SNIPS dataset. **a)** GRU(e:16,h:16). **b)** GRU(e:16,h:16). **c)** Vanilla(e:20,h:20). **d)** LSTM(e:10,h:10). **e)** GRU(e:10,h:10).

To numerically verify the presence of these clusters, we have applied a classical KMeans [29] clustering method with 7 clusters, random initial centroids, and Euclidean distance metric. The resulting state space partition was evaluated against the true labels, with a silhouette technique [35]. The silhouette coefficient of a point measures its intra-cluster and the nearest-cluster distances to the rest of points. The silhouette coefficients take values in the range [-1, 1]. A coefficient near +1 indicates that the point is far from neighboring clusters. Points close to the decision boundary between two neighboring clusters present values around 0. Finally, negative values indicate that the sample might have been assigned to a wrong cluster. The silhouette score is obtained by computing the mean silhouette coefficient on all samples. The score threshold, which is used to assess the quality of a cluster, is commonly set at 0.5. A score above 0.5 indicates a high-quality cluster.

Figure 6 shows the silhouette scores for a trained GRU(e:16,h:16) in two different situations. In Figure 6a) the distances between the points were calculated considering only the top-5 projections of the hidden states (since the intrinsic dimensionality of this RNN configuration is $d = 5$ from Section 4.1). On the other hand, in Figure 6b) fully 16-dimensional hidden states were considered. In both cases, the resulting partitions have similar global scores (dashed red line) above the 0.5 threshold. This comparison confirms the utility of analyzing the state spaces manifold considering uniquely the top-d projections, with d the intrinsic dimensionality of the state space. The

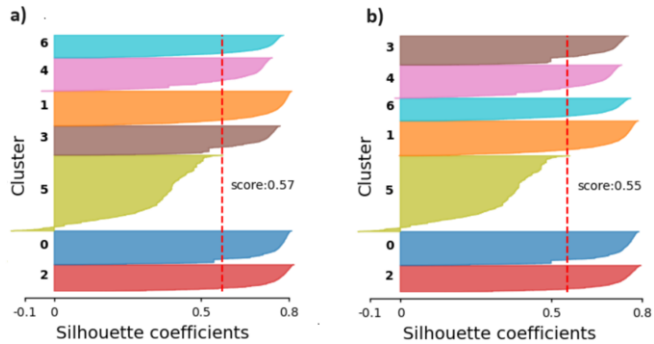

Figure 6. Silhouette evaluation of a GRU(e:16,h:16) state space KMeans partition. **a)** Distances obtained considering only top-5 PCA projections. **b)** 16-dimensional hidden states considered for distances.

partition of the state space in different regions associated with intents is independent of the type of cell, $hidden\_dim$ and $embed\_dim$. In Figures 5a), b) and c) the projections of the top-2 state space projections of different networks are shown for a Vanilla(e:20,h:20), a LSTM(e:10,h:10), and a GRU(e:10,h:10). For all configurations, the silhouette scores associated with a KMeans partition of the state space confirm the presence of clusters of states.

4.3 Sentences trajectories

In this section, we study how sentences move through state space when injected into a trained RNN. In practice, the state space of an RNN is obtained by feeding sentences to the network and aggregating the resulting visited states. In response to each input $\mathbf{x}_i$ the current state $\mathbf{h}_t$ is updated according to Equation 1 obtaining the next state $\mathbf{h}_{t+1}$. Hence, an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, when injected into the network, produces a new sequence $\mathbf{h}_1, \ldots, \mathbf{h}_T$ that describes a trajectory traversing the state space. The points of the trajectory can be projected onto the principal components of the low-dimensional state space using Equation 3.

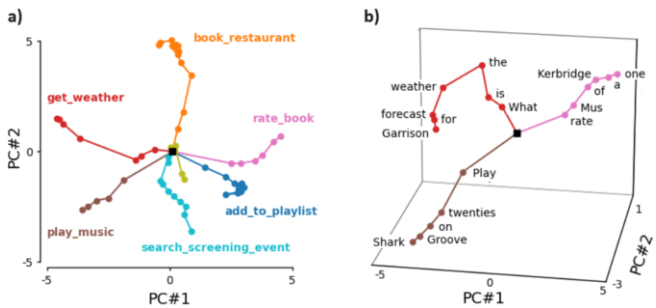

Figure 7. Trajectories of example sentences projected on the state space of a GRU(e:16,h:16) trained on the SNIPS. **a)** Top-2 PCA projections. **b)** Top-3 PCA projections.

Figure 7a) presents the 2D projection of the trajectories associated with an example sentence for each intent. Each $\mathbf{h}_i$ has been

highlighted with bullets. The initial state $\mathbf{h_0}$ of the recurrent cells is marked by black squares, and lines joining the states were added to emphasize the sequence of movements. In Figure 7b) details of three sentences are shown. In red, with "*get_weather*" intent: "*What is the weather forecast for Garrison*". In brown, with "*play_music*" intent; "*Play twenties on Groove Shark*" and, with intent *"rate_book"*, in purple: "*Rate Mus of Kerbridge a one*"). The input tokens are indicated next to the hidden state generated. As tokens are received, the orbits diverge from the origin moving towards concrete outer areas of the state space.

4.4 Model inference mechanism

In this section, we analyze the regions of the state space toward which RNNs direct the trajectories. In intent detection problems, the final hidden state of each sentence plays a crucial role. As shown in Figure 2, for prediction purposes, intermediate states are discarded, and only the last hidden state is considered. Given a sentence, the logits of each class are obtained projecting the last hidden state $\mathbf{h}_T$ onto each of the readout vectors $\mathbf{r}_i$ (that is, the rows of the readout matrix $\mathbf{W}$) as follows:

$$\mathbf{y} = \mathbf{y}_T = \mathbf{W}\mathbf{h}_T = [\mathbf{r}_1|\dots|\mathbf{r}_n]^T\mathbf{h}_T \tag{4}$$

The index i with the highest scalar value $\mathbf{r}_i^T\mathbf{h}_T$ will be output as the predicted intent. In Figures 8a) and b), the last hidden states $\mathbf{h}_T$ of each sentence in the SNIPS test dataset are projected onto its principal components. The states are colored according to the intent of the source sentence. The readout vectors $\mathbf{r_i}$ can be interpreted as directions that can also be projected in the state space. $\mathbf{r_i}$'s are represented following the same color schema. For illustration purposes, some trajectories have been added. We hypothesize that trajectory of sentences evolve through the low-dimensional state space towards concrete peripheral areas distant from the initial states.

We performed a 7-cluster KMeans analysis on the set composed of the final state of each sentence in the SNIPS test dataset. The results for different RNN configurations are presented in Table 4. In all cases, silhouette scores greater than 0.75 clearly confirm the presence of as many clusters of final states as intents. For each configuration, the distances $ds = \{ds_1, \dots, ds_7\}$ between the centroid of each cluster and the initial state $\mathbf{h_0}$ were calculated. In all the configurations tested, the standard deviation of the distances $\sigma(ds)$ is much lower than its mean value $\bar{ds}$. These results suggest that centroids are located in positions roughly equidistant from the initial state $\mathbf{h_0}$.

Table 4. Clusters of final states for different RNN configurations: silhouette scores, statistics of distances centroid - $\mathbf{h_0}$ and alignment data.

Cell type	embed dim	hidden dim	Silhouette score	Alignment mean	Distances $\bar{ds}$	$\sigma(ds)$
Vanilla	20	20	0.75	0.957	5.38	0.30
GRU	16	16	0.80	0.964	4.81	0.34
LSTM	10	10	0.81	0.963	4.09	0.73

Given a sentence with true intent I and final state $\mathbf{h_T}$, to correctly generate a prediction, the trained RNN must ensure that the value of $\mathbf{r}_i^T\mathbf{h}_T$ is greater for $i = I$ than for any other $i \neq I$. To maximize this dot product, for each intent I the readout vector $\mathbf{r_I}$ must be as aligned as possible with the final states $\{\mathbf{h_T}\}_\mathbf{I}$ of the cluster associated with the intent I. Cosine similarity is widely used to measure the degree of alignment between two vectors [9]. It computes the cosine of the angle between both vectors, taking values in the range [-1,1]. Similarity 1 indicates that both vectors are perfectly aligned, pointing in the same direction. Two orthogonal vectors present value 0. Similarities close to -1 indicate that the vectors are aligned but point in opposite directions. In Figure 8c) is shown the cosine similarity between all pairs $(r_i, centroid_j)$ for a trained GRU (e: 16, h: 16). As a result, each r_i has a single almost perfectly aligned centroid, with similarity values greater than 0.9. In Table 4 we have computed the mean value of the distances between the aligned pairs $r\_i, centroid\_j$ for different configurations.

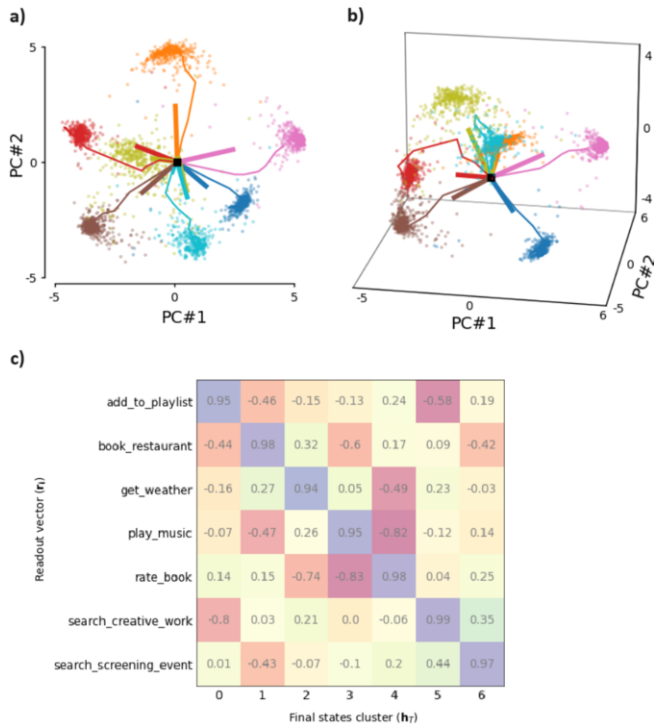

Figure 8. Final hidden states and readout vectors of a GRU(e:16,h:16). An example trajectory for each intent. **a)** Top-2 PCA projections. **b)** Top-3 PCA projections. **c)** Cosine similarity between clusters of final states and readouts.

4.5 Fixed point structure

In the previous section we have shown that sentences traverse a low-dimensional state space in a journey steered by the RNN towards final state clusters almost equidistant from the initial state. Each of these clusters is aligned with a paired readout vector to maximize its dot product. This dynamic suggests the existence of an underlying fixed point structure.

In general, we can write the first-order approximation to the RNN dynamics [24] around an expansion point $(\mathbf{h}_e, \mathbf{x}_e)$ as:

$$\mathbf{h}_t \approx \mathbf{F}(\mathbf{h}_e, \mathbf{x}_e) + \mathbf{J}^{rec}\mathbf{F}|_{(\mathbf{h}_e,\mathbf{x}_e)}\Delta\mathbf{h}_{t-1} + \mathbf{J}^{inp}\mathbf{F}|_{(\mathbf{h}_e,\mathbf{x}_e)}\Delta\mathbf{x}_t \tag{5}$$

where $\Delta\mathbf{h}_{t-1} = \mathbf{h}_{t-1} - \mathbf{h}_e$, $\Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_e$ and $\{\mathbf{J}^{rec}\mathbf{F}, \mathbf{J}^{inp}\mathbf{F}\}$ correspond to the Jacobian matrices of the update function $\mathbf{F}$ particularized at the expansion point. In concrete, the *recurrent Jacobian* $\mathbf{J}^{rec}\mathbf{F}$ captures the local dynamics associated with the recurrence, and the *input Jacobian* $\mathbf{J}^{inp}\mathbf{F}$ represents the sensitivity of the system to input tokens, with:

$$J^{rec}_{ij}\mathbf{F} = \left.\frac{\partial F_i}{\partial h_j}\right|_{(\mathbf{h}_e,\mathbf{x}_e)} \qquad J^{inp}_{ij}\mathbf{F} = \left.\frac{\partial F_i}{\partial x_j}\right|_{(\mathbf{h}_e,\mathbf{x}_e)} \tag{6}$$

When the expansion is performed in a fixed point, then $\mathbf{h}^* = \mathbf{F}(\mathbf{h}^*, \mathbf{x})$ and Equation 5 simplifies to a linear dynamical system equation:

$$\Delta\mathbf{h}_t = \mathbf{h}_t - \mathbf{h}_e \approx \mathbf{J}^{rec}\mathbf{F}|_{(\mathbf{h}^*,\mathbf{x}^*)}\Delta\mathbf{h}_{t-1} + \mathbf{J}^{inp}\mathbf{F}|_{(\mathbf{h}^*,\mathbf{x}^*)}\Delta\mathbf{x}_t \quad (7)$$

Due to the presence of time-dependent inputs $\mathbf{x_t}$ (that is, the tokens associated with sentences injected into the network), RNNs are nonautonomous dynamical systems. Analysis of nonautonomous systems requires sophisticated mathematical tools. For this reason, the reverse engineering point of view attempts to explain the behavior of the system in three steps: a) the obtention of the topological structure of the fixed points for constant input $\mathbf{x}_t$, b) the analysis of the linearized system around these fixed points under constant input, and c) the study of how the linearized behavior is altered (a.k.a. deflected) under the influence of nonconstant external inputs [2].

In this section, we focus on the first step, the identification of the fixed point structure. This approach starts with locating the set of points $\{\mathbf{h}_1^*, \mathbf{h}_2^*, \ldots\}$ in the state space such that $\mathbf{h}_i^* = \mathbf{F}(\mathbf{h}_i^*, \mathbf{x})$ where $\mathbf{x}$ is a constant input to the system. Typically, $\mathbf{x}^*$ is set to $\mathbf{0}$, with a clear physical meaning; given an initial state, the system on its own evolves, describing a trajectory without external energy injected into the system. In related work, not only are truly fixed points considered, but also very slow motion points [39]. In the following, we use the term fixed point to include not only classical fixed points but also these approximate fixed points, that is, $\mathbf{h}_i^* \approx \mathbf{F}(\mathbf{h}_i^*, \mathbf{0})$.

Different numerical procedures can be considered to identify fixed points [23, 39]. The speed of a point in a state space can be characterized as the magnitude q of the displacement generated by the update function $\mathbf{F}$ applied at the point $\mathbf{h}$:

$$q = \frac{1}{2}\|\mathbf{h} - \mathbf{F}(\mathbf{h}, \mathbf{0})\|_2^2 \quad (8)$$

We found slow motion (and zero-motion) points by performing a numerical optimization process that minimizes this quantity q [11]. Typically, state spaces have multiple regions with different behavior. For this reason, the optimization procedure must be run multiple times with different initial conditions (ICs). In our case, more than 25K initial states were considered extracted from the trajectories visited by the sentences in the SNIPS test dataset. The critical points of the loss function with a speed value $q < 10^{-8}$ were considered slow motion or approximate fixed points. Under the assumption of $\mathbf{x} = \mathbf{0}$, Equation 7 simplifies to:

$$\Delta\mathbf{h}_t = \mathbf{h}_t - \mathbf{h}_e \approx \mathbf{J}^{rec}\mathbf{F}|_{(\mathbf{h}^*,\mathbf{0})}\Delta\mathbf{h}_{t-1} \quad (9)$$

The stability of each approximate fixed point $\mathbf{h_i^*}$ is determined by the eigenvalues of $\mathbf{J}^{rec}\mathbf{F}|_{(\mathbf{h}_i^*,\mathbf{0})}$, the Jacobian evaluated at that point. In Table 5 the fixed points of different RNNs trained in the 7-class SNIPS dataset are presented. The number of intents involved in the problem acts as an upper bound of the number of attractors. In all cases, a group of saddle points is identified. The amount of critical points learned by the network depends on the type of cell and the size of the hidden layer and the embedding. The presence of saddle points with an index higher than one suggests the existence of a hierarchy of critical points, visited by the trajectories.

We analyze the location of the critical points projected in the top-3 components of the state space. The distances r_s, r_1, and r_2 (for stable, 1-index and 2-index, respectively) of each type of fixed points to the origin were obtained. The mean distances $\bar{r}$ and the standard deviations $\sigma(r)$ for different network configurations are presented in Table 6.

Table 5. Approximated fixed points ($q < 10^{-8}$) of RNNs trained on the SNIPS dataset.

Cell type	embed dim	hidden dim	#stable points	#saddle points 1-index	#saddle points higher-index
Vanilla	10	10	5	9	7
Vanilla	16	16	4	5	4
GRU	10	10	5	6	1
GRU	16	16	3	3	1

Table 6. Mean and standard deviation (in the projected state space) of distances to origin of attractors (r_s), 1-index (r_1) and 2-index saddles (r_2).

Cell type	embed dim	hidden dim	stable points $\bar{r}_s$	stable points $\sigma(r_s)$	1-index $\bar{r}_1$	1-index $\sigma(r_1)$	2-index $\bar{r}_2$	2-index $\sigma(r_2)$
Vanilla	10	10	3.93	0.10	3.52	0.13	3.07	0.31
Vanilla	16	16	4.73	0.29	4.25	0.38	4.03	0.36
GRU	10	10	3.72	0.32	3.43	0.33	1.72	0
GRU	16	16	4.71	0.15	4.27	0.18	3.29	0

Each family of fixed points can be located on the surface of a hypersphere of radius r_x with $x = e, 1, 2$. This radius depends on the dimension of the embedding and the hidden layer. In Figure 9 is shown the spatial arrangement of the critical points associated with a GRU(e:10,h:10). Each pair of attractors (in green) is separated by a 1-index saddle point (in red). The unique 2-index saddle point is marked in blue.

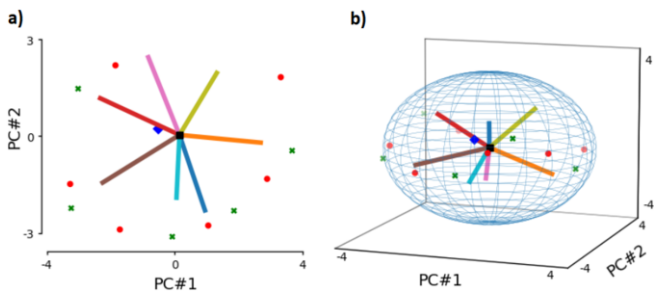

Figure 9. Attractors (green), 1-index saddle points (red), 2-index saddle point (blue) and readout vectors of a GRU(e:10,h:10) trained in the SNIPS dataset. **a)** Top-2 PCA projections. **b)** Top-3 PCA projections.

5 Conclusion

Intent detection is a hard problem which is not completely solved yet. In this paper, we have applied reverse engineering techniques to the intent detection SNIPS dataset with promising results. It is worthy to highlight that highly interpretable low-dimensional representations of their state space were obtained, and we have shown that sentences seem to travel towards concrete regions of the space to generate the predictions. Moreover, unexpected structures of fixed points have been identified underlying the network.

The work presented in this paper can be extended by following several lines, as exploring the role played by the fixed points and their basins of attraction, applying these ideas to other datasets to extend these results to a generic intent detection problem; or applying these techniques to other closely related problems as multi-intent and out-of-scope intent detection among many others. Finally, one particularly interesting line of work can focus on adapting this approach to Transformer architectures.

Acknowledgements

MAGN acknowledges the support by the European Union HORIZON-CL4-2021-HUMAN-01-01 under grant agreement 101070028 (REXASI-PRO) and by TED2021-129438B-I00 / AEI/10.13039/501100011033 / Unión Europea NextGenerationEU/PRTR.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: a system for large-scale machine learning. In *Proc. Conference on Operating Systems Design and Implementation (USENIX)*, pages 265–283, 2016.

[2] K. Aitken, V. V. Ramasesh, A. Garg, Y. Cao, D. Sussillo, and N. Maheswaranathan. The geometry of integration in text classification RNNs. In *International Conference in Learning Representation (ICLR)*, 2021.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference in Learning Representation (ICLR)*, 2015.

[4] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser. SLURP: A Spoken Language Understanding Resource Package. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7252–7262, 2020.

[5] F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407, 2002.

[6] A. Ceni, P. Ashwin, and L. Livi. Interpreting Recurrent Neural Networks Behaviour via Excitable Network Attractors. *Cognitive Computation*, 12(2):330–356, 2020.

[7] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[8] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau. Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces, 2018.

[9] P. Dangeti. *Statistics for machine learning: techniques for exploring supervised, unsupervised, and reinforcement learning models with Python and R*. O'Reilly, 2017.

[10] J. FitzGerald, C. Hench, C. Peris, S. Mackie, K. Rottmann, A. Sanchez, A. Nash, L. Urbach, V. Kakarala, R. Singh, S. Ranganath, L. Crist, M. Britan, W. Leeuwis, G. Tur, and P. Natarajan. MASSIVE: A 1M-Example Multilingual Natural Language Understanding Dataset with 51 Typologically-Diverse Languages, 2022.

[11] M. Golub and D. Sussillo. FixedPointFinder: A Tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of Open Source Software*, 3(31):1003, 2018.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 2016.

[13] D. Hakkani-Tür, G. Tur, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang. Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM. In *Proc. InterSpeech*, pages 715–719, 2016.

[14] R. Hamon, H. Junklewitz, and I. Sanchez. Robustness and explainability of Artificial Intelligence: from technical to policy solutions. Technical Report, Joint Research Center. European Commission, 2020.

[15] J. M. Han, Y. Q. Ang, A. Malkawi, and H. W. Samuelson. Using recurrent neural networks for localized weather prediction with combined use of public airport data and on-site measurements. *Building and Environment*, 192:107601, 2021.

[16] P. Hartman. *Ordinary differential equations*. Number 38 in Classics in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 2nd ed. reprinted from 2nd. ed. birkhauser 1982; reprinted from original john wiley & sons, 1964 edition, 2002.

[17] R. Haschke and J. J. Steil. Input space bifurcation manifolds of recurrent neural networks. *Neurocomputing*, 64:25–38, 2005.

[18] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington. The ATIS Spoken Language Systems Pilot Corpus. In *Human Language Technology (HLT)*, 1990.

[19] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[20] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 2nd edition, 2002.

[21] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Cambridge University Press, 3rd draft edition, 2021.

[22] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. In *International Conference in Learning Representation (ICLR)*, 2016.

[23] G. E. Katz and J. A. Reggia. Using Directional Fibers to Locate Fixed Points of Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3636–3646, 2018.

[24] H. K. Khalil. *Nonlinear Systems*. Pearson, 2013.

[25] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference in Learning Representation (ICLR)*, 2015.

[26] E. Levina and P. J. Bickel. Maximum Likelihood Estimation of Intrinsic Dimension. In *Proc. Neural Information Processing Systems Conference (NIPS)*, 2004.

[27] B. Liu. *Sentiment Analysis and Opinion Mining*. Cambridge University Press, 2015.

[28] J. Liu, Y. Li, and M. Lin. Review of Intent Detection Methods in the Human-Machine Dialogue System. *Journal of Physics: Conference Series*, 1267(1):012059, July 2019.

[29] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[30] N. Maheswaranathan, A. Williams, M. Golub, S. Ganguli, and D. Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, volume 32, pages 15670–15679, 2019.

[31] M. Martelli. *Introduction to discrete dynamical systems and chaos*. Wiley, New York, NY, 1st edition, 1999.

[32] L. Prechelt. Early Stopping — But When? In *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture notes in Computer Science*, pages 53–67. Springer Berlin Heidelberg, 1996.

[33] L. Qin, T. Xie, W. Che, and T. Liu. A survey on spoken language understanding: Recent advances and new frontiers, 2021.

[34] S. Ravuri and A. Stolcke. Recurrent neural network and LSTM models for lexical utterance classification. In *Proc. Interspeech*, pages 135–139, 2015.

[35] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[36] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[37] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. *IEEE Transactions on Visualization and Computer Graphics*, 24 (1):667–676, 2018.

[38] S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2nd edition, 2015.

[39] D. Sussillo and O. Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, 2013.

[40] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[41] G. Tur and R. De Mori, editors. *Spoken language understanding: systems for extracting semantic information from speech*. Wiley, 2011.

[42] Z. Yi and K. K. Tan. *Convergence Analysis of Recurrent Neural Networks*, volume 13. Springer US, 2004.