Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Dialog manager for conversational AI

Master's thesis

Bc. Petr Marek

Field of study: Open Informatics
Subfield: Artificial Inteligence
Supervisor: Ing. Jan Šedivý, CSc.

Prague, May 23, 2018

Thesis Supervisor:
Ing. Jan Šedivý, CSc.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Marek** Jméno: **Petr** Osobní číslo: **420127**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Dialogový manažer pro konverzační umělou inteligenci

Název diplomové práce anglicky:

Dialog manager for Conversational AI

Pokyny pro vypracování:

Review the current technologies for Conversational AI dialogue management. Design and implement a new dialog manager for intelligent speakers such as Alexa or Google Home. Train and test the dialog manager on a data collected from the Alquist rule-based application. Develop and test the dialog manager on favorite topics such as movies, sports or politics. Deploy the application and measure the performance of the implemented dialogue manager in real traffic.

Seznam doporučené literatury:

[1] Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning, Jason D. Williams, Kavosh Asadi, Geoffrey Zweig
[2] Convolutional Neural Networks for Sentence Classification, Yoon Kim
[3] End-to-end joint learning of natural language understanding and dialogue manager, Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, Li Deng
[4] Learning End-to-End Goal-Oriented Dialog, Antoine Bordes, Y-Lan Boureau, Jason Weston
[5] Enriching Word Vectors with Subword Information, Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Šedivý, CSc., velká data a cloud computing CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.02.2018** Termín odevzdání diplomové práce: ___________

Platnost zadání diplomové práce: **30.09.2019**

Ing. Jan Šedivý, CSc.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 23. 5. 2018

....
Bc. Petr Marek

Abstract

The dialogue management is a task of conversational artificial intelligence. The goal of the dialogue manager is to select the appropriate response to the conversational partner conditioned by the input message and recent dialogue state. The thesis aims to apply recurrent neural networks for dialog management task with as little hard-coded rules as possible. We will perform experiments on the Dialogue bAbI Task datasets and the dataset of dialogues collected from interactions between users and socialbot Alquist. We will measure the turn and dialogue accuracy of several architectures of dialogue manager. We will use Bayesian hyperparameter optimization to improve the accuracy of proposed architectures.

Keywords: Conversational AI, dialogue manager, hybrid code networks, recurrent neural networks, convolutional neural networks, word embeddings, bayesian hyperparameter optimization, dialogue bAbI Tasks

Abstrakt

Dialogový management je úkol konverzační umělé inteligence. Cílem dialogového manageru je zvolit vhodnou reakci pro dialogového partnera závislou na vstupní zprávě a aktuálním stavu dialogu. Cílem této práce je aplikovat rekurentní neuronové sítě na úkol dialogového managementu s použitím co nejmenšího počtu předpřipravených pravidel. Provedeme experimenty na souboru dialogových bAbI dat a na souboru dat dialogů získaných z interakcí mezi uživateli a socialbotem Alquistem. Změříme přesnost zvolení odpovědi a zvolení odpovědí v celém dialogu několika architektur dialogových manažerů. Použijeme Bayesovskou optimalizaci hyperparametrů pro zvýšení přesnosti navrhovaných architektur.

Klíčová slova: Konverzační umělá inteligence, dialogový manager, hybrid code networks, rekurentní neuronové sítě, konvoluční neuronové sítě, slovní embeddingy, bayesovská optimalizace hyperparametrů, dialogový bAbI dataset

Acknowledgements

I would like to thank my mentor Ing. Jan Sedivý, CSc. for creating work-space for students called eClub Prague. I would like to thank my colleagues who worked on Alquist during Alexa Prize competition.

List of Tables

5.1 List of predicted actions from Dialog bAbI task 5 30
5.2 List of predicted actions from Dialog bAbI task 6 30
5.3 List of clusters and examples of responses of Alquist conversational dataset . 33

6.1 bAbI Task 5 results 38
6.2 Hyperparameter optimization bounds 39
6.3 Word2vec bAbI Task 6 hyperparameter optimization results 39
6.4 Word2vec+CNN bAbI Task 6 hyperparameter optimization results 39
6.5 Word2vec+RNN bAbI Task 6 hyperparameter optimization results 40
6.6 fastText bAbI Task 6 hyperparameter optimization results 40
6.7 fastText+CNN bAbI Task 6 hyperparameter optimization results 40
6.8 fastText+RNN bAbI Task 6 hyperparameter optimization results 40
6.9 Testing accuracy 45

List of Figures

2.1 Relationships between word embedding vectors [14] 5
2.2 Continuous bag-of-words and skip-gram models [15] 5
2.3 Example of GloVe word-word co-occurrence matrix [12] 6
2.4 Perceptron [20] 7
2.5 Deep neural network [20] 8
2.6 Dropout [30] 10
2.7 Grid and random hyperparameter search [32] 11
2.8 Modules of recurrent neural network [41] 13
2.9 Modules of Long Short Term Memory [41] 13
2.10 Convolutional neural network [46] 14
2.11 Max pooling layer [49] 15

3.1 Seq2Seq [69] 19
3.2 Memory network [71] 20
3.3 Single and three-layer end-to-end memory network [72] 22
3.4 Hybrid code networks [1] 24

4.1 Hybrid code networks with bag-of-words and average of word embeddings input 25
4.2 Hybrid code networks with recurrent input layer 26
4.3 Hybrid code networks with convolutional input layer 26
4.4 Convolutional neural networks for sentence classification [65] 27

6.1 bAbI Task 6 Training turn accuracy 41
6.2 bAbI Task 6 Training dialogue accuracy 41
6.3 bAbI Task 6 Validation turn accuracy 42
6.4 bAbI Task 6 Validation dialogue accuracy 42
6.5 Alquist conversational dataset Training turn accuracy 43
6.6 Alquist conversational dataset Training dialogue accuracy 43
6.7 Alquist conversational dataset Validation turn accuracy 44
6.8 Alquist conversational dataset Validation dialogue accuracy 44

Contents

Abstract **vii**

Acknowledgements **ix**

List of Tables **xi**

List of Figures **xiii**

1 Introduction **1**
Motivation . . . 1
Thesis structure . . . 2

2 Theoretical Background **3**
2.1 Dialogue manager . . . 3
2.1.1 Context . . . 3
2.1.2 Retrieval-based dialogue managers . . . 3
2.1.3 Generative-based dialogue managers . . . 3
2.1.4 Rule-based dialogue managers . . . 4
2.1.5 End-to-end dialogue managers . . . 4
2.1.6 Hybrid-based dialogue managers . . . 4
2.2 Word embeddings . . . 4
2.2.1 Word2vec . . . 5
2.2.2 GloVe . . . 6
2.2.3 fastText . . . 6
2.3 Neural networks . . . 7
2.3.1 Training . . . 9
2.3.2 Overfitting . . . 10
2.3.3 Hyperparameters . . . 10
2.4 Recurrent neural networks . . . 12
2.4.1 Long Short Term Memory . . . 13
2.5 Convolutional neural networks . . . 14

3 Related work **17**
3.1 Rule-based systems . . . 17
3.2 TF-IDF match . . . 17
3.3 Semantic similarity . . . 17
3.4 Supervised models . . . 18
3.5 Seq2Seq . . . 18
3.6 Memory networks . . . 19
3.7 Hybrid code networks . . . 22

4 Implementations **25**
4.1 Hybrid code networks . . . 25
4.2 Hybrid code networks with recurrent neural network . . . 26
4.3 Hybrid code networks with convolutional neural network . . . 26

5 Datasets **29**
5.1 Dialog bAbI Tasks Data . . . 29
5.1.1 Dialog bAbI Task 5 . . . 29
5.1.2 Dialog bAbI Task 6 . . . 30
5.2 Alquist conversational dataset . . . 32

6 Experiments **37**
6.1 bAbI Task 5 . . . 37
6.2 bAbI Task 6 . . . 38
6.2.1 Bayesian hyperparameter optimization . . . 38
6.3 Alquist conversational dataset . . . 41
6.4 Results . . . 43

7 Conclusion **47**

A Dialog bAbI task 5 example **55**

B Dialog bAbI task 6 example **57**

C Alquist Conversational Dataset examples **59**

Chapter 1

Introduction

Personal voice assistants and text chatbots are newly emerging types of user interface. Their increasing popularity drives the need for better dialogue managers. This need will be accelerated even more by the industry automatization, in which voice-based user interface backed by conversational artificial intelligence will play the important role.

Dialogue management is an essential task in the domain of conversational artificial intelligence. The dialogue manager is the main part of the dialogue system, which communicates with users in natural human language. The main goal of the dialogue manager is to select the most appropriate response. The decision of dialogue manager is conditioned on the message received from the conversational partner and the state of the dialogue.

In this thesis, we will explore the theoretical background of dialogue managers, and we will describe the recent approaches of dialogue management. We will use three datasets for measuring the performance of dialogue managers. The first two will be dialogue bAbI Tasks 5 and 6. The third dataset will be Alquist conversational dataset, which consists of dialogues collected during one year from interactions between users and socialbot Alquist.

We will propose new architectures of dialogue managers inspired by the Hybrid code networks[1], using word2vec and fastText word embedding vectors, and convolutional and recurrent neural networks as inputs to the dialogue system. We will use hyperparameter optimization to find the optimal set of hyperparameters and demonstrate, that hyperparameter optimization can lead to improved performance. We will measure the turn and dialogue accuracy on the dialogue bAbI Tasks 5 and 6 and the Alquist conversational dataset. Afterward, we will compare the results with our baseline of Hybrid code networks.

Motivation

My motivation to work on this thesis is socialbot Alquist, which was the second place winner of Alexa Prize 2017 competition and is competing in the Alexa Prize 2018 again. The goal of the competition is to create conversational agent able to hold a coherent and engaging conversation with the user for 20 minutes about popular topics [2].

We face the problem of lack of appropriate learning data for the domain of popular topics such as movies, sports, news, books or fashion in this competition. We used rule-based dialog manager for Alexa Prize 2017, but it became evident that this approach is not sustainable for growing requirements of users on the quality and depth of conversations. It leads us to use machine learning-based dialogue manager in Alexa Prize 2018 which uses data collected from the previous rule-based system.

Thesis structure

The first part of this thesis (chapters 2 and 3) focuses on the theory behind dialogue managers. The second part (chapter 4) describes the implemented architectures of dialogue managers. The third part (chapter 5) describes used datasets. The fourth part (chapter 6) explains the experiments and presents the results. The fifth part (chapter 7) evaluates the results of the thesis.

Chapter 2

Theoretical Background

2.1 Dialogue manager

Dialogue manager is the crucial part of the dialogue system. The task of dialogue manager is to decide the next action of dialogue system according to the current context[3]. The context can include the current and previous messages, recognized entities, detected intent or sentiment of the message.

We can divide the dialogue managers according to how they produce the response to Retrieval-based dialogue managers and Generative-based dialogue managers[4]. We can also divide them according to their system to decide the next action to Rule-based dialogue managers, End-to-end dialogue manages and Hybrid dialogue managers.

2.1.1 Context

The context is the important part of the dialogue managers. The dialogue manager can be stateful thanks to the context. It is required because human conversation relies on its context[3]. The context contain all the messages and responses of the dialogue, recognized entities, intents of all previous messages or results of the API calls. It can also contain information about the user, like his preferences, location or age.

2.1.2 Retrieval-based dialogue managers

The retrieval-based dialogue managers use the set of predefined responses. They use rules or ranking mechanisms to select the next response[3]. These systems are not able to generate novel responses. Thanks to the set of predefined responses, the system designer has complete control over the result. The retrieval-based dialogue manager cannot in theory produce grammatically incorrect responses.

2.1.3 Generative-based dialogue managers

The generative-based dialogue managers do not rely on the set of predefined responses. They instead generate the whole response[5]. It allows them to produce novel responses. These dialogue manager usually relies on machine learning, mainly on the models similar to models of machine translation. These models are usually hard to train, and they require a large amount of data. The lack of control over generated responses can be problematic, as these models tend to produce grammatically incorrect, short and generic responses[4].

2.1.4 Rule-based dialogue managers

The core of rule-based dialogue managers is the set of rules producing the response based on the state of the dialogue. Systems of this type are more easily tunable. They can also be very robust in the narrow domains. However, it is very complicated to find the right set of rules for more complicated domains[1]. Notable rule-based dialogue manager is ELIZA[6].

2.1.5 End-to-end dialogue managers

The end-to-end dialogue managers use the machine learning to learn the policy to select or generate the response. The main advantage is that these dialogue manager can be learned without any domain-specific knowledge[7]. The disadvantage is that this method requires many training data. The generic, short and non-diverse responses like "I don't know" or "That is interesting" learned by these systems can also be problematic[8]. External knowledge is hard to insert into them too. These reasons make end-to-end dialogue systems more suited for non-goal oriented tasks than for goal-oriented tasks[9].

2.1.6 Hybrid-based dialogue managers

The hybrid-based dialogue managers combine the rule-based dialogue managers and end-to-end dialogue managers. It applies rules to the input and output processing[1]. It can be masking of the entities or transformation of context for the input, and making API calls or retrieving information from knowledge bases and inserting their results into the response for the output. The dialogue policy is learned from the data. These dialogue managers allow external knowledge to be inserted in the dialogue, and they can be more robust without the need to create significant sets of rules[1].

2.2 Word embeddings

The word embeddings is a technique of NLP, in which words or phrases from the input dictionary are mapped to a vector of real numbers. The words are mapped from space with one dimension per word into space with lower dimension. The advantage of the word embeddings is the fact that it removes the sparsity of input data. The word embeddings also allow the words with similar meaning to have a similar vector representation. This representation is useful for capturing syntactic and semantic regularities. Relation-specific vector offset characterizes each relationship. It allows us to perform vector-oriented reasoning based on the offsets between words [10]. The example of relationships between word vectors is demonstrated in the figure 2.11. The notable word embeddings are the Word2Vec[11], GloVe[12] and fastText[13].

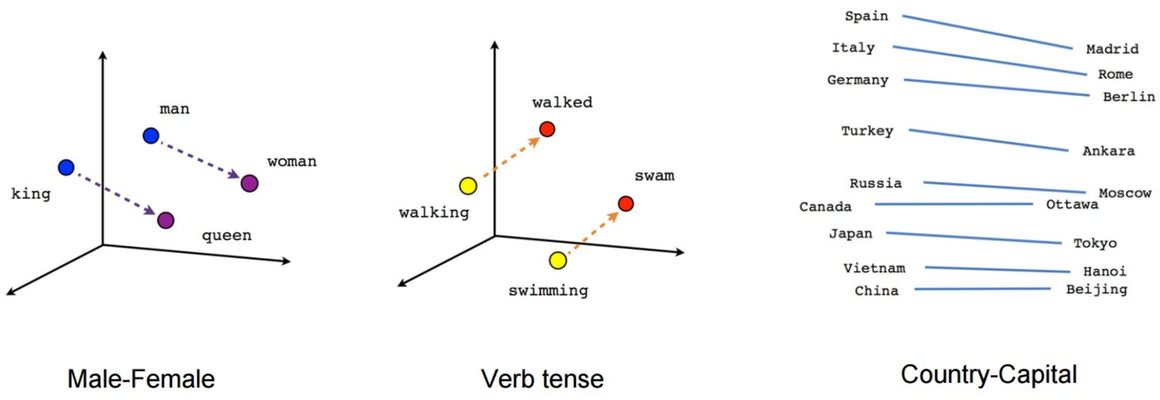

Figure 2.1: Relationships between word embedding vectors [14]

2.2.1 Word2vec

Word2vec is a group of models used to create word embedding vectors which were invented by Tomas Mikolov. Two models of Word2vec are continuous bag-of-words and skip-gram.

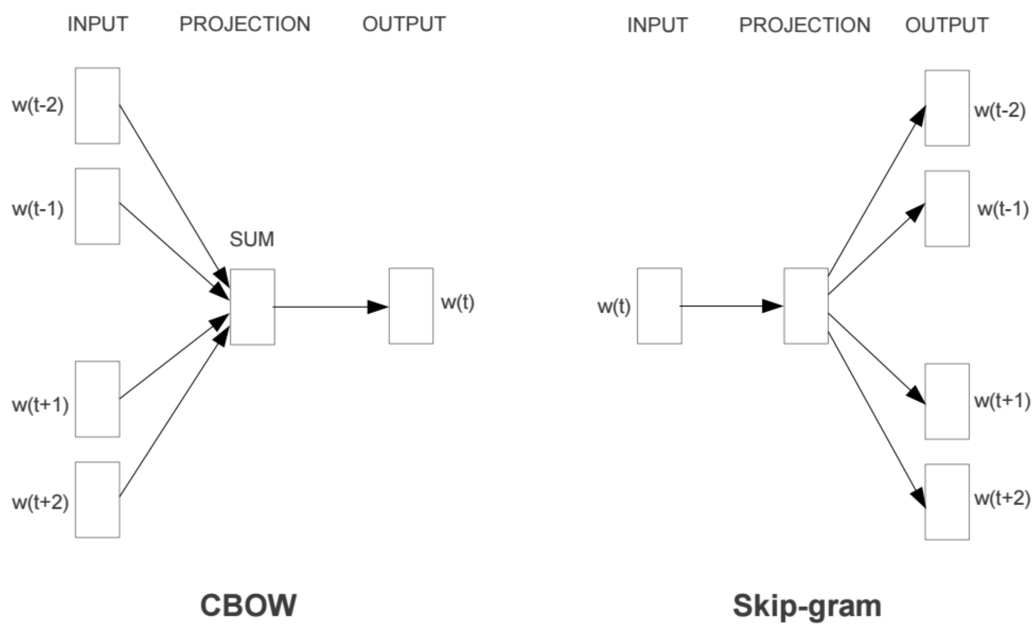

Figure 2.2: Continuous bag-of-words and skip-gram models [15]

The task of the continuous bag-of-words model is to predict the word out of context of surrounding words. The order of context words does not play any role in this model. The skip-gram model predicts the surrounding context words based on the given the word. Closer context words are weighted more than distant context words[11]. Both models achieve comparable results. Continuous bag-of-words model is faster and more suitable for larger datasets. Skip-gram model performs better in case of infrequent words and is suitable for small datasets[16]. The models can be trained using hierarchical softmax or negative sampling[11].

2.2.2 GloVe

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 2.3: Example of GloVe word-word co-occurrence matrix [12]

GloVe stands for Global Vectors for Word Representation, and it is a model developed by Jeffrey Pennington, Richard Socher and Christopher D. Manning from Stanford University. The GloVe model is trained on the non-zero elements of word-word co-occurrence matrix. This matrix tabulates, how frequently words co-occur with one another in the corpus. GloVe is a log-bilinear model with a weighted least-squares objective. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words probability of co-occurrence[12].

2.2.3 fastText

The fastText model is based on the skip-gram model, where each word is represented as a bag of character n-grams. A vector representation is associated with each character n-gram and words are being represented as the sum of these representations. By using only a distinct vector representation for each word, the skip-gram model ignores the internal structure of words, but fastText is suitable for morphologically rich languages thanks to the utilization of character n-grams [13]. For n=3, the word "where" is represented by

$$<wh, whe, her, ere, re> \quad \text{and} \quad <where>$$

for example. We use all n-grams larger or equal to 3 and smaller or equal to 6 in practice. FastText model is also capable of building word vectors for words that do not appear in the training set. For such words, we average the vector representation of its n-grams [13]. FastText also significantly drops the time of training thanks to a hierarchical softmax based on the Huffman coding tree [17]. The memory requirements of fastText embedding vectors can be reduced by quantization and by pruning the vocabulary without significant loss of the performance [18].

2.3 Neural networks

Neural networks are a class of machine learning algorithms taking inspiration from biological brain. It learns to solve the task using the examples without any prior knowledge. It consists of a large number of interconnected elements called neurons. The single neuron is equivalent to a perceptron[19].

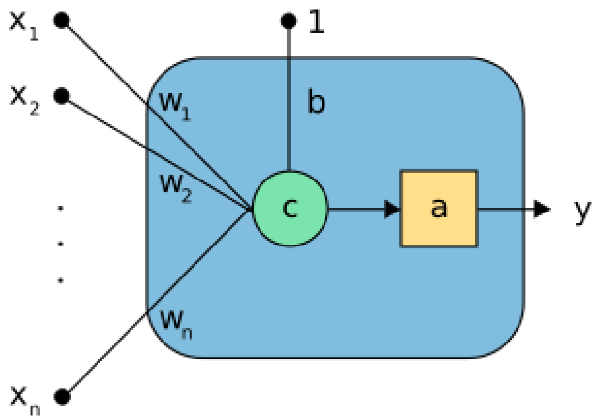

Figure 2.4: Perceptron [20]

$$v = \sum_{i=0}^{m} x^i w^i + b$$

Neurons are usually grouped into layers. The layers of neural networks form acyclic graph, in which information travels in one way only. There is also a modification of neural network, in which the output information can be fed to the input again. This modification is called Recurrent neural networks. The processing of input by layer can be represented as a single matrix multiplication between input vector x_k and weight matrix W, and the sum of bias vector b.

$$x_{k+1} = x_k W + b$$

The activation function is applied to the output of each layer of the neural network. There are several possible activation functions to use. We will list examples.

Step activation function

The step activation function was used in the early perceptron[19]. However, it is not useful for gradient-based learning methods such as backpropagation due to its non-continuation.

$$f(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

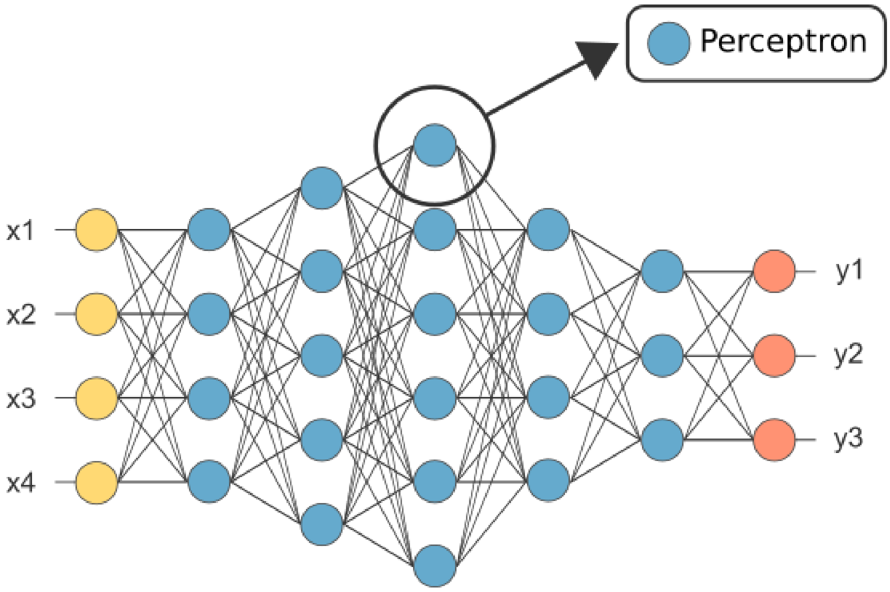

Figure 2.5: Deep neural network [20]

Logistic sigmoid activation function

Logistic sigmoid activation function is a function which is differentiable. This attribute makes it useful for gradient-based learning methods. However, it suffers from vanishing gradient, where the gradient of the function drops too close to zero for the extreme input values [21].

$$f(x) = \frac{1}{1 + e^{-x}}$$

Tanh activation function

Unlike the Logistic sigmoid activation function, the Tanh activation function is symmetric around the origin. Its been shown that this property leads to faster convergence of the training [22].

$$f(x) = tanh(x)$$

ReLU activation function

The Rectified Linear Unit[23] addresses the problem of the vanishing gradient because, for the inputs bigger than 0, the derivative of the ReLU activation function equals 1. The problem is the value of derivative for the inputs smaller than 0, which equals to zero. This can lead to dying ReLU problem. The ReLU neurons can be pushed into states, in which they are inactive for any possible input. The gradient does not flow in such state, and the neuron is stuck in the inactive state. It is another form of vanishing gradient problem. Various modifications of ReLU, such as Leaky ReLU[24], PReLU[25] or ELU[26] try to solve this problem.

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

Softmax function

The Softmax function is function applied to the output layer. It transforms the vector of arbitrary real values into a vector of a probability distribution. Softmax is commonly used for multiclass classification[27].

$$f(x)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for j = 1...K}$$

2.3.1 Training

The goal of neural network's training process is to find the parameters of the neural network, which minimize the value of loss function. Two examples of loss functions are mean squared error and categorical cross entropy. The mean squared error is defined as

$$MSE = \frac{1}{n} \sum_{n}^{i=1} (Y_i - \widehat{Y}_i)^2$$

where Y_i is the correct value and the $\widehat{Y_i}$ is prediction.

The categorical cross entropy is defined as

$$CE = -\sum_{i=1}^{n} Y_i log \widehat{Y_i}$$

where Y_i is the correct value and the $\widehat{Y_i}$ is prediction.

The most used technique to find the parameters minimizing the value of the loss function is the gradient descent using backpropagation algorithm. The gradient descent is a method to find the local minimum of the function. The finding takes place during multiple iterations. The method calculates the derivation of the function with respect to the parameters. It uses the derivation to update the weights accordingly.

$$w_{i+1} = w_i + \lambda_i \nabla f_{(w_i)}$$

There is also approximation variant of gradient descent called stochastic gradient descent. Stochastic gradient descent is useful when we work with large datasets. Instead of optimizing using the whole dataset at once, we randomly select one example or a small batch of them.

The backpropagation algorithm[28] applies the chain rules of derivatives of composed functions, representing each layer, to update the weights. The forward pass through the neural network is performed at first. We compute the loss given the output of neural network and the ground truth. The last step is to update the weights of layers from output to the input of the neural network.

2.3.2 Overfitting

Overfitting is a common problem of the neural networks. It is caused by their expressive power. The neural network instead of learning the generalization of the given problem learns to remember correct output for each training example. One of the ways how to detect the overfitting is to use the validation dataset and measure the classification error and loss there.

There are several possibilities how to deal with overfitting. We can use early stopping, regularization methods or dropout. When we use the early stopping, we stop the training process if the error of the validation dataset stops to improve.

The most used regularization method is L2 regularization. It adds an extra term to the loss function, which is the sum of the squares of all the weights in the network multiplied by regularization parameter divided by the size of training parameter. This extra term forces the neural network to prefer smaller values of parameters. This fact is helpful because network learns to use all available features and it stops to rely on the single one, which leads to better generalization[29].

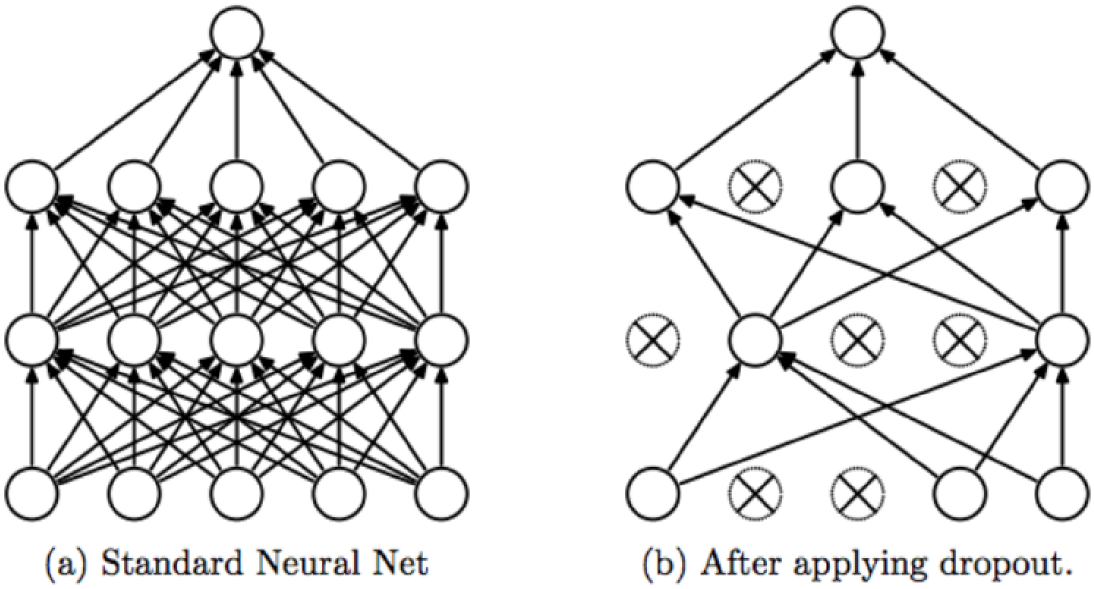

Figure 2.6: Dropout [30]

Dropout[30] is a method of preventing overfitting, in which we keep the neuron turned on with some predefined probability. It forces the neural network to not rely on the output of a single neuron, thus making it more robust. The intuition behind the dropout is that we average several networks trained in the ensemble.

2.3.3 Hyperparameters

Hyperparameters are the parameters of the machine learning algorithm, which have to be set before the training process. The hyperparameters have an impact on the results of the training. We use the validation set to find set of hyperparameters which achieve the lowest classification error[31]. For neural networks, these hyperparameters can include learning rate, number of layers, number of neurons in a layer, activation functions, keep probability of dropout or regularization parameter. There are several ways of finding the best hyperparameters.

Manual search

This approach allows us to use our prior knowledge of the problem. However, we have to guess the correct set of hyperparameters, observe the results of the training and manually make a new guess of better hyperparameters given previous results.

Grid search

Grid search is exhaustive search thought manually selected subset of the space of hyperparameters. The machine learning is performed on all cartesian products of hyperparameters and set performing the best on the validation set is selected. The grid search suffers from the curse of the dimensionality, but it can be paralyzed thanks to independence between runs of the algorithms[32].

Random search

Random search replaces the exhaustive search of Grid search by testing randomly selected samples. Random search is better suited to domains, in which only part of the hyperparameters has the impact on the result of the learning process[32]. Random search is easily parallelizable and we can input the prior knowledge by specifying the distribution, from which we will sample the hyperparameters.

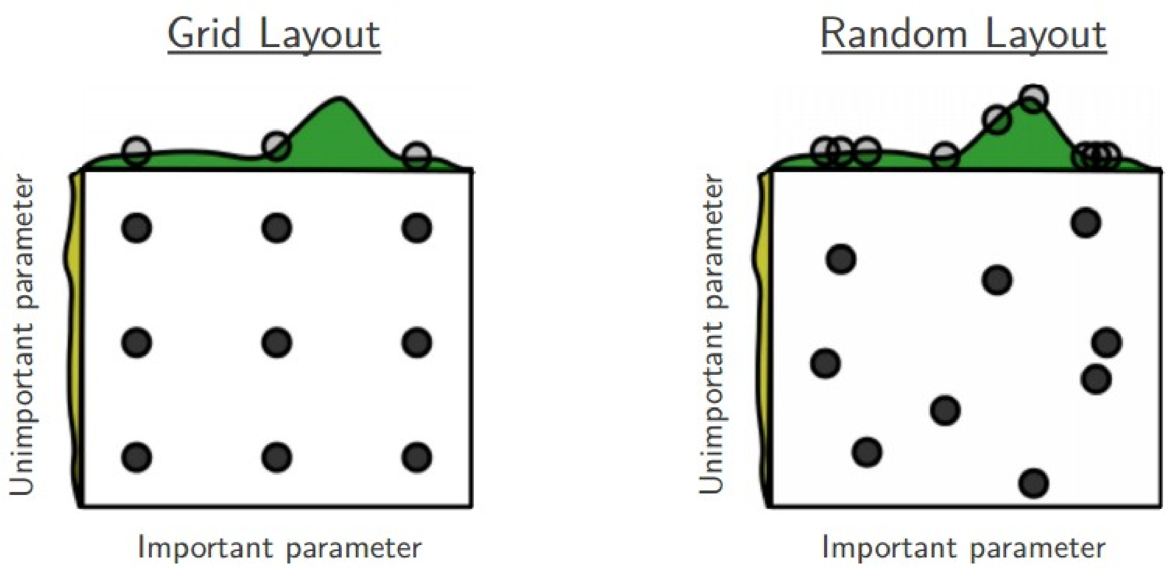

Figure 2.7: Grid and random hyperparameter search [32]

Bayesian optimization

Bayesian optimization method for hyperparameter search builds a probabilistic model of function mapping the hyperparameters to the loss function of machine learning algorithm. The method iteratively evaluates the sets of hyperparameters and updates its probabilistic model. It aims to gather observations giving the most information about the optimization function and location of its minimum. It has been shown that Bayesian optimization achieves better results in fewer number of iterations then Grid and Random search thanks to its ability to reason about the results of the training run before it is executed[33][34].

Gradient-based optimization

For the machine learning algorithms, for which we can compute the gradient with respect to the hyperparameters, we can find the best hyperparameters using the gradient descent[35].

Evolutionary optimization

Evolutionary optimization is process for finding the hyperparameters inspired by the biological concept of evolution. We sample the initial population, evaluate the fitness of the individuals, we remove the worst individuals from the population, we modify the individuals by operations of mutation and crossover and repeat this process from the second step until we find a satisfactory result or the performance is no longer improving. Evolutionary optimization can be used to find hyperparameters[33], architecture of the network[36][37] or even to find the weights of the network[38].

2.4 Recurrent neural networks

The conventional neural networks can work only with the inputs of fixed length for which they produce fixed size outputs. They also work with presumption, that inputs are mutually independent [39]. The recurrent neural network is a modification of neural networks, which can work with arbitrary long sequences of inputs and the previous inputs [40] can influence following inputs.

Recurrent neural network (RNN) produces output and state vector based on the combination of input and its previous state vector. The state vector works as network's memory, which depends on the all previous inputs. The network can remember previous computations thanks to this memory. Thus it can work with mutually dependent inputs of arbitrarily lengths[40].

The most basic Recurrent neural network can be expressed as:

$$s_t = f(x_t U + s_{t-1} V)$$

$$y_t = g(s_t W)$$

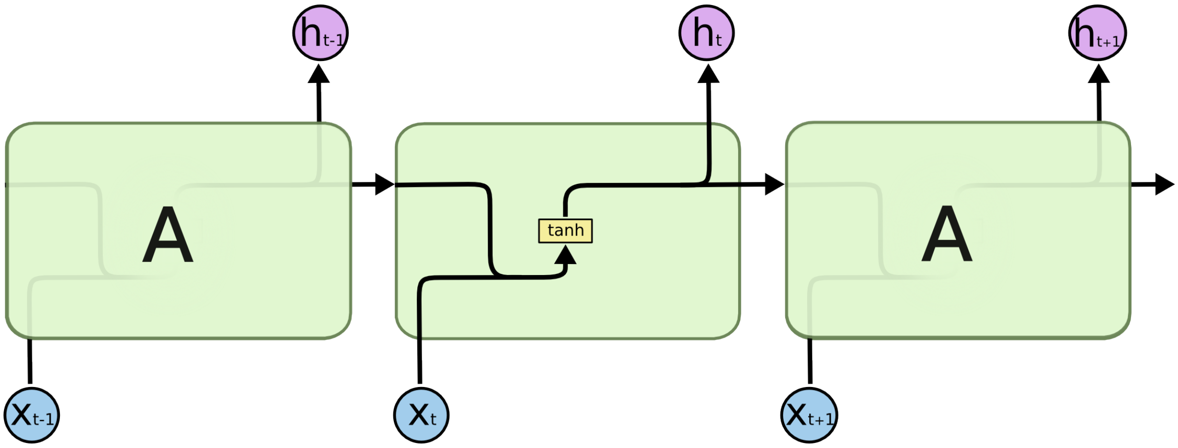

Figure 2.8: Modules of recurrent neural network [41]

where s_t, x_t and y_t are state, input and output vector at time t; U, V and W are weight matrices; and f and g are activation functions.

RNN should be able to use a memory of arbitrarily long input sequences. However, they struggle to work with long sequences in practice [42]. Long Short Term Memory (LSTM) is a modification of RNN which addresses this problem.

2.4.1 Long Short Term Memory

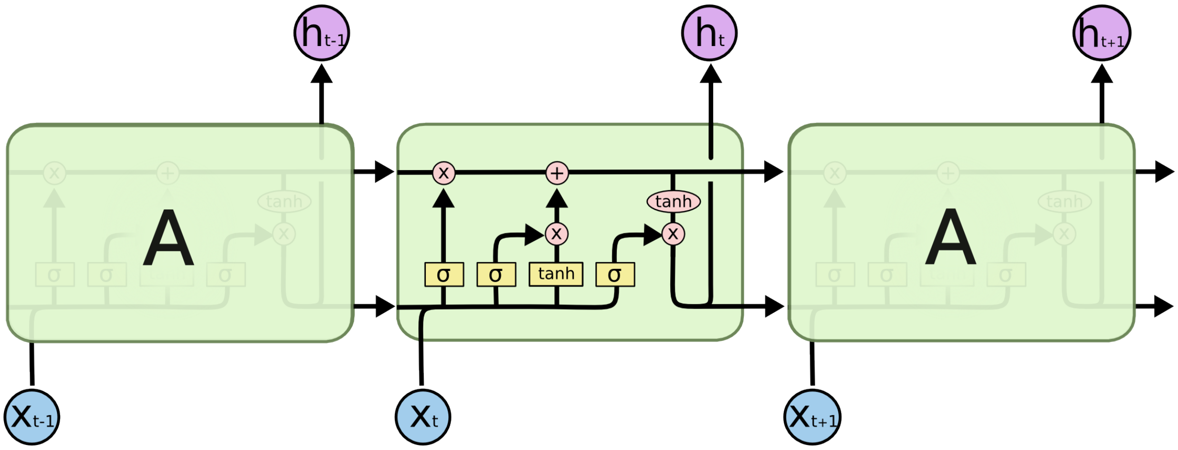

Figure 2.9: Modules of Long Short Term Memory [41]

The LSTM [43] is version of RNN, which is designed to remember information over a big number of steps. The reason for this is the introduction of the cell state, which flows through all steps of computation. Each LSTM cell can work with the content of cell state through forget gate, input gate, and the output gate.

The LSTM can be expressed as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

where x_t is input at state t, C_{t-1} and h_{t-1} are cell state and output of state $t-1$ and $*$ is elementwise multiplication.

There is also RNN cell called Gated recurrent unit[44] (GRU), which tries to solve the same problems as LSTM with smaller number of parameters. It has been demonstrated, that performance of GRU and LSTM is similar, only GRU achieving better performance on smaller dataset[45].

2.5 Convolutional neural networks

Regular neural networks don't scale well to big inputs as images for example. The number of parameters grows quickly with the size of input. Also the full connectivity of the neurons is not necessary and may lead to overfitting[40].

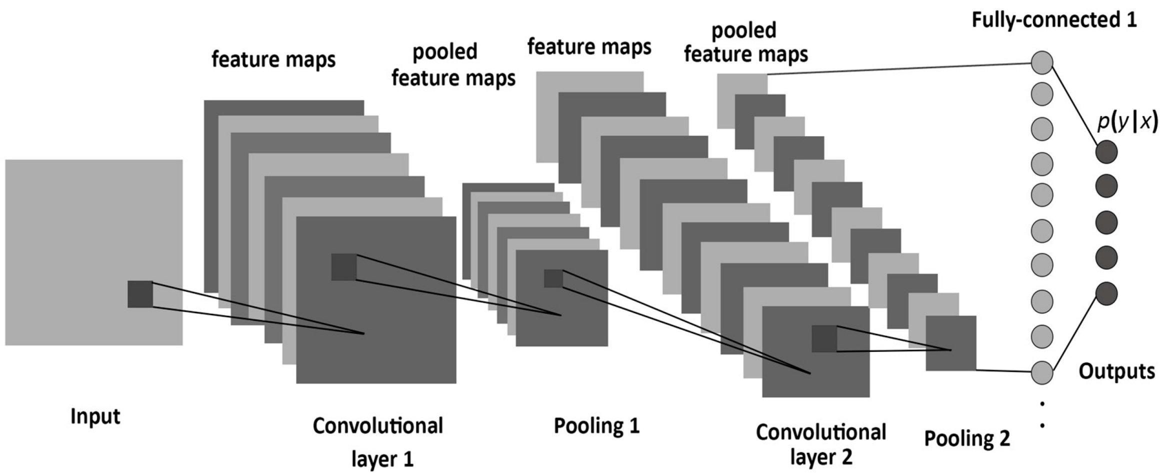

Figure 2.10: Convolutional neural network [46]

The convolutional layer consists of a set of learnable filters. These filters are spatially small, but extends through full depth of input. We slide each filter across the width and height of the input and we compute the dot product between entries of the filter and input at any position. This will create a new activation map. The parameters of learnable filters are shared across all possible spatial positions.

The convolutional layer is usually followed by pooling layer. The pooling layer is a form of non-linear down-sampling. There are several possible variants of pooling layers like average pooling, L2-norm pooling or the most common max pooling[47]. Max pooling divides the input into non-overlapping rectangles. It forms new feature map formed from the maximum of each rectangle. The motivation behind max pooling is, that the exact location of the feature is not important. Important is its rough location against other features. It has been demonstrated, that max pooling performs better than average pooling in practise[48]. The main advantage of the pooling operation is the fact, that it reduces the spatial size of processed activation map and thus reduce the number of parameters and necessary computations.

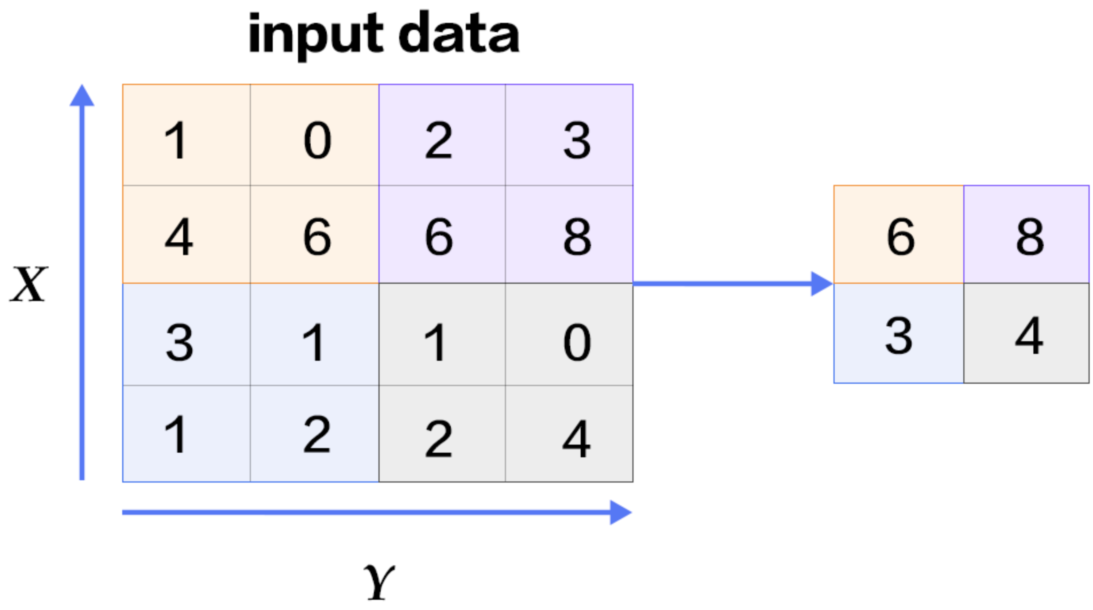

Figure 2.11: Max pooling layer [49]

The convolutional neural networks are typically applied to analyze visual imagery. They have been successfully applied to image classification[50][51][52][53][54], face recognition[55][56], scene labeling[57][58], human pose estimation[59][60] or document analysis[61][62]. However they can be also applied in non-visual domains like speech recognition[63][64] or text classification[65][66].

Chapter 3

Related work

We will review the possible types of retrieval and generative dialogue managers, including rule-based dialogue managers, End-to-end dialogue managers and hybrid-based dialogue managers.

3.1 Rule-based systems

Rule-based systems[9] work with the hand-coded rules. The hand-coded rules decide the appropriate answer by detecting patterns in the user's messages and deciding based on the presence of patterns in the message and actual state of the dialog. The rule-based systems work well in the constrained domains like form-filling dialogues. However, rule-based systems become less straightforward and not so accurate with growing domain and real user interactions.

3.2 TF-IDF match

The method of TF-IDF match[9] uses matching score between the input and response, or between input and message from the training set to select the right response. We use TF-IDF weighted cosine similarity between bag-of-words vectors. If we match the input and responses, then the response with the maximal score is selected. If we match the input and message from the training set than the response following the message from the training set with the maximal score is selected.

3.3 Semantic similarity

The method of semantic similarity works similar to TF-IDF match. The difference is, that we match average of word embedding vectors (Word2vec[11], GloVe[12], fastText[13]) by cosine similarity. We can use pretrained word embedding vectors, or we can train the word embedding vectors using our training set.

3.4 Supervised models

Supervised models[9][67] predicts the next response given the previous conversation. The candidate response y is scored against the input x in the following way:

$$f(x, y) = (Ax)^T By$$

where A and B are $d \times V$ word embedding matrices. Input and output are treated as summed bag-of-embeddings. The embedding are trained with margin ranking loss l:

$$l = \sum_{y \neq \overline{y}}^{N} max(0, f(x, \overline{y}) - f(x, y) + m)$$

where m is the margin. We sample N negative responses $\overline{y}$ per example and train with SGD. We can think about this method as information retrieval model with learned matching function.

3.5 Seq2Seq

Seq2Seq[68] is a class of models able to learn the mapping between input and output sequence of arbitrary lengths. The model consists of encoder Long Short-Term Memory (LSTM)[43] and decoder LSTM. The encoder LSTM maps input sequence into a large fixed-dimensional vector representation, and encoder LSTM converts the output sequence from the vector representation.

The goal of the model is to estimate the conditional probability $p(y_1, ..., y_{T'}|x_1, ...x_T)$, where $x_1, ...x_T$ is an input sequence and $y_1, ..., y_{T'}$ is an corresponding output sequence. Sequence lengths T and T' may differ. The model computes the conditional probability by obtaining the fixed-dimensional representation v of the input sequence $x_1, ...x_T$, given by the last hidden state of encoder LSTM. Model than computes the probability of $y_1, ..., y_{T'}$ by decoder LSTM whose initial hidden state is set to v:

$$p(y_1, ..., y_{T'}|x_1, ...x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1})$$

Each $p(y_1|v, y_1, ...y_{t-1}$ distribution is represented with a softmax over all the words in the vocabulary. We require that each sequence ends with a special symbol “<EOS>” representing end of sequence in practise. Model also achieves better results, if we reverse the order of input sequence. Model is asked to map c, b, a to a', b', c', d' instead of a, b, c to a', b', c', d'. This way it is easier for Stochastic gradient descent to “establish communication” between input and output, because a is in close proximity to a' and b is fairly close to b' and so on.

The seq2seq models were initially used for machine translation. However, we can use them as generative dialogue managers[7]. The seq2seq model learns to produce response conditioned on the previous message. This model can generate simple and basic conversations.

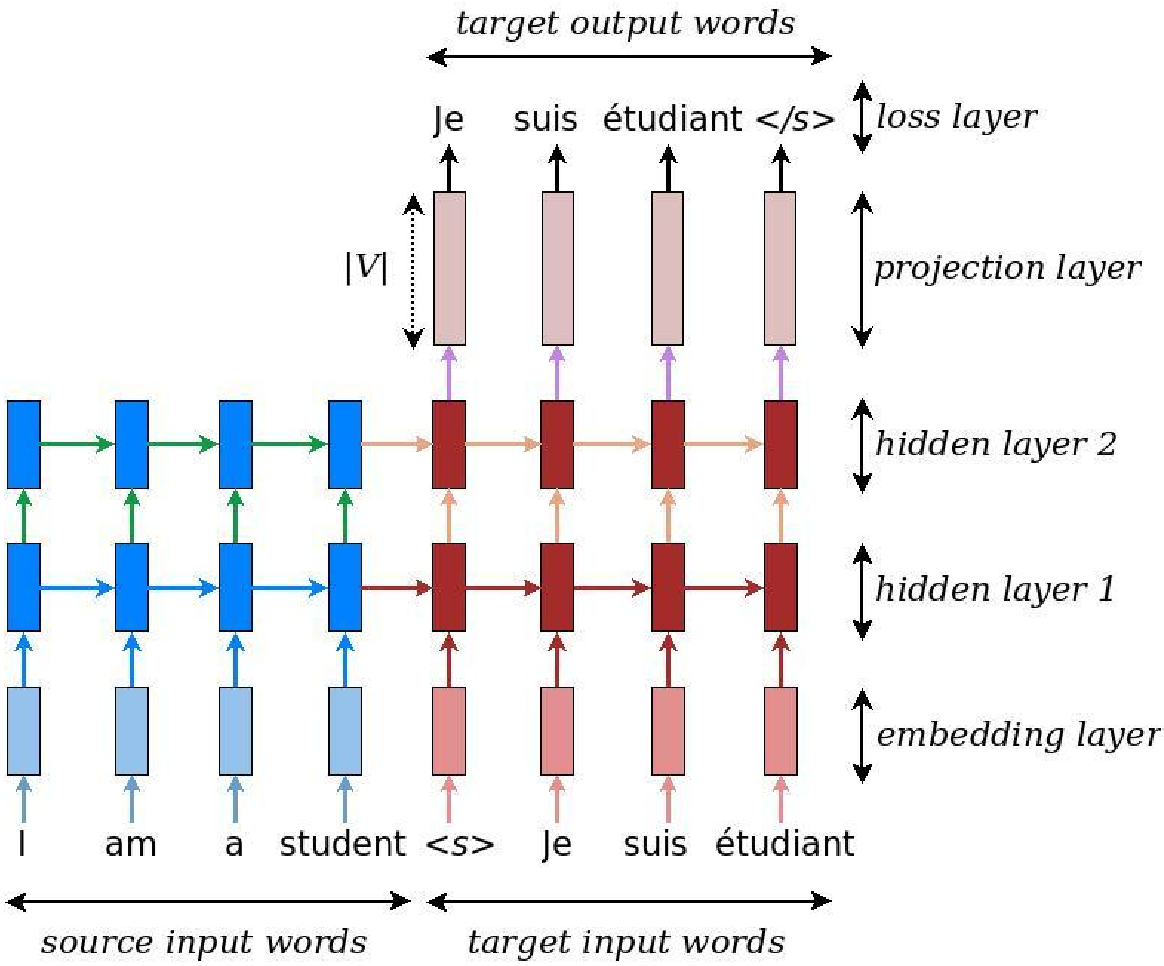

Figure 3.1: Seq2Seq [69]

3.6 Memory networks

Memory networks[70] are class of models, combining inference components with the long-term memory component. The memory network learns how to use these two components jointly. The memory can be read and written to, and the model is taught how to operate it efficiently and how to use it for inference. A memory network consists of four components I, G, O and R:

- I (input feature map) component converts the incoming input into internal feature representation.
- G (generalization) component updates old records stored in the memory given the new input. We call this component Generalization component because it can be used by the network to compress and generalize its records for later use.
- O (output feature map) produces a new output given the input and saved records in the memory. The new output is in the feature representation space.

- R (response) converts the output from feature representation space into desired response format. This format can be textual representation for example.

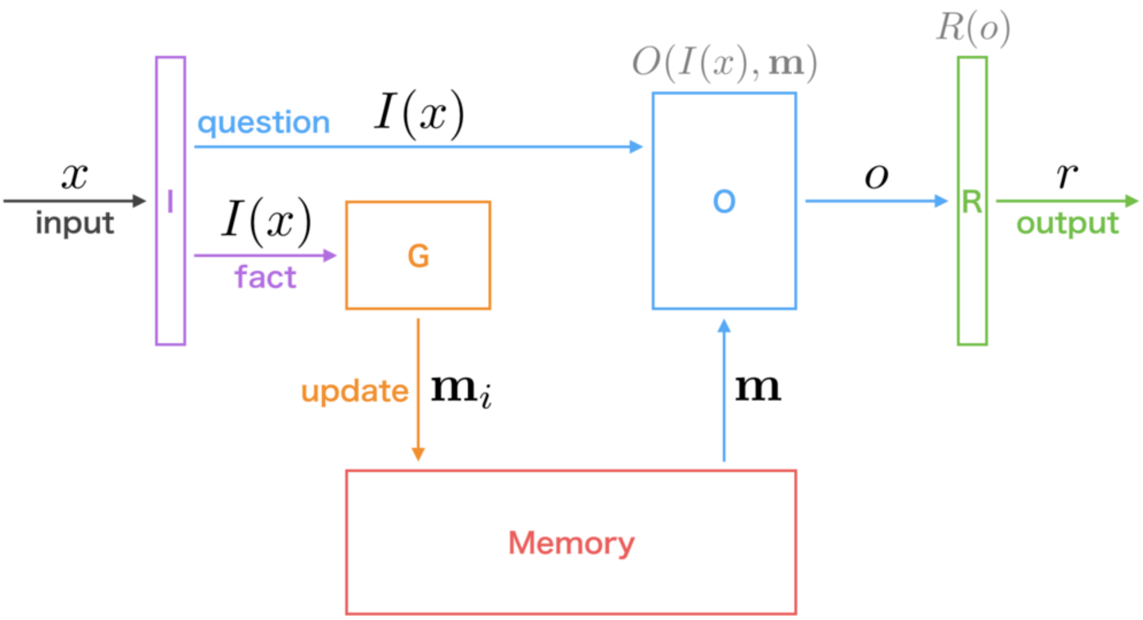

Figure 3.2: Memory network [71]

Given the input x the flow of the model is as follows:

1. Component I (input feature map) coverts the input x into internal feature representation $I(x)$.

2. The memories m_i are updated by component G (generalization) given the new input converted into internal feature representation $I(x)$:

$$m_i = G(m_i, I(x), m), \forall i$$

3. Component O (output feature map) computes output features o given the new input converted into internal feature representation and the updated memory:

$$o = O(I(x), m)$$

4. Component R (response) decodes output features o into final response:

$$r = R(o)$$

Components can be in any form, like rule-based or it can use any existing machine learning algorithm like SVM or decision tree.

Interesting implementation of memory networks is End-To-End Memory Networks[72]. This implementation can be trained end-to-end, and hence requires less supervision during training. It makes them easier to use in the real world scenarios.

The End-To-End Memory Networks has the following flow:

1. We are given an input set $x_1, ..., x_i$ to be stored in the memory. The set is converted into memory vectors m_i by embedding matrix A and stored in the memory.

2. The query q is embedded by embedding matrix B with the same dimensions as embedding matrix A. Embedded query q creates an internal state u.

3. We compute the match between internal state u and memory m by:

$$p = Softmax(u^T m)$$

   Vector p forms probability vector over the inputs.

4. Each input x_i has its output vector c_i. We create vector c_i by embedding it by embedding matrix C. The response vector o from the memory is a sum of transformed inputs c_i weighted by the probability vector p:

$$o = \sum_i p_i c_i$$

5. The sum of response vector o and internal state u is passed through matrix W and softmax to produce the probabilities of response labels:

$$\hat{a} = Softmax(W(o + u))$$

We can train the whole model by modifying matrices A, B, C and W by back-propagation thanks to the fact, that function from input to output is smooth. We teach the model by minimizing a standard cross-entropy loss between predicted response $\hat{a}$ and true label a. Training is performed using stochastic gradient descent.

There is also multiple layers variant of End-To-End Memory Networks. The changes are following:

- The input to layers above the first is the sum of o^k and u^k from layer k:

$$u^{k+1} = o^k + u^k$$

- Each layer has its own embedding matrices A^k and C^k.

- The output of network uses output of final layer $K + 1$:

$$\hat{a} = Softmax(W(u^{K+1})) = Softmax(W(o^k + u^k))$$

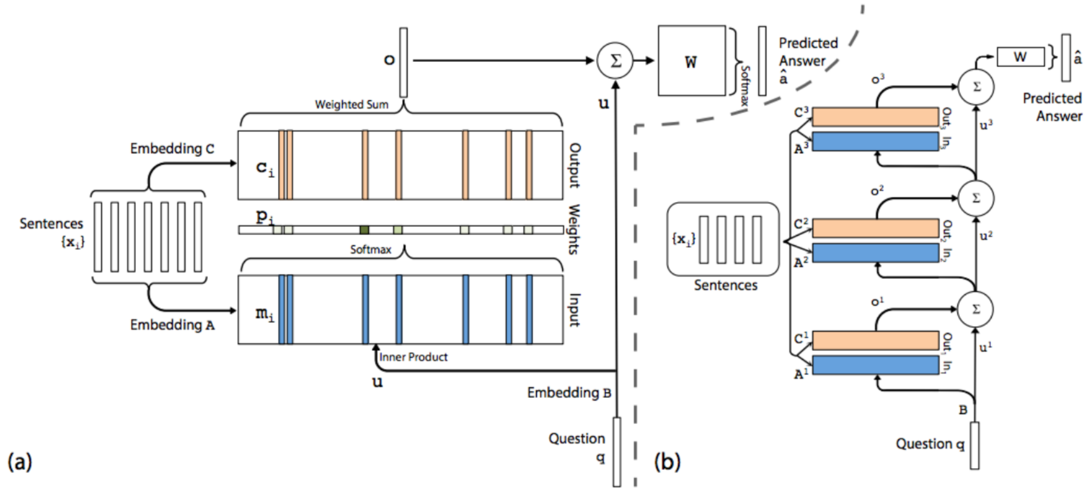

Figure 3.3: Single and three-layer end-to-end memory network [72]

3.7 Hybrid code networks

Hybrid code networks[1] is model created by Jason D. Williams, Kavosh Asadi and Geoffrey Zweig from Microsoft Research and Brown University. Hybrid code networks combine the recurrent neural network and domain-specific knowledge in the form of software and system action templates. The main benefit of this model is the ability to work with the considerably reduced amount of training examples.

The Hybrid code networks consists of four components:

- Recurrent neural network
- Domain-specific software
- Domain-specific action templates
- Entity extraction module

Recurrent neural network and developer code maintain state of the dialog. Domain-specific action can be a textual communicative action or call to API. Entity extraction module is a conventional entity extraction module.

The message send to Hybrid code network is processed in the following way:

1. We create the bag of words of the message.
2. We calculate the average of the world embedding vectors (Word2vec[11], GloVe[12], fastText[13]). We can use pretrained word embedding vectors, or we can train the word embedding vectors using our training set.
3. The entity extraction module finds the entities and masks them by their types in the message. For example, extraction module finds entity "Seattle" of type "city" in the message "I would like to know a weather forecast for Seattle." The message is masked to "I would like to know a weather forecast for <city>"

4. The founded entities are passed to the entity tracking code provided by the developer.

5. The code can also provide the action mask in the form of the bit mask. The action mask indicates actions which are permitted in the current timestamp. For example, the action to weather API is prohibited if the user has not yet provided the place and the date of the forecast.

6. The code can also provide other context features which will be helpful for distinguishing among actions, such as currently presented entities.

7. Bag of words, message embedding, and context features are concatenated and passed as an input to the recurrent layer, such as LSTM[43] or GRU[44]. Additional features of the recurrent layer can optionally be previously predicted action and result of the API call.

8. The recurrent neural network computes the hidden state, which is used in the next timestamp for the new message and passed to the dense layer with a softmax activation function. The dimension of dense layer is the same as the number of distinct system action templates. Thus the result of softmax is the distribution of action templates.

9. We apply the action mask to the result of softmax as an element-wise multiplication, and we normalize the result back to the probability distribution. The non-permitted actions obtain zero probability thanks to this operation.

10. We select the action template from the probability distribution.

11. The selected action is passed to developer code, which substitutes in entities. For example template "<city> right?" to "Seattle right?".

12. The next step depends on the type of action. If the action is API call, we call the corresponding API by invoking developer code. If the action is text, we render text to the user.

13. The cycle repeats for the next message.

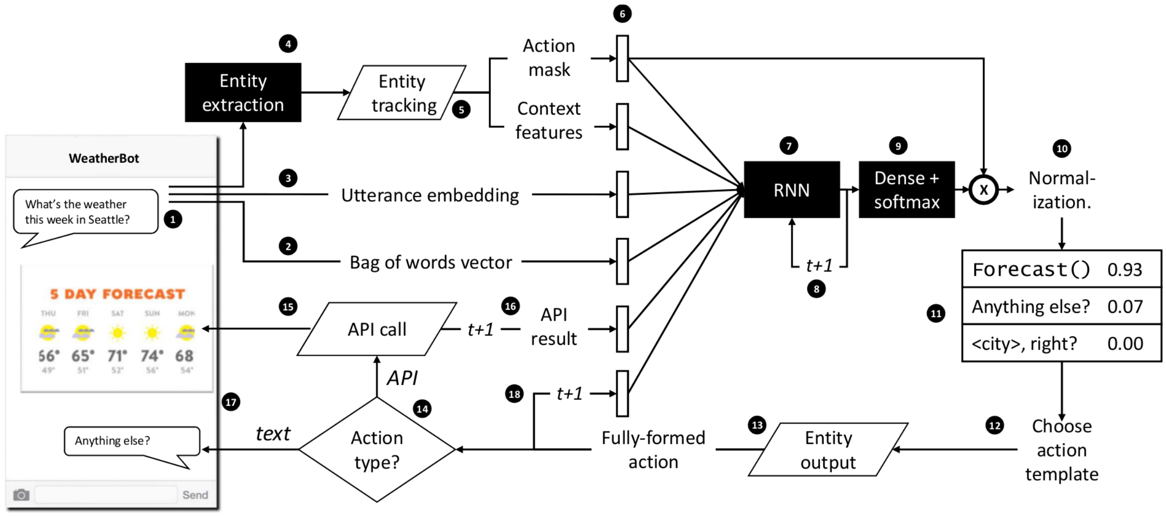

Figure 3.4: Hybrid code networks [1]

Chapter 4

Implementations

We perform the experiments using the following implemented architectures of neural networks. We derived the architectures from Hybrid code networks[1].

4.1 Hybrid code networks

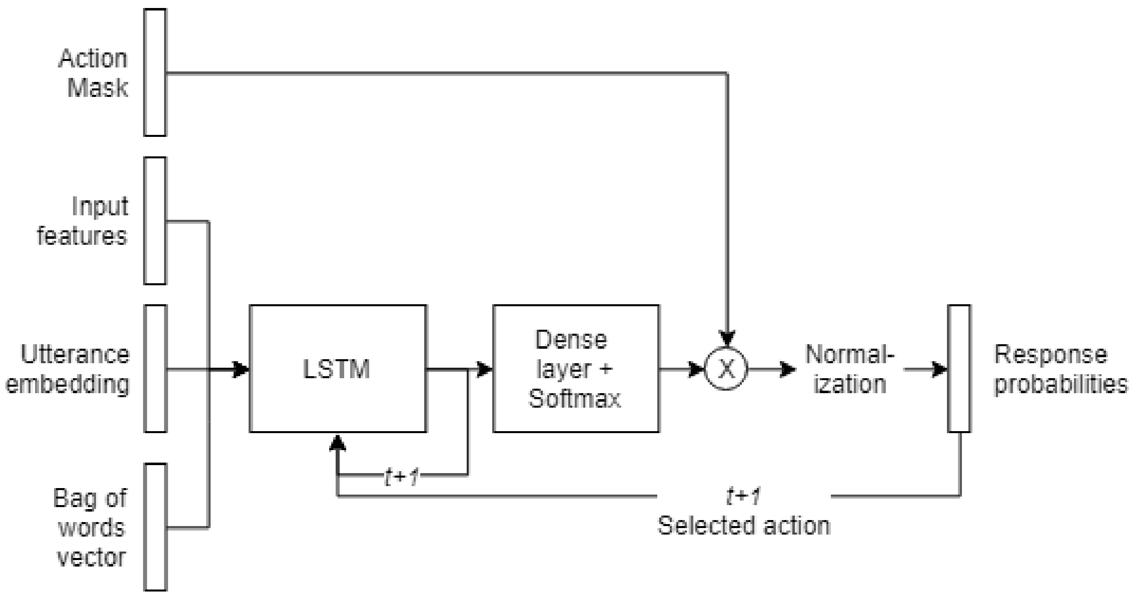

Figure 4.1: Hybrid code networks with bag-of-words and average of word embeddings input

The model is shown in Figure 4.1. The inputs to the model are Action mask, Input features, Utterance embedding and Bag of words vectors. We create Utterance embedding vector by averaging word vectors of the input message. The Input features vector can contain arbitrary features useful for the selecting the correct response. Utterance embedding, Bag of words and Input features vectors are passed to LSTM, dense layer and Softmax function. The inner state of LSTM is used for the next input message. The output of Softmax function is element-vise multiplied by the vector of Action mask. The resulting vector is normalized to the probability distribution. The normalized vector describes the probability of responses.

We select the response with the highest probability and use it as a feature for the next message.

4.2 Hybrid code networks with recurrent neural network

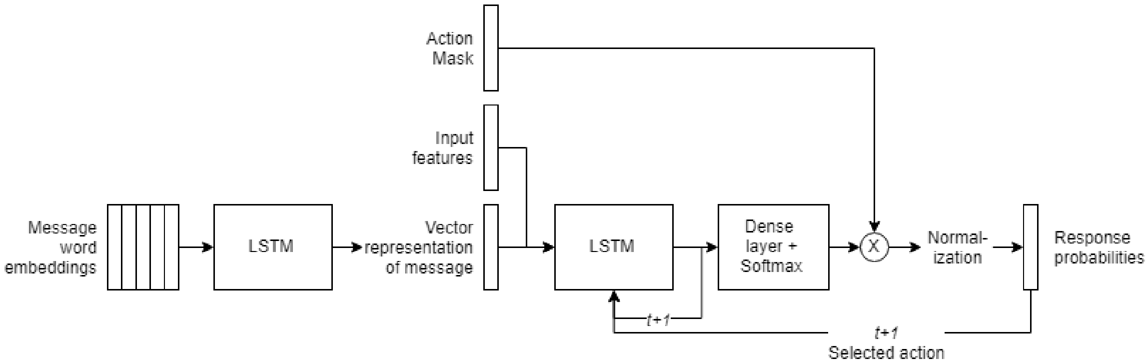

Figure 4.2: Hybrid code networks with recurrent input layer

The model is shown in Figure 4.2. The inputs to the model are Action mask, Input features, and Message word embeddings. The Message word embeddings are passed to LSTM layer, which creates the vector representation of the user's message. The representation of the user's message, together with Input features are passed to LSTM, dense layer and Softmax function. The inner state of LSTM is used for the next message. We element-vise multiply the output of Softmax function by the vector of Action mask and normalize the product to the probability distribution. We select the response with the highest probability and use it as the feature for the next message.

4.3 Hybrid code networks with convolutional neural network

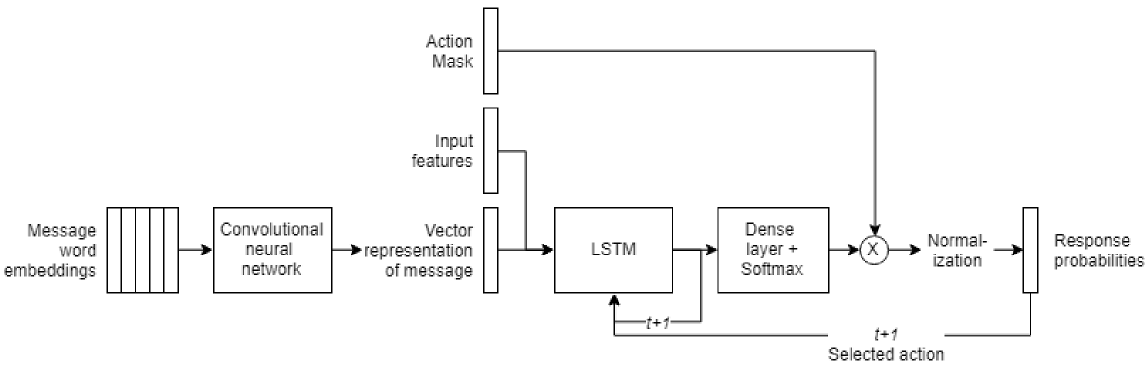

Figure 4.3: Hybrid code networks with convolutional input layer

The model is shown in Figure 4.3. The inputs to the model are Action Mask, Input Features, and Message word embeddings. The Message world embeddings are passed to the convolutional neural network, which creates the vector representation of the message.

The convolutional neural network is inspired by Convolutional Neural Networks for Sentence Classification[65] created by Yoon Kim from New York University. Schema of the model is shown in the Figure 4.4. The input to this model is message represented by word embeddings. The message is padded to the length of the longest messages. We apply the convolutional filters of size 2, 3, 4, and 5 to the message. We apply the max-pooling over vectors resulting from the convolutional filter and concatenate the results of max-pooling. The vector of concatenated results of max-pooling is vector representation of input message. We don't use the fully connected layer at the end of the model and multiple word embeddings as demonstrated in [65] in our model.

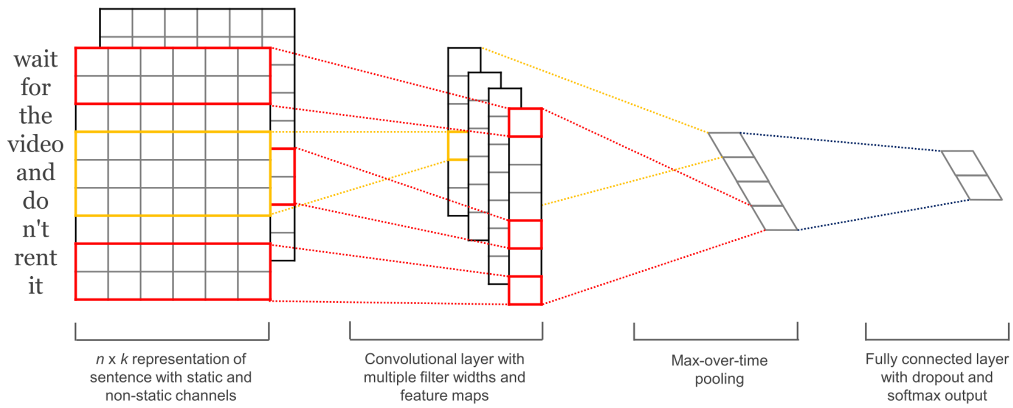

Figure 4.4: Convolutional neural networks for sentence classification [65]

The rest of the model remains the same as a model which we described in Section 4.2.

Chapter 5

Datasets

We perform the experiments on the Dialog bAbI Task Data and Alquist conversational dataset. Each of these datasets is described in the following section.

5.1 Dialog bAbI Tasks Data

The Dialog bAbI Task Data [9] is dataset created by Antoine Bordes, Y-Lan Boureau and Jason Weston from Facebook AI Research. The goal of this dataset is to propose an open resource to test end-to-end dialog systems in a way that favors reproducibility and comparisons, and is lightweight and easy to use. The dataset is divided into six tasks with increasing difficulty in the domain of restaurant reservation. The tasks are following: Issuing API calls, Updating API calls, Displaying options, Providing extra information, Conducting full dialogs and DSTC2. We don't use the Tasks 1 to 4, because task 5 combines them all. We describe the last two tasks (Task 5 and 6) in the following sections.

5.1.1 Dialog bAbI Task 5

The Dialog bAbI Task 5 consists of a combination of Dialog bAbI Tasks 1 to 4. The language used in this task is synthetically generated. There are 43 patterns for the user's messages and 20 for the bot responses. The dialogues are underlined by the knowledge base, which contains information about restaurants. Each restaurant is defined by a type of cuisine (10 types), location (10 types), price range (3 types), rating (1 to 8) and availability for party size (2, 4, 6 or 8 people). There is also phone number and address in the knowledge base for each restaurant. The dataset is divided into training, validation and two testing sets. Each set consists of 1,000 dialogues. The first testing set uses the vocabulary of the training set. The second testing set contains out-of-vocabulary words. The example of the dialog included in the dataset is in the Appendix A.

We mask the values in the messages and responses in the following way. We replace each value token (French) by its type (r_cuisine). We transform the input "I would like some French restaurant which is cheap" to "I would like some r_cuisine restaurant which is r_price"

Table 5.1: List of predicted actions from Dialog bAbI task 5

any preference on a type of cuisine
api_call r_cuisine r_location r_number r_price
great let me do the reservation
hello what can i help you with today
here it is r_address
here it is r_phone
how many people would be in your party
i'm on it
is there anything i can help you with
ok let me look into some options for you
sure is there anything else to update
sure let me find an other option for you
what do you think of this option: r_name
where should it be
which price range are looking for
you're welcome

for example. We perform the masking because entity recognition and extraction is not the main focus of this work. This also reduces the number of distinct inputs, which helps dialog manager to select the right response. We reduce the number of responses from 20 to 16 by masking.

5.1.2 Dialog bAbI Task 6

The Dialog bAbI Task 6 consists of real human-bot dialogues. It consists of data from DSTC2 [73], which was originally used to learn dialog state tracking. The creators of the Dialog bAbI Task 6 used dialog transcripts to create a user and bot utterances and to create underlining knowledge base. The dialogues are noisier and harder to learn due to speech recognition errors, and users don't have a deterministic behavior. The example of the dialog included in the dataset is in the Appendix B.

The dataset is split into 3,249 training dialogues, 403 validation dialogues, and 402 testing dialogues. The user can request only three fields: cuisine, location, and price. The values were masked as in the case of Dialog bAbI Task 5. The responses which occur only rarely was masked by 'UNK' action. This leads to 56 possible responses. Testing set contains actions, which are not present in the training set. Such actions were also masked by 'UNK' action. There are 40% of dialogues in the testing set, which contains 'UNK' action, which significantly limits the maximal possible dialogue accuracy of retrieval based dialogue managers.

Table 5.2: List of predicted actions from Dialog bAbI task 6

UNK
api_call r_cuisine r_location r_price
Can I help you with anything else?

Did you say you are looking for a restaurant in the r_location of town?
Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area , price range or food type . How may I help you ?
I am sorry but there is no other r_cuisine restaurant in the r_location of town
I am sorry but there is no other r_cuisine restaurant in the r_price price range
I am sorry but there is no other r_cuisine restaurant that matches your request
I am sorry but there is no r_cuisine restaurant that matches your request
I'm sorry but there is no r_cuisine restaurant in the r_location of town
I'm sorry but there is no r_cuisine restaurant in the r_location of town and the r_price price range
I'm sorry but there is no restaurant serving r_cuisine food
I'm sorry but there is no restaurant serving r_price r_cuisine food
Let me confirm , You are looking for a restaurant and you dont care about the price range right?
Let me confirm , You are looking for a restaurant in the r_price price range right?
Ok , a restaurant in any part of town is that right?
r_name is a great restaurant
r_name is a great restaurant serving r_cuisine food and it is in the r_price price range
r_name is a great restaurant serving r_price r_cuisine food in the r_location of town .
r_name is a nice place in the r_location of town
r_name is a nice place in the r_location of town and the prices are r_price
r_name is a nice place in the r_location of town serving tasty r_cuisine food
r_name is a nice restaurant in the r_location of town in the r_price price range
r_name is a nice restaurant in the r_location of town serving r_cuisine food
r_name is in the r_location part of town .
r_name is in the r_price price range
r_name is on r_address
r_name serves r_cuisine food
r_name serves r_cuisine food .
r_name serves r_cuisine food in the r_price price range
Sorry but there is no other r_cuisine restaurant in the r_price price range and the r_location of town
Sorry but there is no other restaurant in the r_price price range and the r_location of town
Sorry I am a bit confused ; please tell me again what you are looking for .
Sorry there is no r_cuisine restaurant in the r_location of town
Sorry there is no r_cuisine restaurant in the r_price price range
Sorry there is no r_price restaurant in the r_location of town serving r_cuisine food

Sorry would you like r_cuisine food or you dont care
Sorry would you like r_cuisine or r_cuisine food?
Sorry would you like something in the r_location or in the r_location
Sorry would you like something in the r_price price range or in the r_price price range
Sorry would you like something in the r_price price range or you dont care
Sorry would you like the r_location of town or you dont care
Sorry, I can't hear you
Sure , r_name is on r_address
The phone number of r_name is r_phone
The post code of r_name is r_post_code
The price range at r_name is r_price .
There are restaurants . That area would you like?
There are restaurants in the r_price price range and the r_location of town . What type of food would you like?
There are restaurants serving r_cuisine food . What area do you want?
There are restaurants serving r_cuisine in the r_price price range . What area would you like?
What kind of food would you like?
What part of town do you have in mind?
Would you like something in the r_price , r_price , or r_price price range?
You are looking for a r_cuisine restaurant right?
You are looking for a restaurant is that right?
You are looking for a restaurant serving any kind of food right?
you are welcome

5.2 Alquist conversational dataset

The Alquist conversational dataset was collected from the socialbot Alquist competing in the Alexa Prize 2017 held by Amazon. The goal of the competition was to hold 20 minutes long coherent and engaging conversation with the user about popular topics. The dataset consists of 37805 dialogues between the user and the socialbot about books. There are 344464 message-response pairs in total. The average length of dialogues is 9.11 pairs, the median is 7 pairs, and there are 23633 unique responses. The dataset was collected on the live version of the system running on the Amazon Echo devices used by customers living in the USA between November 2017 and March 2018. The system was not significantly changed during this period, so the structure of data does not change in time. The dataset is noisy and hard to learn because it contains voice recognition errors and part of the messages come from uncooperative users. Messages from uncooperative users are hard to interpret or out of the domain of books. The example of the dialog included in the dataset is in the Appendix C.

All of 23633 unique responses can be clustered into 30 semantically unique responses. This reduction can be achieved thanks to the fact, that dialogues in socialbot Alquist are represented as state graph. Each node in state graph correspondences to one of 30 semantically unique responses.

The huge number of unique responses is caused by the fact, that responses contain information about authors, books and fun-facts about books and authors obtained from TodayILearned subreddit[1]. Another reason is randomization of phrases of responses. The social bot Alquist chooses randomly between two formulations of the semantically same response like "Tell me, what is your favorite book?" and "What is your favorite book?" for example. The collected data contains the information about the current state of the dialog, which we used to cluster the responses into 30 clusters of responses with the same meaning. Thus we don't have information about the class of response in the dataset. Instead, we have information about the class of cluster of responses.

Table 5.3: List of clusters and examples of responses of Alquist conversational dataset

- Oh, I understand. Do you like it? - I understand. Do you like it?
- I can recommend you The Wings of the Dove. Its author is Henry James. Hmm, it felt OK, but nothing to write home about! What other book would you recommend me? - I can recommend you Light in August. Its author is William Faulkner. A that one was nice to read! Any other book, you would like to talk about?
- Oh, the The Name of the Wind by Patrick Rothfuss? Is it correct? - Ahh, the The Bad Beginning by Lemony Snicket? Am I right?
- Can I recommend you some?
- Oh. Can you say the name of book again? - Sorry, This is embarrassing. I still don't get it. Tell me different book please.
- By the way, kate, here are a few topics we can chat about. I'll be glad to tell you the latest news or gossip, and I also like to talk about video games, books, or movies. Fashion, music, or sports, are not bad either.
- Oh, I am sorry that you didn't like it. A great book! I heard that, Suzanne Collins, author of The Hunger Games, was a writer on Clarissa Explains It All Anyway, let's get back to it. How long is it? - Oh, I am sorry that you didn't like it. A great book! What's your favourite character?

[1]https://www.reddit.com/r/todayilearned/

- Can you tell me more about it? - Can you describe it for me?
- Oh, I am sorry that you didn't like it. Hmm, it felt OK, but nothing to write home about! I heard that, 50 Shades of Grey was originally a twilight fan-fiction Anyway, let's get back to it. What's your favourite character? - Oh, I am sorry that you didn't like it. An that one was OK! What's your favourite part of the book?
- So which one? - So, what's your favourite book?
- I understand. Any other book, you would like to talk about? - Oh, I see. Remember that you can say, let's talk about movies or sports, if you want to switch topic. Anyway, back to the books. Any other book, you would like to talk about?
- Oh, I wish I could have an interesting conversation about this fascinating book, but I still haven't learned enough about it. Maybe next time! - Oh, I wish I could have a thorough conversation about this fascinating book, but I still haven't learned enough about it. Maybe next time!
- By the way, joe, here are a few topics we can chat about. Do you wanna chat for example about movies, sports, music, , or video games? I can also tell you some of the latest news or gossip if you're interested.
- Can you describe it for me?
- Oh, I wish I could have a thorough conversation about this fascinating book, but I still haven't learned enough about it. Maybe next time! - Oh, I wish I could have a deep conversation about this fascinating book, but I still haven't learned enough about it. Maybe next time!
- I said, Oh, I see. How many times have you finished it?
- Oh, the A Game of Thrones from George R.R. Martin? Is it correct? - Oh, the Off the Record by K.A. Linde? Is it correct?
- Can I recommend you some?
- My favorite book is The Lord of the Rings written by J. R. R. Tolkien. I like Sam and Frodo. They are just such a good adventuring pair. Frodo is wise and distant, but he would fail on his own. And Sam doesn't want to be there but is too loyal to leave. It's nice to see an unlikely set of hero's that isn't almost perfect. It's your turn now. What's your favorite book?

- It seems that you rather liked it, that's great. An that one was OK! I heard that, that author Jack London, best known for his book 'Call of the Wild' about Alaskan adventures, also wrote a dystopian scifi novel... and he was a socialist Anyway, let's get back to it. What's your favourite character? - It seems that you rather liked it, that's great. An that one was OK! I know that, The author of Holes, Louis Sachar, also wrote the Wayside School series Anyway, let's get back to it. What's your favourite character?
- By the way, , here are a few topics we can chat about. I love chatting about sports, movies, music, holidays, or video games. I can also tell you some of the latest news or gossip if you're interested. - By the way, , here are a few topics we can chat about. We can chat for example about the latest news, gossip, video games, books, fashion, sports, or movies. I can also sing you a song.
- I can recommend you Light in August. Its author is William Faulkner. A that one was nice to read! I read that, when Snopes creator David Mikkelson first began posting on Internet newsgroups in the late 1980s, he created the username "snopes" based on the surname of a family in a William Faulkner novel Over time, "snopes" gained a reputation for his ability to thoroughly research and debunk false claims Anyway, let's get back to it. Any other book, you would like to talk about? - I can recommend you The Catcher in the Rye. Its author is J.D. Salinger. A that one was nice to read! I heard that, Author J.D Salinger served in the U.S Army's Counter Intelligence Corps (CIC) during World War II He could speak both French and German and his work involved interviewing enemy prisoners and civilians Anyway, let's get back to it. Any other book, you would like to talk about?
- Wow, that's great that you enjoyed it. A great book! I read that, that you can listen to The Hitchhikers Guide to The Galaxy audio book, in it's entirety, on Grooveshark Anyway, let's get back to it. How long is it? - Wow, that's great that you enjoyed it. An that one was OK! I read that, F Scott Fitzgerald wrote to Will Cather explaining his fear that she might feel he plagiarized her character Marian Forrester from Lost Lady in his book The Great Gatsby Anyway, let's get back to it. How long is it?
- I said, Would you recommend it to me?
- I understand. How long is it? - I understand. What's your favourite part of the book?
- What book do you like?
- I'm sorry, I don't know it. What is it about? - I'm sorry, I don't know that one. Can you describe it for me?

- Wow, that's great that you enjoyed it. I liked this one a lot, it was great. I read that, that Truman Capote and Harper Lee were childhood friends and that she helped him interview residents and investigators for the book In Cold Blood Anyway, let's get back to it. How long is it? - Wow, that's great that you enjoyed it. I really enjoyed reading this one. It was very nice. I know that, J.K Rowling was turned down by 26 publishers, when she finally got a deal her publisher told her to also get a day job as she'll not make a living from childrens books Anyway, let's get back to it. What's your favourite part of the book?
- Remember that you can say, let's talk about movies or sports, if you want to switch topic. Anyway, back to the books. What other book would you recommend me? - Remember that you can say, tell me joke or fun fact, if you want to switch topic. Anyway, back to the books. Is there any other book you like?
- It seems that you rather liked it, that's great. An that one was OK! I read that, that George Orwell, author of "1984" and "Animal Farm," gave the U.K government a list he had written of people he suspected of being "crypto-communists" Anyway, let's get back to it. How long is it? - It seems that you rather liked it, that's great. Hmm, it felt OK, but nothing to write home about! How long is it?

Chapter 6

Experiments

We did several experiments to evaluate the architectures described in the Chapter 4. We evaluate each architecture using pretrained word2vec[11] embeddings and fastText[13] embeddings trained on the dataset. We used bAbI Task 6 dataset and Alquist conversational dataset for testing. We also used bAbI Task 5 dataset to verify the replicability of the results of baseline architecture as presented in the [1].

We evaluate the turn accuracy and dialogue accuracy of the models to be in line with the papers on bAbI Task datasets [9]. The turn accuracy is the ratio of correctly predicted dialogue turns to all dialogue turns in the dataset. The dialogue accuracy is the ration of dialogues, in which all turns were predicted correctly, to all dialogues in the dataset.

We used the Bayesian Optimization of hyperparameters. We show that this technique can find values of hyperparameters, which improve the performance of the system.

6.1 bAbI Task 5

We used the Hybrid code network architecture to verify the results measured on bAbI Task 5 dataset presented in [1]. We used the 128 hidden units and trained the model for 12 epochs. We used pretrained word2vec embeddings, which were static and not updated during the training.

The action mask consists of the following if-then rules:

- Don't ask for the entity which is already known
- Do not make API call if all preconditions have not been satisfied yet
- Don't present the results, if the results from the database have not been received yet

We didn't use any additional input features.

We measured the testing set containing the out-of-vocabulary words. We achieved the same results of 100% turn and dialogue accuracy on the testing set containing out-of-vocabulary words. We didn't test further models on this dataset because of achieved accuracies. The results are presented in the table 6.1.

Table 6.1: bAbI Task 5 results

Model	Task5-OOV	
	Turn Acc.	Dialog Acc.
Borders and Weston (2017) [9]	77.7%	0.0%
Liu and Perez (2016) [74]	79.4%	0.0%
Seo et al. (2016) [75]	96.0%	-
Williams, Asadi and Zweig (2017) [1]	**100%**	**100%**
Replicated Hybrid code network model	**100%**	**100%**

6.2 bAbI Task 6

We measured the performance of all three models described in the chapter 4 using the bAbI Task 6 dataset.

We evaluated each model using two types of embedding vectors. We used the pretrained 300-dimensional word2vec embedding vectors trained on News[1]. Word2vec embeddings were static and not updated during training. The second set of embedding vectors which we used were 300-dimensional fastText embedding vectors pretrained on the training set of bAbI Task 6 dataset. The embeddings were trained during 100 epochs. The fastText embeddings were fixed and not updated during the training too.

The action mask consists of following if-then rules:

- Do not make API call if all preconditions have not been satisfied yet
- Don't present the results, if the results from the database have not been received yet

We used the additional input features:

- Entities present in the current message
- Entities present in context
- Has been the database queried yet
- Is the result of database query empty
- Does the requested type of cuisine exist
- Has been any result of database query presented yet
- Have been all results of database query presented already

6.2.1 Bayesian hyperparameter optimization

We performed the hyperparameter optimization using Bayesian optimization[2] on the validation set. We performed 30 rounds of training with a different set of hyperparameters for each model. Each training was 30 epochs long. We evaluated the performance of model after each

[1] GoogleNews-vectors-negative300.bin
[2] We used the implementation from Scikit-Optimize library https://scikit-optimize.github.io/

Table 6.2: Hyperparameter optimization bounds

Hyperparameter	Min. value	Max. value
LSTM size	10	512
Input LSTM size*	10	512
Convolutional filtersˆ	5	64
LSTM dropout	0.5	1
Input LSTM dropout*	0.5	1
Convolutional dropoutˆ	0.5	1
Fully connected dropout	0.5	1
Learning rate	1E-5	1
Activation function	tanh; relu	
Input activation function*	tanh; relu	
Adam epsilon	1E-8; 0.1; 1	
Adam beta1	0.5; 0.9	

ˆ are used in CNN input only, * are used in RNN input only

Table 6.3: Word2vec bAbI Task 6 hyperparameter optimization results

LSTM size	85	512	15	184	512	25	10	35	10
LSTM dropout	0.92	0.5	0.98	0.89	0.5	0.79	1.0	0.74	1.0
Fully connected dropout	0.59	0.89	0.87	0.55	0.5	0.97	0.5	0.5	0.78
Learning rate	0.001	0.0007	0.02	0.002	0.003	0.005	0.01	0.003	0.003
Activation function	tanh	relu	tanh	tanh	relu	relu	tanh	tanh	tanh
Adam epsilon	1E-8	0.1	0.1	1.0	1.0	1E-8	1.0	1.0	1.0
Adam beta1	0.5	0.5	0.5	0.9	0.5	0.9	0.5	0.5	0.5
Turn accuracy	**71.3%**	69.2%	68.9%	64.4%	63.4%	61.3%	61.3%	59.7%	58.3%

epoch and saved the weights of model achieving the highest turn accuracy on the validation set for each set of hyperparameters. The bounds of hyperparameters for each model are presented in the table 6.2.

We show the nine hyperparameter settings achieving the highest validation turn accuracies for each model and both sets of embedding vectors in the tables 6.3, 6.4, 6.5, 6.6, 6.7 and 6.8. The hyperparameter settings are sorted by the turn accuracy in the tables.

We trained each model with the hyperparameters, which achieved the highest validation accuracy. We used the 12 epochs of training. We measured the turn accuracy on validation set after each training epoch and saved the weights achieving highest validation turn accuracy, which we used for testing. The graphs of progress of turn and dialogue accuracies for training

Table 6.4: Word2vec+CNN bAbI Task 6 hyperparameter optimization results

LSTM size	109	512	512	192	512	416	356	455	504
Convolutional filters	6	15	5	20	62	50	40	47	60
LSTM dropout	0.79	0.98	0.76	0.93	0.96	0.77	0.85	0.96	0.86
Convolutional dropout	0.84	0.91	1.0	0.74	0.65	0.64	0.5	0.65	0.88
Fully connected dropout	0.93	0.95	0.86	0.57	0.65	0.59	0.82	0.98	0.90
Learning rate	0.005	0.0006	0.0003	0.05	0.003	0.005	0.2	0.001	0.001
Activation function	tanh	tanh	relu	relu	relu	tanh	tanh	relu	relu
Adam epsilon	0.1	0.1	1E-8	1.0	0.1	1.0	1.0	1.0	0.1
Adam beta1	0.5	0.9	0.9	0.5	0.5	0.5	0.5	0.9	0.9
Turn accuracy	**70.4%**	68.7%	68.7%	68.0%	67.7%	66.7%	62.8%	62.8%	62.5%

Table 6.5: Word2vec+RNN bAbI Task 6 hyperparameter optimization results

LSTM size	219	345	493	109	76	137	473	265	134
Input LSTM size	312	507	15	403	11	77	371	216	255
LSTM dropout	0.74	0.99	0.88	0.65	0.66	0.74	0.51	0.94	0.55
Input LSTM dropout	0.91	0.77	0.74	0.51	1.0	0.66	0.86	0.69	0.78
Fully connected dropout	0.98	0.90	0.79	0.68	0.93	0.51	0.94	0.68	0.63
Learning rate	0.00005	0.04	0.02	0.0002	0.003	0.007	0.0004	0.004	0.0001
Activation function	relu	tanh	tanh	relu	relu	relu	tanh	tanh	relu
Input activation function	tanh	relu	tanh	tanh	relu	tanh	tanh	relu	relu
Adam epsilon	1E-8	0.1	0.1	1E-8	0.1	0.1	0.1	1.0	1E-8
Adam beta1	0.9	0.9	0.9	0.5	0.5	0.5	0.5	0.5	0.9
Turn accuracy	**65.5%**	64.9%	64.9%	63.5%	62.6%	62.5%	62.5%	62.4%	60.4%

Table 6.6: fastText bAbI Task 6 hyperparameter optimization results

LSTM size	55	366	512	438	254	203	512	10	190
LSTM dropout	0.85	0.57	0.95	0.65	0.65	0.56	1.0	1.0	0.5
Fully connected dropout	0.82	0.59	0.75	0.64	0.53	0.81	0.5	1.0	0.83
Learning rate	0.008	0.007	0.01	0.04	0.0001	0.00003	0.00001	0.01	0.009
Activation function	relu	tanh	relu	relu	relu	tanh	tanh	relu	relu
Adam epsilon	1E-8	1.0	0.1	1.0	1E-8	1E-8	1E-8	1.0	0.1
Adam beta1	0.9	0.5	0.5	0.5	0.5	0.9	0.9	0.9	0.9
Turn accuracy	**69.4%**	68.9%	68.4%	68.3%	68.1%	67.2%	65.7%	65.5%	65.4%

Table 6.7: fastText+CNN bAbI Task 6 hyperparameter optimization results

LSTM size	245	444	255	50	465	264	469	469	37
Convolutional filters	21	50	31	7	5	46	10	62	63
LSTM dropout	0.80	0.70	0.61	0.97	0.50	0.81	0.90	0.92	0.98
Convolutional dropout	0.72	0.76	0.80	0.52	0.60	0.68	0.50	0.51	0.98
Fully connected dropout	0.79	0.96	0.88	0.95	0.50	0.82	0.51	0.99	0.65
Learning rate	0.0001	0.05	0.00003	0.01	0.004	0.001	0.00001	0.00001	0.00001
Activation function	relu	tanh	tanh	relu	tanh	tanh	relu	relu	relu
Adam epsilon	1E-8	1.0	1E-8	1.0	0.1	1E-8	1E-8	1E-8	1E-8
Adam beta1	0.5	0.9	0.5	0.5	0.9	0.9	0.9	0.5	0.5
Turn accuracy	**71.5%**	69.8%	69.1%	68.6%	65.6%	65.6%	64.9%	60.9%	58.7%

Table 6.8: fastText+RNN bAbI Task 6 hyperparameter optimization results

LSTM size	505	313	504	290	498	510	139	21	229
Input LSTM size	199	400	18	470	488	485	399	64	149
LSTM dropout	0.94	0.61	0.98	0.70	0.54	0.98	0.88	0.99	0.51
Input LSTM dropout	0.97	0.62	0.78	0.72	0.58	0.98	0.69	0.88	0.75
Fully connected dropout	0.76	0.83	0.55	0.75	0.57	0.96	0.74	0.50	0.93
Learning rate	0.0003	0.004	0.006	0.00003	0.0001	0.7	0.0001	0.00001	0.00009
Activation function	relu	relu	tanh	tanh	tanh	tanh	relu	tanh	relu
Input activation function	tanh	tanh	relu	tanh	relu	tanh	tanh	relu	tanh
Adam epsilon	1E-8	0.1	0.1	1E-8	1E-8	1.0	0.1	1E-8	0.1
Adam beta1	0.5	0.5	0.9	0.5	0.9	0.9	0.5	0.9	0.9
Turn accuracy	**68.0%**	67.3%	67.3%	67.0%	66.8%	61.0%	60.4%	55.5%	54.9%

and validation sets after each epoch for all models are in figures 6.1, 6.2, 6.3 and 6.4. The final results on testing set and its comparison with other models are in the table 6.9.

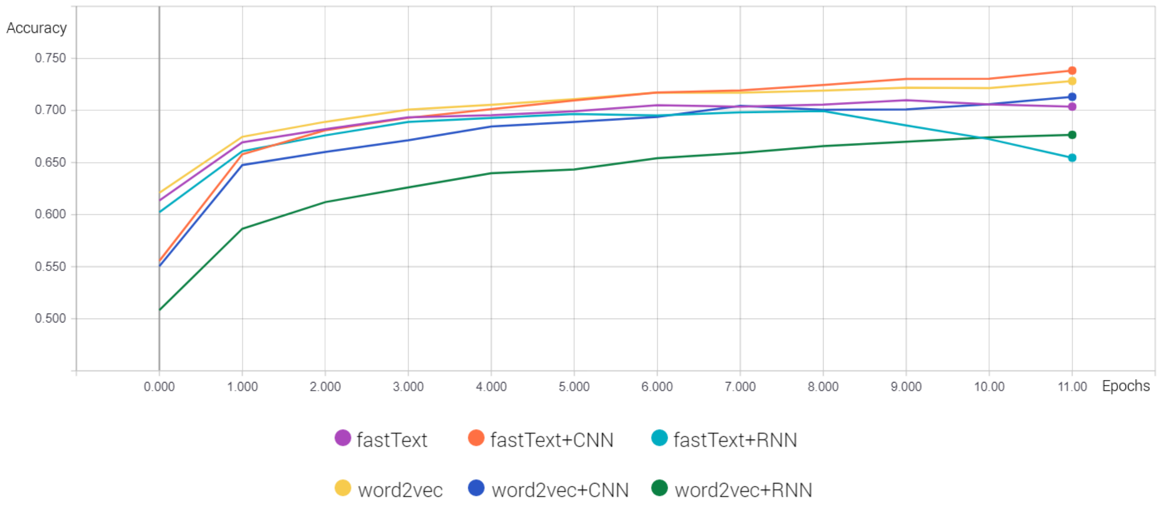

Figure 6.1: bAbI Task 6 Training turn accuracy

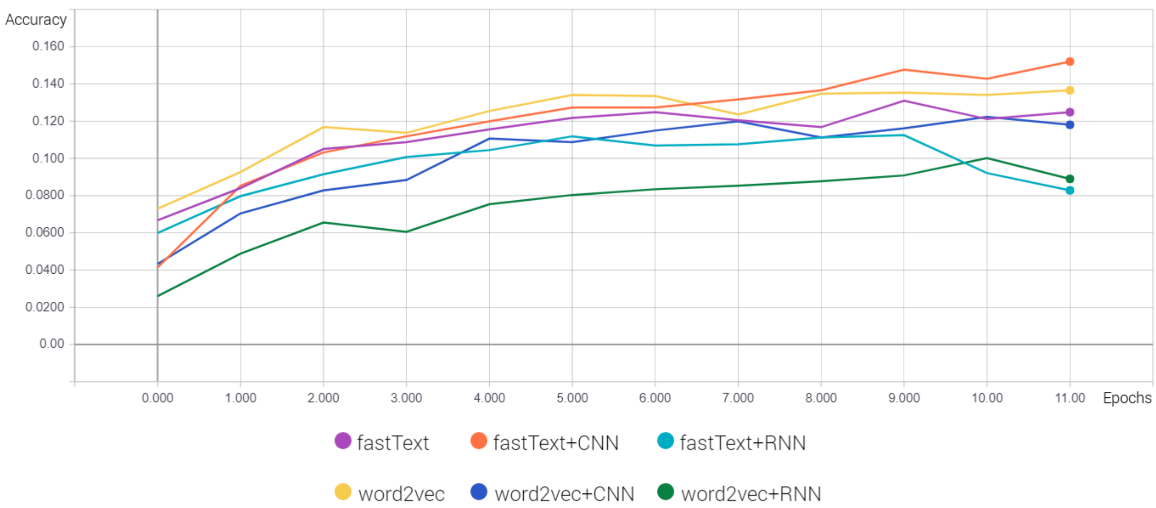

Figure 6.2: bAbI Task 6 Training dialogue accuracy

6.3 Alquist conversational dataset

We measured the performance of all models on the Alquist conversational dataset. We used the pretrained word2vec embedding vectors and fastText embedding vectors trained for 100 epochs on the training set of Alquist conversational dataset. We didn't perform hyperparameter optimization due to a long time of training. We used hyperparameters founded in the section 6.2.1 for each model instead.

We didn't use the action mask, because it is not clear which rules would improve the accuracy of models for Alquist conversational dataset.

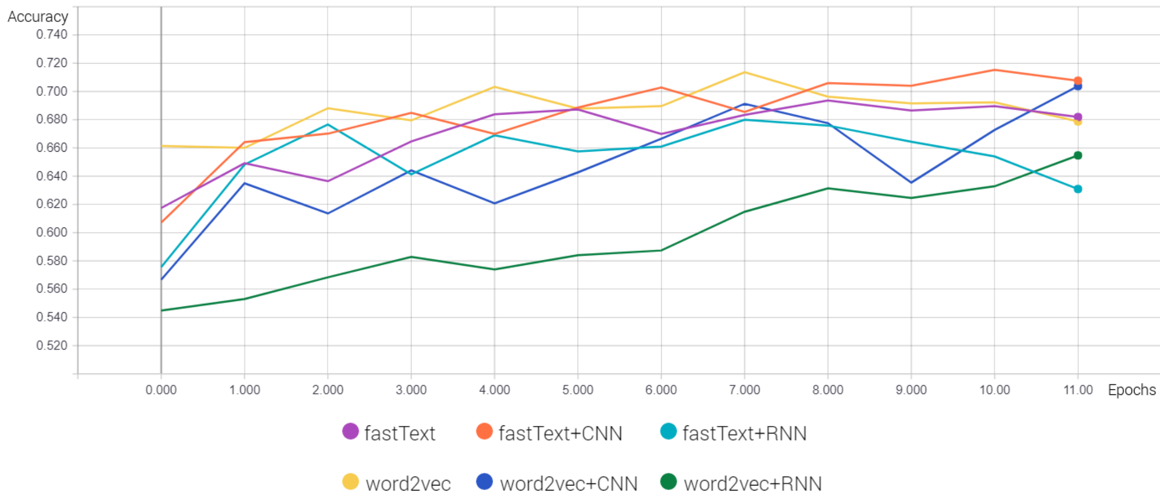

Figure 6.3: bAbI Task 6 Validation turn accuracy

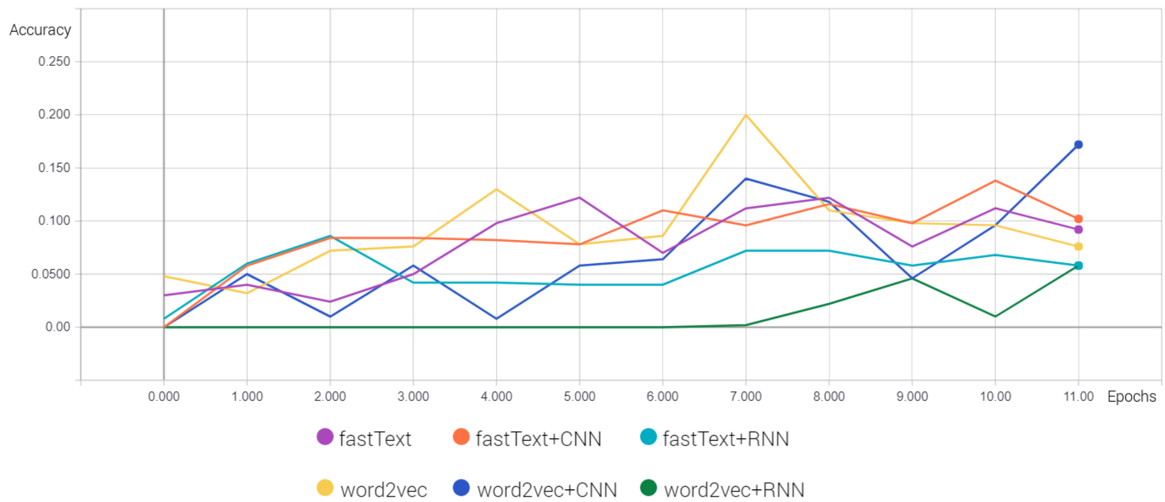

Figure 6.4: bAbI Task 6 Validation dialogue accuracy

We used the additional input feature signalizing, whether the requested book was founded in the database or not.

We trained the models for 12 epochs. We measured the turn accuracy after each epoch and saved the weights which achieved the highest turn accuracy. We measured turn and dialogue accuracies on the testing set using saved weights.

The graphs of progress of turn and dialogue accuracies for training and validation sets after each epoch for all models are in figures 6.5, 6.6, 6.7 and 6.8. The drop of the line recording progression of accuracy for word2vec+RNN model is affected by the error in the training. We didn't rerun the experiment, because there is no evidence, that the model would achieve any significantly better results. The final results on testing set and its comparison with other models are in the table 6.9.

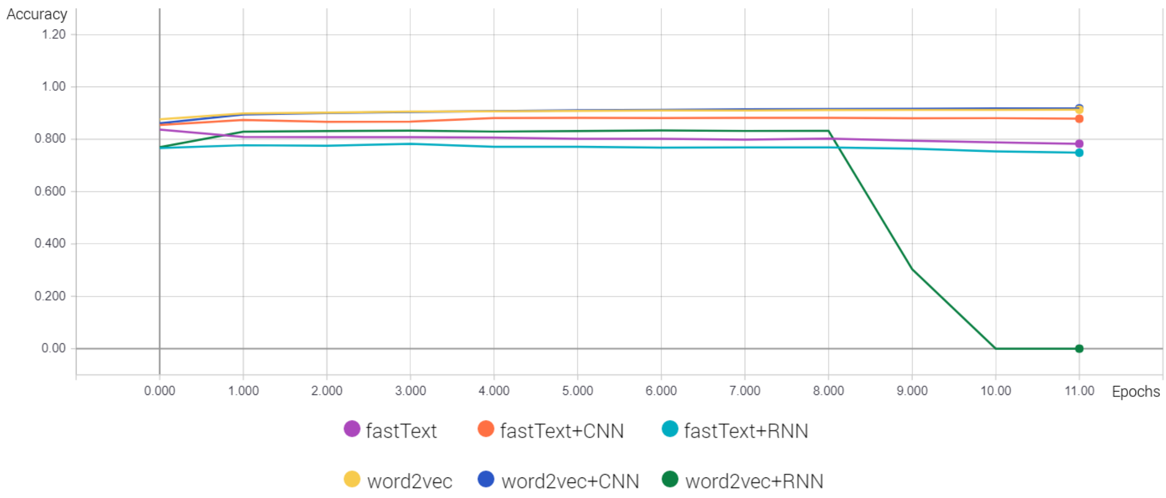

Figure 6.5: Alquist conversational dataset Training turn accuracy

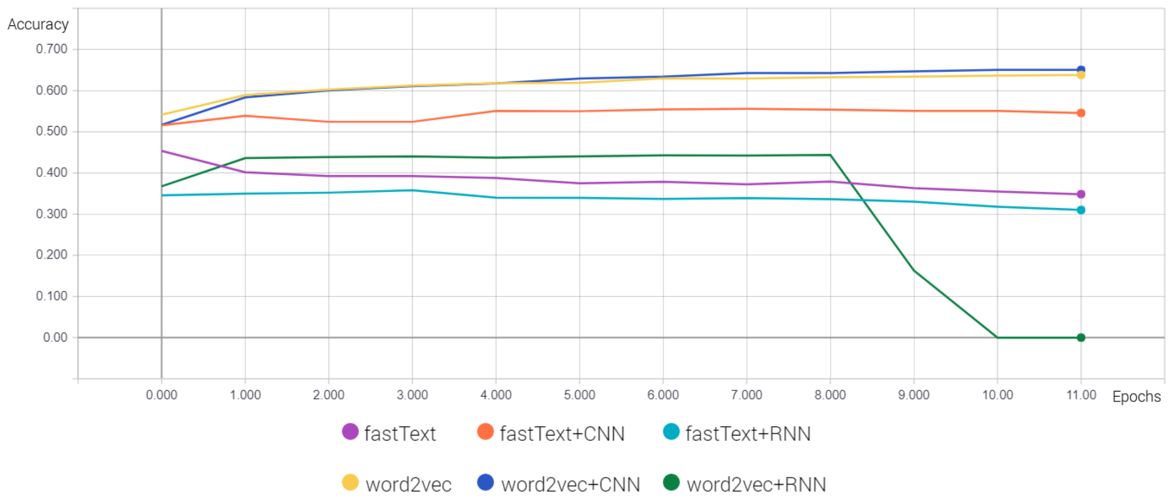

Figure 6.6: Alquist conversational dataset Training dialogue accuracy

6.4 Results

We will evaluate the results of experiments in this section.

The best model regarding turn accuracy on bAbI Task 6 dataset is model using convolutional input layer and fastText embedding vectors, which outperformed the baseline [1]. This model achieved turn accuracy of 58.9%, which is better than 55.6% achieved by [1]. For the dialogue accuracy on bAbI Task 6 dataset, the baseline [1] has not been outperformed. Our best model using bag-of-words and average of fastText embedding vectors as input features achieved dialogue accuracy of 0.8% which is less than 1.9% achieved by the baseline.

The best model regarding turn accuracy on Alquist conversational dataset is model using convolutional input layer and word2vec embedding vectors, which achieved turn accuracy of 92.6%. For the dialogue accuracy on Alquist conversational dataset, the best performing model is model using bag-of-words and average of word2vec embedding vectors as input features. It achieved the accuracy of 68.0%. The comparison with the baseline is not possible,

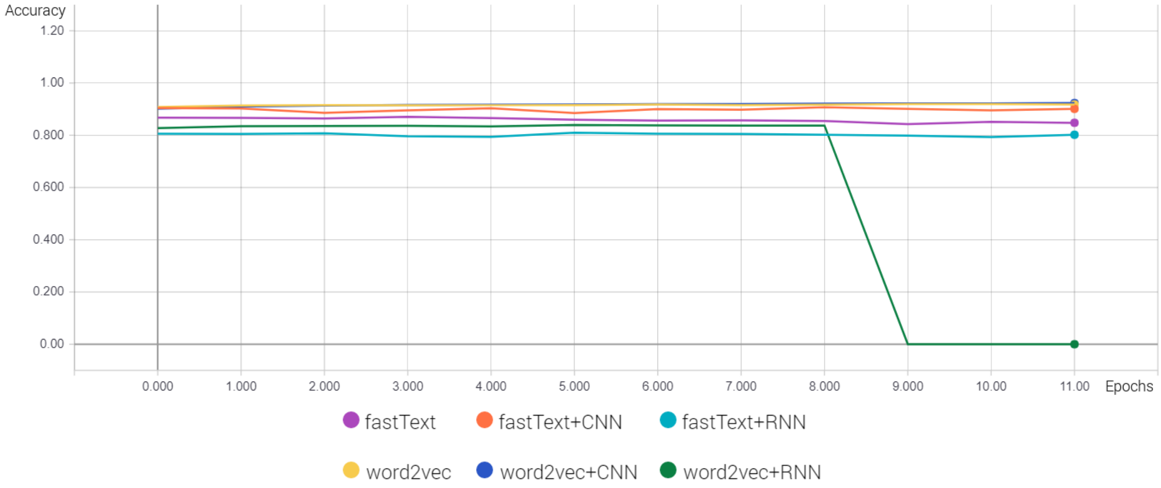

Figure 6.7: Alquist conversational dataset Validation turn accuracy

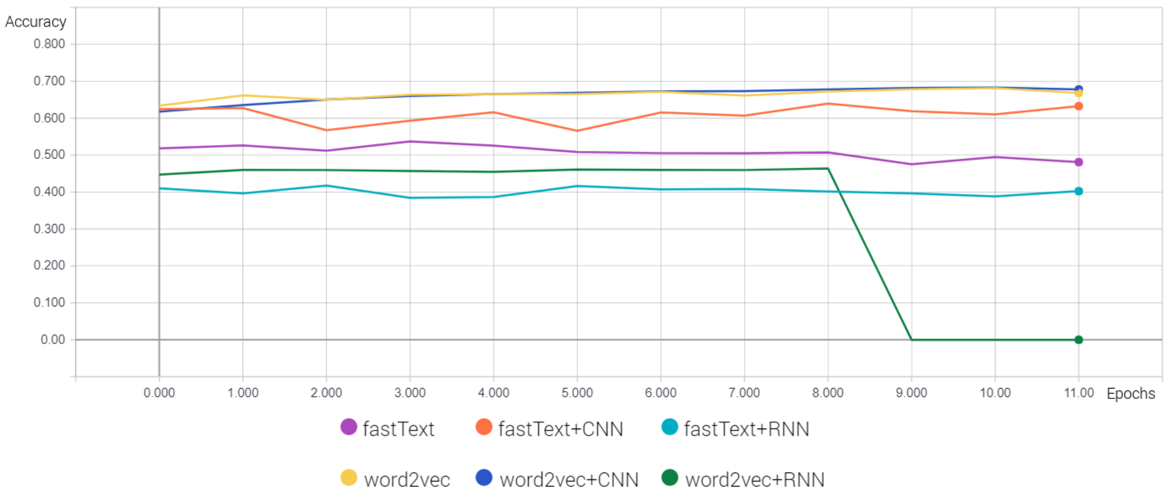

Figure 6.8: Alquist conversational dataset Validation dialogue accuracy

because it has not been tested on Alquist conversational dataset.

The first thing we can notice, is that we outperformed our baseline (55.6%) described in [1] by fastText (57.6%), fastText+CNN (58.9%), word2vec (57.4%) and word2vec+CNN (56.3%) architectures in turn accuracy. Only the models containing recurrent input layer perform worse than the baseline. Possible cause of this improvement is the hyperparameter optimization, which founded the set of hyperparameters with the best performance. The method of hyperparameter optimization is not reported in the paper describing the baseline model. This is the reason, why we can't confirm this hypothesis.

Our models underperformed the baseline model in dialogue accuracy on bAbI Task 6 dataset. One of the reason may be the action mask, which can differ in our model implementations. The action mask is described only vaguely in the paper describing baseline model[9]. Different rules forming the action mask may lead to worse results.

The best performing models are the models using convolutional input layer (fastText+CNN

Table 6.9: Testing accuracy

Model	bAbI6		Alquist	
	Turn Acc.	Dialogue Acc.	Turn Acc.	Dialogue Acc.
Bordes and Weston (2017) [9]	41.1%	0.0%	-	-
Liu and Perez (2016) [74]	48.7%	1.4%	-	-
Eric and Manning (2017) [76]	48.0%	1.5%	-	-
Seo et al. (2016) [75]	51.1%	-	-	-
Williams, Asadi and Zweig (2017) [1]	55.6%	**1.9%**	-	-
fastText	57.6%	0.8%	86.9%	51.7%
fastText+CNN	**58.9%**	0.5%	90.6%	63.0%
fastText+RNN	54.9%	0.3%	80.6%	40.5%
word2vec	57.4%	0.4%	92.2%	**68.0%**
word2vec+CNN	56.3%	0.1%	**92.6%**	67.8%
word2vec+RNN	54.6%	0.1%	83.9%	45.2%

and word2vec+CNN). It holds true for bAbI Task 6 dataset and also for Alquist conversational dataset. The second best models are models using bag-of-words and average of embedding vectors as input features (fastText and word2vec). The worst models use recurrent input layer (fastText+RNN and word2vec+RNN). These models don't perform better than the baseline model.

Models using fastText embedding vectors performs better than corresponding models using word2vec embedding vectors in case of the smaller bAbI Task 6 dataset. On the other hand, models using word2vec embedding vectors achieve higher turn accuracy than corresponding models using fastText word embeddings in the case of the bigger Alquist conversational dataset.

Chapter 7

Conclusion

The goal of this thesis was to research the current methods of dialogue management for conversational artificial intelligence, implement the dialogue manager and apply it to the data of interactions between humans and socialbot Alquist collected during Alex Prize 2017 [2].

We first introduced the dialogue manager and described methods how they select the responses. We divided the dialogue managers into retrieval-based and generative-based, and rule-based, machinelearning-based and hybrid-based dialogue managers. We also described the context and its importance for conducting the dialogues in natural language. We then described the representation of words by word embedding vectors and introduced word2vec, GloVe and fastText word embeddings, which we used later in experiments. We also described the neural networks, recurrent neural networks and convolutional neural networks, which are used in our proposed dialogue manager. We described the methods of hyperparameter optimization, which we decided to use to find the best set of hyperparameters for our dialogue manager.

We explored the recent dialogue managers. We started from rule-based dialogue managers, and continued through dialogue managers using TF-IDF vectors with cosine similarity function, word embedding vectors with cosine similarity and supervised models. We described the functioning of the seq2seq models, memory networks and hybrid code networks in detail.

We described the dialogue bAbI Task 5 and 6 datasets and how we preprocessed them for the experiments. We also presented the Alquist conversational dataset, and we described how we extracted data containing dialogues about books from the dataset of all conversations between humans and mostly rule-based socialbot Alquist.

We proposed three architectures of our dialogue manager inspired by the Hybrid code networks. The baseline architecture uses the same inputs as the Hybrid code networks, which are bag-of-words and average of word embeddings. The next two architectures use the recurrent neural network and convolutional neural networks respectively.

We measured the dialogue and turn accuracy of the baseline architecture on the dialogue bAbI Task 5 dataset. We replicated results of [1] by achieving 100% of both turn and dialogue accuracy. We measured the dialogue and turn accuracy of all three proposed architectures on dialogue bAbI Task 6 dataset. We measured each architecture with two sets of word embedding vectors. The first set was pretrained word2vec vectors trained on the news. The second test was trained directly on the dialogue bAbI Task 6 dataset. We used Bayesian hyperparameter optimization to find the optimal set of hyperparameter for each architecture. We demonstrated that hyperparameter optimization leads to better turn accuracy. We measured the turn and dialogue accuracy with the best-found set of hyperparameters on Alquist conversational dataset.

We achieved better performance in the turn accuracy than our baseline [1]. Our model containing convolutional neural network as input layer and fastText word embeddings achieved turn accuracy of 58.9% versus the 55.6% of the baseline reported in [1]. The model with the convolutional neural network as input layer and word2vec word embeddings achieved only a slightly worse accuracy of 57.4% than our the best architecture. We were not able to outperform the baseline model in case of dialogue accuracy. We compared the turn accuracy of models with bag-of-words input and average of word embeddings and convolutional network and recurrent network inputs. The best performance gave the model with the convolutional neural network as the input layer, followed by the bag of words and average of word embeddings, and recurrent neural network as the input layer, for both datasets and sets of word embeddings. We also tested the influence of the word embedding vectors. The fastText word embeddings, which we trained directly on the training dataset, performed better on the smaller dataset dialogue bAbI Task 6. The word2vec word embedding vectors, which were pretrained on the news dataset, performed better on the bigger Alquist conversational dataset.

To conclude, we showed that our proposed architecture with the convolutional neural network as an input layer can outperform the baseline model of Hybrid code networks [1]. We also confirmed that it is possible to apply it to the data collected from the rule-based dialogue system and achieve reasonable turn accuracy.

We would like to explore simpler architectures as our future work. We would like to focus on architectures not so complicated as Hybrid code networks relying on the stateful LSTM.

Bibliography

[1] J. D. Williams, K. Asadi, and G. Zweig, “Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning”, 2017.

[2] A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, E. King, K. Bland, A. Wartick, Y. Pan, H. Song, S. Jayadevan, G. Hwang, and A. Pettigrue, “Conversational ai: The science behind the alexa prize”, 2018.

[3] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. 2017, https://web.stanford.edu/~jurafsky/slp3/.

[4] D. Britz, *Deep Learning for Chatbots*. 2015, http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.

[5] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, “Building end-to-end dialogue systems using generative hierarchical neural network models”, 2016.

[6] J. Weizenbaum, “Eliza - a computer program for the study of natural language communication between man and machine”, 1966.

[7] O. Vinyals and Q. V. Le, “A neural conversational model”, 2015.

[8] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, “Deep reinforcement learning for dialogue generation”, 2016.

[9] A. Bordes, Y.-L. Boureau, and J. Weston, “Learning end-to-end goal-oriented dialog”, 2017.

[10] T. Mikolov, W. tau Yih, and G. Zweig, “Linguistic regularities in continuous space word representations”, 2013.

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

[12] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation”, 2014.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information”, 2016.

[14] *Vector Representations of Words*, https://www.tensorflow.org/versions/master/tutorials/word2vec.

[15] R. Varma, *Language Models, Word2Vec, and Efficient Softmax Approximations.* 2017, http://rohanvarma.me/Word2Vec/.

[16] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation", 2013.

[17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification", 2016.

[18] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models", 2016.

[19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", 1958.

[20] R. Lopez, *Perceptron: the main component of neural networks*, https://www.neuraldesigner.com/blog/perceptron-the-main-component-of-neural-networks.

[21] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions", 1998.

[22] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop", 1998.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", 2010.

[24] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models", 2013.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", 2015.

[26] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)", 2016.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[28] Y. LeCun, "A theoretical framework for back-propagation", 1988.

[29] M. A. Nielsen, *Neural Networks and Deep Learning.* Determination Press, 2015, http://neuralnetworksanddeeplearning.com.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", 2014.

[31] M. Claesen and B. D. Moor, "Hyperparameter search in machine learning", 2015.

[32] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", 2012.

[33] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization", 2011.

[34] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms", 2012.

[35] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based hyperparameter optimization through reversible learning", 2015.

[36] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, "Evolving deep neural networks", 2017.

[37] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks", 2017.

[38] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning", 2018.

[39] D. Britz, *Introduction to RNNs.* 2015, `http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/`.

[40] A. Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks.* 2015, `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

[41] C. Olah, *Understanding LSTM Networks.* 2015, `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[42] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", 1994.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory", 1997.

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

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", 2014.

[46] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks", 2017.

[47] A. Karpathy, *Convolutional Neural Networks for Visual Recognition.* 2015, `http://cs231n.github.io/convolutional-networks/`.

[48] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition", 2010.

[49] *Recognize Digits*, `http://www.paddlepaddle.org/docs/develop/book/02.recognize_digits/index.html`.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", 2015.

[51] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, 2015.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”, 2014.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, 2012.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, 1998.

[55] H. Khalajzadeh, M. Mansouri, and M. Teshnehlab, “Face recognition using convolutional neural network and simple logistic classifier”, 2012.

[56] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection”, 2015.

[57] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation”, 2015.

[58] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs”, 2014.

[59] A. Toshev and C. Szegedy, “Deeppose: Human pose estimation via deep neural networks”, 2014.

[60] M. Linna, J. Kannala, and E. Rahtu, “Real-time human pose estimation from video with convolutional neural networks”, 2014.

[61] P. Y. Simard, D. Steinkraus, and J. C. Platt, “Best practices for convolutional neural networks applied to visual document analysis”, 2003.

[62] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga, “Pixel-wise binarization of musical documents with convolutional neural networks”, 2017.

[63] O. Abdel-Hamid, A. rahman Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition”, 2014.

[64] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. Courville, “Towards end-to-end speech recognition with deep convolutional neural networks”, 2014.

[65] Y. Kim, “Convolutional neural networks for sentence classification”, 2014.

[66] M. Hughes, S. Kotoulas, and T. Suzumura, “Medical text classification using convolutional neural networks”, 2017.

[67] B. Bai, J. Weston, R. Collobert, and D. Grangier, “Supervised semantic indexing”, 2009.

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

[69] M.-T. Luong, “Neural machine translation”, 2016.

[70] J. Weston, S. Chopra, and A. Bordes, “Memory networks”, 2015.

[71] *Memory Networks (MemNN)*, `http://deeplearning.hatenablog.com/entry/memory_networks`.

[72] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “End-to-end memory networks”, 2015.

[73] M. Henderson, B. Thomson, and J. Williams, “The second dialog state tracking challenge”, 2014.

[74] F. Liu and J. Perez, “Gated end-to-end memory networks”, 2016.

[75] M. J. Seo, H. Hajishirzi, and A. Farhadi, “Query-regression networks for machine comprehension”, 2016.

[76] M. Eric and C. D. Manning, “A copy-augmented sequence-to-sequence architecture gives good performance on task oriented dialogue”, 2017.

Appendix A

Dialog bAbI task 5 example

good morning
hello what can i help you with today
i'd like to book a table with italian food
i'm on it
[silence]
where should it be
in paris
how many people would be in your party
for six people please
which price range are looking for
in a cheap price range please
ok let me look into some options for you
[silence]
api_call italian paris six cheap
[database results returned]
[silence]
what do you think of this option:
resto madrid cheap spanish 8stars
no this does not work for me
sure let me find an other option for you
[silence]
what do you think of this option:
resto madrid cheap spanish 6stars
do you have something else
sure let me find an other option for you
[silence]
what do you think of this option:
resto madrid cheap spanish 1stars

it's perfect
great let me do the reservation
may i have the phone number of the restaurant
here it is resto madrid cheap spanish 1stars phone
thanks
is there anything i can help you with
no thank you
you're welcome

Appendix B

Dialog bAbI task 6 example

Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area , price range or food type . How may I help you ?
im looking for canapes food in any price range
I am sorry but there is no canapes restaurant that matches your request
how about european
api_call european R_location R_price
[database results returned]
[silence]
michaelhouse cafe is a great restaurant serving european food and it is in the expensive price range
food an
What part of town do you have in mind?
any any part of town
michaelhouse cafe serves european food .
may i have the address and phone number please
The phone number of michaelhouse cafe is michaelhouse cafe phone
thank you good bye
you are welcome

Appendix C

Alquist Conversational Dataset examples

What book do you like?
13 reasons why
Ahh, the Thirteen Reasons Why by Jay Asher? Am I right?
yes
Would you recommend it to me?
yes
Oh, I see. Do you like it?
yes
Wow, that's great that you enjoyed it. I really enjoyed reading this one. It was very nice. How was it, isn't it too long?
no
Oh, I understand. Remember that you can say, tell me joke or fun fact, if you want to switch topics. Anyway, back to the books. Is there any other book you like?
yes
So which one?
fault in our stars
I'm sorry, I don't know this book. What is it about?
the fault in our stars
Oh, I see. How was it, isn't it too long?
yes
I understand. What other book would you recommend me?
paper towns
Oh, the Paper Towns from John Green? Is it correct?
yes
Would you recommend it to me?
yes
I understand. Do you like it?
yes
Wow, that's great that you enjoyed it. A that one was nice to read! How was it, isn't it too long?
no
I see. Is there any other book you like?
yes
So which one?

abundance of Catherines
Oh, the An Abundance of Katherines by John Green? Is it correct?
yes
Should I read it?
yes
I understand. Do you like it?
yes
Wow, that's great that you enjoyed it. An that one was OK! What's your favourite character?

What book do you like?
the war that saved my life
I'm sorry, I don't know that one. Can you describe it for me?
it's about this girl who lives in England and there were bombs so he had to move to someone in London he now she's living her life there
Oh, I see. How long is it?
oh it's about 300 pages
I understand. Remember that you can say, let's talk about movies or sports, if you want to switch topics. Anyway, back to the books. What other book would you recommend me?
I would recommend reading a series of unfortunate events
Oh, the The Bad Beginning from Lemony Snicket? Is it correct?
yes
How many times have you finished it?
1:57 times
I understand. Do you like it?
yes do you
Wow, that's great that you enjoyed it. It was fine, but nothing exceptional. I know that, author Daniel Handler (Lemony Snicket) made racist comments towards Jacqueline Woodson at an award ceremony. Anyway, let's get back to it. What's your favourite part of the book?
and then change your and go to another person Taos an cow Olaf finds them anyway
Oh, I see. Any other book, you would like to talk about?