Improving task for dialog system: a proposal

Felipe Salvatore

June 2, 2018

Abstract

to do

Contents

Chapter 1

Introduction

1.1 Old quali introduction

In this PhD thesis we will investigate the problem of **building a non-task-oriented dialog system**. This problem is central to the field of *artificial intelligence* (AI) due to the seminal paper by Alan Turing [24] in which he proposed an imitation game. The idea behind the game was simple: three players A, B and C can interact only by nonpersonal communication. C knows that he will interact with a computer (A) and a person (B), but he does not know which is which. C should exchange some conversation both with A and B; after some time he should guess who is the computer and who is the human. The goal of A is to be as human as possible in order to fool C; and the goal of B is to help C come to the right answer.

This is the famous *Turing Test*. At that time Turing proposed that test as a way to make more concrete the philosophical question "can machines think?". This can lead to a whole discussion of the nature of thinking and intelligence. As a mature field, AI has distantiate itself from these abstract an general questions and have concentrated in *building agents to solve tasks reserved exclusively to humans*: playing chess, driving a car, cleaning a room, writing a letter to a friend, etc.

The sub-area of AI that deals with problems involving human language is called *natural language processing* (NLP). It concentrate in tasks such text understanding (por mais coisa). NLP , as the field of AI as a whole, have change dramatically in the past 10 years. For some time the main paradigm in the field was the so called **knowledge base approach**: the main goal

of AI was to develop intelligent systems capable of solving certain classes of problems by having a *representation* or *model* of the world. In this view, a decision by a machine is synonymous to *inferring in a formal language*[14].

A different point of view is that AI should construct systems that acquire their knowledge via data observation. More useful than programming hand-coded rules in an AI agent, we should enable that agent to extract patters from the complexity of data. This is the **machine learning approach**. One extension of this view is called **deep learning** where we learn i) how to map a representation of the data to a desirable output and ii) how to represent the data.

After this brief presentation we can state our objectives more clearly: here we will investigate how to build a non-task-oriented dialog system using a deep learning family of models. This family of models is know as **Recurrent Neural Networks** (RNNs). RNNs have achieve increasing success in a variety of NLP tasks such as machine translation, speech recognition, sentiment analysis, language modeling, etc. Together with this success the community has proposed also a variety of new architectures. We will explore some of these architectures for the task of dialog generation.

1.1.1 Theoretical framework

1.1.2 Machine learning

In particular for NLP it is usual to encode a linguistic feature as a dense vector. Linguistic features tent to be discrete entities (a word, a part-of-speech tag, etc.). So for some time the basic tool in the field was the use of 'one-hot vectors', i.e, a vector with only one entry being 1 and all the rest being 0. In this case each feature is its own dimension. The problem with that approach is that features become completely independent from one another. After the popularization of embeddings techniques [15] nowadays we let a model learn the best representation of a linguistic feature in some feature space in order to capture some correlation by the data.

1.1.3 Neural network

A neural network is a non-linear function $f(\boldsymbol{x}; \theta)$. It is defined by a collection of parameters $\boldsymbol{\theta}$ and a collection of non-linear transformations. It is usual to represent f as a compositions of functions:

$$f(\boldsymbol{x};\theta) = f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{W}_1,\boldsymbol{b}_1);\boldsymbol{W}_2,\boldsymbol{b}_2) \tag{1.1}$$
$$= softmax(\boldsymbol{W}_2(\sigma(\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1))+\boldsymbol{b}_2) \tag{1.2}$$

The output of these intermediary functions are referred as *layers*. So in the example above, $\boldsymbol{x}$ (the output of the identity function) is the *input layer*, $f^{(1)}(\boldsymbol{x};\boldsymbol{W}_1,\boldsymbol{b}_1)$ is the *hidden layer* and $f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{W}_1,\boldsymbol{b}_1);\boldsymbol{W}_2,\boldsymbol{b}_2)$ is the *output layer*. Since each layer is a vector, we normally speak about the *dimension* of a layer. For historical reasons we also say that each entry on a layer is a *node* or a *neuron*. Models with a large number of hidden layers are called *deep models*, for this reason the name *deep learning* is used.

A neural network is a function approximator: it can approximate any Borel measurable function from one finite dimensional space to another with any desired nonzero amount of error. This theoretical result is know as the *universal approximation theorem*[3]. Without entering in the theoretical concepts, it suffice to note that the family of Borel mensurable functions include all continuous functions on a closed and bounded subset of $\mathbb{R}^n$.

One thing to be noted is that the universal approximation theorem states that a neural network with at last one hidden layer with any 'squashing' activation function (such as the logistic sigmoid activation function) can be used to approximate the desired function. So it is natural to question why should anybody use a neural network with more than one hidden layer. The answer is that the theorem does not guarantee that a training algorithm will find the correct function generating on the data that we use for the training. Also a neural network with only one layer can use a hidden size of large dimension to represent one function, when a deep model can have more layers but with significantly smaller dimension. The main result in [23] makes this problem clear: for every positive integer k, there exist neural networks with $\Theta(k^3)$ layers, $\Theta(1)$ nodes per layer, and $\Theta(1)$ distinct parameters which can not be approximated by networks with $\mathcal{O}(k)$ layers and $o(2^k)$ nodes.

> Any time we choose a specific machine learning algorithm, we are implicitly stating some set of prior beliefs we have about what kind of function the algorithm should learn. Choosing a deep model encodes a very general belief that the function we want to learn should involve composition of several simpler functions. This can be interpreted from a representation learning point as

> saying that we believe the learning problem consist of discovering a set of underlying factors of variation that van in turn be described in terms of other, simpler underlying factors of variation. Alternately, we can interpret the use of a deep architecture as expressing a belief that the function we want to learn id a computer program consisting of multiple steps, where each step makes use of the previous step's output. These intermediate outputs are not necessarily factors of variation but can instead be analogous to counters or pointers that the network uses to organize its internal processing.[5, p. 195].

1.2 Intuit introduction

One of the main goals in Natural Language Processing is to build agents capable of *understanding language and reasoning*, i.e., agents capable of carrying a conversation as if they were human. This goal is as old as the field of *artificial intelligence* itself [24], and it is a very ambitious one. A smart research strategy is to break this problem in smaller versions of itself. So nowadays we divide dialog systems in two categories: *goal-driven systems* and *non-goal-driven systems*; the former includes *chatbots*, normally used in the industry for technical support services or information acquisition; the latter is a term used to refer to any conversational agent with no explicit purpose.

Although non-goal-driven systems may seen interesting for its comprehensiveness, it presents a central problem: *there is no good quantitative metric to compare non-goal-driven agents* [9, 11]. A more fruitful approach is the one from the goal-driven systems literature: developed a series of synthetic tasks in the form of question answering (QA) to test different capabilities of the competing models [1, 7, 26]. Each task tries to assert one prerequisite to full language understanding.

Here we propose to expand the work done in [1, 26] by *adding new testbeds for complex semantic relationships*: **Entailment-QA**. The motivation behind this expansion is to guarantee that an end-to-end machine learning model can perform *complex linguistic inferences*. We focus on two kind of inferences: the ones defined by *logical operators* and others defined by *word knowledge*.

1.3 Motivation

sdasdasd

1.4 Objectives

dasdadsdad

1.5 Organization

asdsadsd

Chapter 2

Background

Different deep learning architectures are used in NLP: **convolutional architectures** have a good performance in tasks were it is required to find a linguistic indicator regardless of its position (e.g., document classification, short-text categorization, sentiment classification, etc); high quality word embeddings can be achieved with models that are a kind of **feedforward neural network** [15]. But for a variety of works in natural language we want to capture regularities and similarities in a text structure. That is way **recurrent** and **recursive** models have been widely used in the field. Here we are focused on generative models and since recurrent models have been producing very strong results for language modeling [4], we will concentrate on them.

2.1 RNN

Recurrent Neural Network is a family of neural network specialized in sequential data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\tau$. As a neural network, a RNN is a parametrized function that we use to approximate one hidden function from the data. As before we can take the simplest RNN as a neural network with only one hidden layer. But now, what make RNNs unique is a recurrent definition of one of its hidden layer:

$$\boldsymbol{h}^{(t)} = g(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \tag{2.1}$$

$\boldsymbol{h}^{(t)}$ is called *state*, *hidden state*, or **cell**.
Is costumerely to represent a RNN as a ciclic graph 2.1

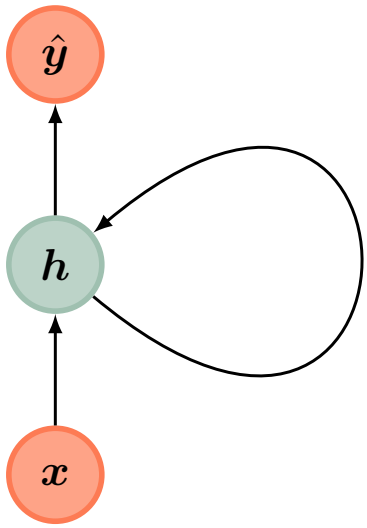


Figure 2.1: Ciclic representation

This recurrent equation can be unfolded for a finite number of steps τ. For example, when $\tau = 3$:

$$\begin{aligned} \boldsymbol{h}^{(3)} &= g(\boldsymbol{h}^{(2)}, \boldsymbol{x}^{(3)}; \boldsymbol{\theta}) && (2.2)\\ &= g(g(\boldsymbol{h}^{(1)}, \boldsymbol{x}^{(2)}; \boldsymbol{\theta}), \boldsymbol{x}^{(3)}; \boldsymbol{\theta}) && (2.3)\\ &= g(g(g(\boldsymbol{h}^{(0)}, \boldsymbol{x}^{(1)}; \boldsymbol{\theta}), \boldsymbol{x}^{(2)}; \boldsymbol{\theta}), \boldsymbol{x}^{(3)}; \boldsymbol{\theta}) && (2.4)\\ & && (2.5) \end{aligned}$$

Hence for any finite step τ we can describe the model as a DAG 2.1.

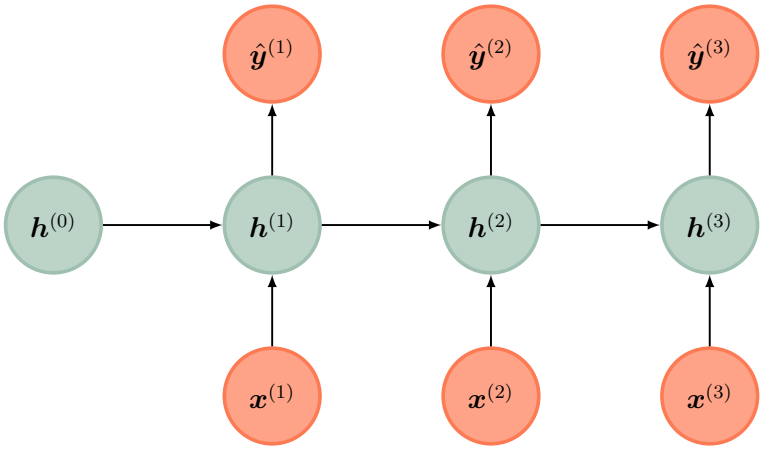


Figure 2.2: Unfolded computational graph

Using a concret example consider the following model define by the equations:

$$f(\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}; \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{U}, \boldsymbol{c}, \boldsymbol{b}) = \hat{\boldsymbol{y}}^{(t)} \tag{2.6}$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax(\boldsymbol{V}\boldsymbol{h}^{(t)} + \boldsymbol{c}) \tag{2.7}$$

$$\boldsymbol{h}^{(t)} = g(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{W}, \boldsymbol{U}, \boldsymbol{b}) \tag{2.8}$$

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{b}) \tag{2.9}$$

Using the graphical representation the model can be view as:

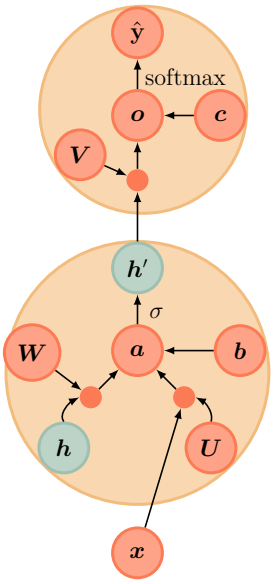


Figure 2.3: Graph of a RNN

!!! explicar como funciona o treinamento !!!
An RNN with a loss function:

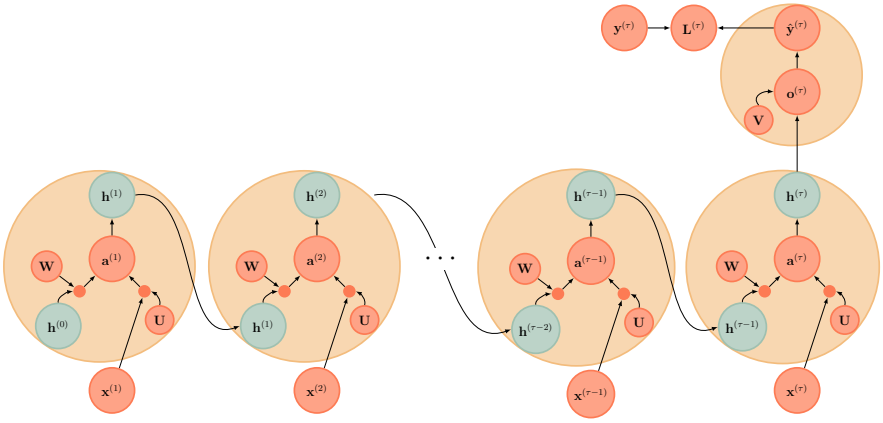


Figure 2.4: The computational graph to compute the training loss of a RNN

2.2 GRU

To capture long-term dependencies on a RNN the authors of the paper [2] proposed a new architecture called **gated recurrent unit** (**GRU**). This model was constructed to make each hidden state $\boldsymbol{h}^{(t)}$ to adaptively capture dependencies of different time steps. It work as follows, at each step t one candidate for hidden state is formed:

$$\widetilde{\boldsymbol{h}}^{(t)} = tahn(\boldsymbol{W}(\boldsymbol{h}^{(t-1)} \odot \boldsymbol{r}^{(t)}) + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{b}) \tag{2.10}$$

where $\boldsymbol{r}^{(t)}$ is a vector with values in $[0, 1]$ called a **reset gate**, i.e., a vector that at each entry outputs the probability of reseting the corresponding entry in the previous hidden state $\boldsymbol{h}^{(t-1)}$. Together with $\boldsymbol{r}^{(t)}$ we define an **update gate**, $\boldsymbol{u}^{(t)}$. It is also a vector with values in $[0, 1]$. Intuitively we can say that this vector decides how much on each dimension we will use the candidate update. Both $\boldsymbol{r}^{(t)}$ and $\boldsymbol{u}^{(t)}$ are defined by $\boldsymbol{h}^{(t-1)}$ and $\boldsymbol{x}^{(t)}$; they also have specific parameters:

$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_r\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r\boldsymbol{x}^{(t)} + \boldsymbol{b}_r) \tag{2.11}$$

$$\boldsymbol{u}^{(t)} = \sigma(\boldsymbol{W}_u\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u\boldsymbol{x}^{(t)} + \boldsymbol{b}_u) \tag{2.12}$$

At the end the new hidden state $\boldsymbol{h}^{(t)}$ is defined by the recurrence:

$$\boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)} \odot \widetilde{\boldsymbol{h}}^{(t)} + (1 - \boldsymbol{u}^{(t)}) \odot \boldsymbol{h}^{(t-1)} \tag{2.13}$$

Note that the new hidden state combines the candidate hidden state $\widetilde{\boldsymbol{h}}^{(t)}$ with the past hidden state $\boldsymbol{h}^{(t-1)}$ using both $\boldsymbol{r}^{(t)}$ and $\boldsymbol{u}^{(t)}$ to adaptively copy and forget information.

It can appear more complex, but we can view the GRU model just as a refinement of the standard RNN with a new computation for the hidden state. Let $\boldsymbol{\theta} = [\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{b}]$, $\boldsymbol{\theta}_u = [\boldsymbol{W}_u, \boldsymbol{U}_u, \boldsymbol{b}_u]$ and $\boldsymbol{\theta}_r = [\boldsymbol{W}_r, \boldsymbol{U}_r, \boldsymbol{b}_r]$; and $\text{aff}(\boldsymbol{\theta})$ be the following operation:

$$\text{aff}(\boldsymbol{\theta}) = \boldsymbol{W}\boldsymbol{h} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{b} \tag{2.14}$$

With similar definitions for $\text{aff}(\boldsymbol{\theta}_u)$ and $\text{aff}(\boldsymbol{\theta}_r)$. Figure 2.3 shows the hidden state of the GRU model for time step t. If compared with Figure 2.1 we can see that the basic structure is the same, just the way of computing the hidden state $\boldsymbol{h}^{(t)}$ has changed.

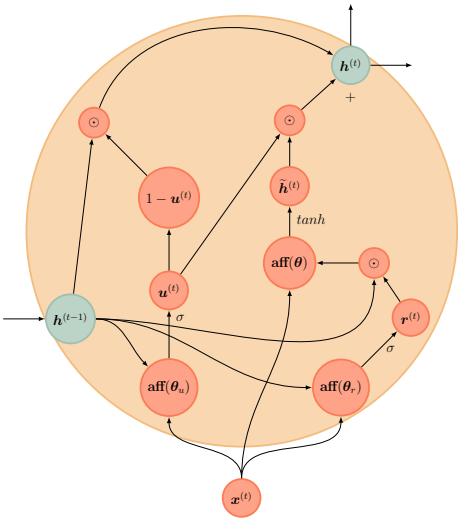


Figure 2.5: GRU hidden cell

2.3 LSTM

Long short-term memory (LSTM) is one of the most applied versions of the RNN family of models. Historically it was developed before the GRU model, but conceptually we can think in the RNN as an expansion of the model presented in the last session. Because of notation differences they can look different. LSTM is based also with parametrized gates; in this case three: the **forget gate**, $\boldsymbol{f}^{(t)}$, the **input gate**, $\boldsymbol{i}^{(t)}$, and the **output gate**, $\boldsymbol{o}^{(t)}$. There gates are defined only by $\boldsymbol{h}^{(t-1)}$ and $\boldsymbol{x}^{(t)}$ with specific parameters:

$$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f) \tag{2.15}$$

$$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i) \tag{2.16}$$

$$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o) \tag{2.17}$$

Intuitively $\boldsymbol{f}^{(t)}$ should control how much informative be discarded, $\boldsymbol{i}^{(t)}$ controls how much information will be updated, and $\boldsymbol{o}^{(t)}$ controls how munch each component should be outputted. A candidate cell, $\tilde{\boldsymbol{c}}^{(t)}$ is formed:

$$\tilde{\boldsymbol{c}}^{(t)} = tahn(\boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{b}) \tag{2.18}$$

and a new cell $\tilde{\boldsymbol{c}}^{(t)}$ is formed by forgetting some information of the previous cell $\tilde{\boldsymbol{c}}^{(t-1)}$ and by adding new values from $\tilde{\boldsymbol{c}}^{(t)}$ (scaled by the input gate)

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \otimes \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \otimes \tilde{\boldsymbol{c}}^{(t)} \tag{2.19}$$

The new hidden state, $\boldsymbol{h}^{(t)}$, is formed by filtering $\boldsymbol{c}^{(t)}$:

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \otimes tanh(\boldsymbol{c}^{(t)}) \tag{2.20}$$

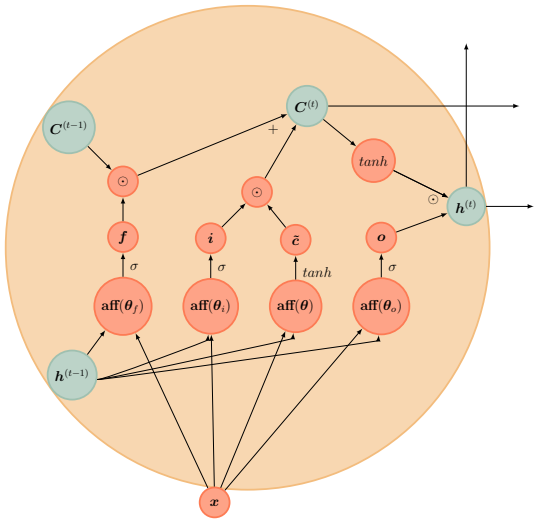


Figure 2.6: LSTM hidden cell

2.4 Language model

We call *language model* a probability distribution over sequences of tokens in a natural language.

$$P(x_1, x_2, x_3, x_4) = p$$

This model is used for different nlp tasks such as speech recognition, machine translation, text auto-completion, spell correction, question answering, summarization and many others.

The classical approuch to a languange model was to use the chain rule and a markovian assumptiom, i.e., for a specific n we assume that:

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t|x_1, \ldots, x_{t-1}) = \prod_{t=1}^{T} P(x_t|x_{t-(n+1)}, \ldots, x_{t-1}) \quad (2.21)$$

This gave raise to models based on n-gram statistics. The choice of n yields different models; for example *Unigram* language model ($n = 1$):

$$P_{uni}(x_1, x_2, x_3, x_4) = P(x_1)P(x_2)P(x_3)P(x_4) \quad (2.22)$$

where $P(x_i) = count(x_i)$.

Bigram language model ($n = 2$):

$$P_{bi}(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_3) \quad (2.23)$$

where

$$P(x_i|x_j) = \frac{count(x_i, x_j)}{count(x_j)}$$

Higher n-grams yields better performance. But at the same time higher n-grams requires a lot of memory[6].

Since [16] the approach has change, instead of using one approach that is specific for the language domain, we can use a general model for sequential data prediction: a RNN.

So, our learning task is to estimate the probability distribution

$$P(x_n = \text{word}_{j^*}|x_1, \ldots, x_{n-1})$$

for any $(n-1)$-sequence of words $x_1, \ldots, x_{n-1}$.

We start with a corpus C with T tokens and a vocabulary $\mathbb{V}$.

Example: **Make Some Noise** by the Beastie Boys.

Yes, here we go again, give you more, nothing lesser
Back on the mic is the anti-depressor
Ad-Rock, the pressure, yes, we need this
The best is yet to come, and yes, believe this
...

- $T = 378$
- $|\mathbb{V}| = 186$

The dataset is a collection of pairs $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}$ is one word and $\boldsymbol{y}$ is the immediately next word. For example:

$(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)})$ = (Yes, here).

$(\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)})$ = (here, we)

$(\boldsymbol{x}^{(3)}, \boldsymbol{y}^{(3)})$ = (we, go)

$(\boldsymbol{x}^{(4)}, \boldsymbol{y}^{(4)})$ = (go, again)

$(\boldsymbol{x}^{(5)}, \boldsymbol{y}^{(5)})$ = (again, give)

$(\boldsymbol{x}^{(6)}, \boldsymbol{y}^{(6)})$ = (give, you)

$(\boldsymbol{x}^{(7)}, \boldsymbol{y}^{(7)})$ = (you, more)

...

Notation

- $\boldsymbol{E} \in \mathbb{R}^{d,|\mathbb{V}|}$ is the matrix of word embeddings.
- $\boldsymbol{x}^{(t)} \in \mathbb{R}^{|\mathbb{V}|}$ is one-hot word vector at time step t.
- $\boldsymbol{y}^{(t)} \in \mathbb{R}^{|\mathbb{V}|}$ is the ground truth at time step t (also an one-hot word vector).

The language model is similar as the RNN described above. It is defined by the following equations:

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)} \tag{2.24}$$

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{e}^{(t)} + \boldsymbol{b}) \tag{2.25}$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax(\boldsymbol{V}\boldsymbol{h}^{(t)} + \boldsymbol{c}) \tag{2.26}$$

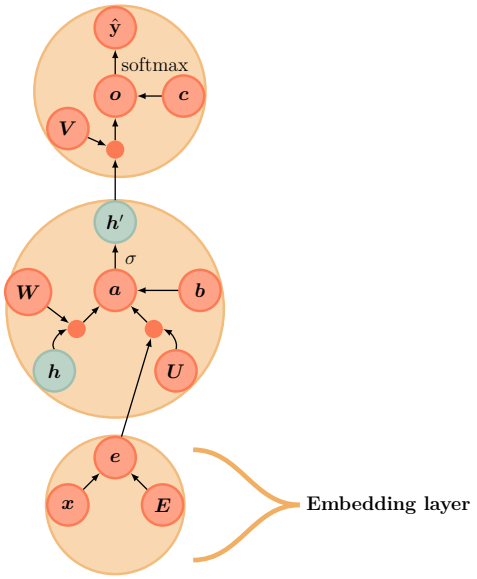


Figure 2.7: Simple language model

At each time t the point-wise loss is:

$$\begin{aligned} L^{(t)} &= CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) && (2.27)\\ &= -\log(\hat{\boldsymbol{y}}_{j^*}) && (2.28)\\ &= -\log P(x^{(t+1)} = \text{word}_{j^*} | x^{(1)}, \ldots, x^{(t)}) && (2.29) \end{aligned}$$

The loss L is the mean of all the point-wise losses

$$L = \frac{1}{T}\sum_{t=1}^{T} L^{(t)} \tag{2.30}$$

Evaluating a language model. We can evaluate a language model using a *extrinsic evaluation*: How our model perform in a NLP task such as text auto-completion. Or a *intrinsic evaluation*: Perplexity (PP) can be thought as the weighted average branching factor of a language.

Given $C = x_1, x_2, \ldots, x_T$, we define the perplexity of C as:

$$PP(C) = P(x_1, x_2, \ldots, x_T)^{-\frac{1}{T}} \tag{2.31}$$

$$\tag{2.32}$$

$$= \sqrt[T]{\frac{1}{P(x_1, x_2, \ldots, x_T)}} \tag{2.33}$$

$$\tag{2.34}$$

$$= \sqrt[T]{\prod_{i=1}^{T} \frac{1}{P(x_i|x_1, \ldots, x_{i-1})}} \tag{2.35}$$

we can relate Loss and Perplexity:

Since

$$L^{(t)} = -\log P(x^{(t+1)}|x^{(1)}, \ldots, x^{(t)}) \tag{2.36}$$

$$= \log(\frac{1}{P(x^{(t+1)}|x^{(1)}, \ldots, x^{(t)})}) \tag{2.37}$$

$$\tag{2.38}$$

We have that:

$$L = \frac{1}{T}\sum_{t=1}^{T} L^{(t)} \tag{2.39}$$

$$= \log\left(\sqrt[T]{\prod_{i=1}^{T} \frac{1}{P(x_i|x_1, \ldots, x_{i-1})}}\right) \tag{2.40}$$

$$= \log(PP(C)) \tag{2.41}$$

So another definition of perplexity is

$$2^L = PP(C) \tag{2.42}$$

2.5 Seq2seq

fsdfdsfdsf

2.6 Atention

fksdhfjsdgjf

Chapter 3

Dialog Systems

3.1 How to evaluate dialogs?

BLUE

This metric was proposed in [18] for automatic translation. It compares n-grams (up to 4) of the candidate translation with the n-grams of the reference translation and count the number of matches; it also ads brevity penalty for too short translations. The formula for this metrics is:

$$BLUE = min\left(1, \frac{output - length}{reference - lenght}\right)\left(\prod_{n=1}^{4} precision_n\right)^{\frac{1}{4}} \tag{3.1}$$

Where $precision_n$ is number of n-gram overlap between the candidate and the reference divided by the number of all n-grams in the candidate. BLUE scores ranges from 0 to 1. Typically this score is computed over an entire corpus and was originally designed for use with multiple reference sentences. To give one simple example we will use the following Portuguese sentence:

'em plano aberto, a cidade parece linda'

The reference translation is

'in a wide shot, the city looks beautiful'

Now consider two candidates:

Metric	c_1	c_2
$precision_1$	5/7	4/7
$precision_2$	3/6	2/6
$precision_3$	2/5	1/5
$precision_4$	1/4	0/4
brevity penalty	7/8	7/8
$BLUE$	0.38	0

c_1: 'in the open, the city looks beautiful'
c_2: 'in open plan, the city looks gorgeous'

Table 3.1 shows the precision for each n-gram (up to 4) and the BLUE score for each candidate.

3.2 Entailment-QA

An intelligent agent should distinguish between *meaningful* and *nonsensical* speech. To do that the agent can make use of background knowledge, but it can also point out *gaps in the speech's rationality*. Logic is the study of reasoning. Over the time it became a complex discipline with its own concepts, tools and language. Here we will use some logical notions like *entailment* and *contradiction* to build a set of synthetic tasks. All tasks presented are focused on the distinction between *sound* and *unsound* speech. The difference between tasks resides in the use of certain semantic structures.

In [26], among a variety of useful tasks the authors introduce two tasks that deals with logical inference: *basic deduction* and *basic induction*. The basic deduction task offers questions of the form:

P^1 are afraid of Q^1
P^2 are afraid of Q^2
P^3 are afraid of Q^3
P^4 are afraid of Q^4
c^1 is a P^1
c^2 is a P^2
c^3 is a P^3
c^4 is a P^4

What is c^j afraid of? A: P^j

where P^j and Q^j are animals (e.g., "cats", "mice", "wolfs", etc.) and c^j are names (e.g., "Jessica", "Gertrud", "Emily", etc.). The underlying relation under focus here is the *membership* relation: if Emily is a cat and cats are afraid of wolfs then Emily is afraid of wolfs. This task is solvable by the current models, in [26] the best accuracy for this task is 100%.

The basic induction task is composed by questions of the form:

c^1 is a P^1
c^1 is C^1
c^2 is a P^2
c^2 is C^2
c^3 is a P^3
c^3 is C^3
c^4 is a P^4
c^4 is C^4
c is a P^j
What color is c? A: C^j

Where P^j is a animal, C^j is a color, and c^j is a name. Here the agent is asked to remember the relation between animal and color, and when presented a new name of an animal in the example, the agent should infer the color using past examples. Similar to the deduction task, in [26] is reported that this task is completely solved.

These two tasks present a first step towards *a complete set of tasks to tests inference capabilities*. One aspect that is missing though is the use of logical operators such as *boolean connectives* and *first-order quantifiers*. Those operators play a big role on everyday speech, hence a natural way of expanding this work is by creating a set of tasks that make agents learn *valid logic structures*.

To highlight the speech structure we will make use of an artificial language, but, to be clear, this is just an exposition tool. We are concerned in logical structures *only used in everyday speech*.

Boolean Connectives The first task is focused only on some propositional connectives $\wedge$ (and), $\vee$ (or), $\neg$ (not). The agent is given two sentences s_1 and s_2, and he is asked if s_1 entails s_2 or not (a yes/no question). The position of the sentences is important here. We look at only six general cases:

- Entailment
  - $\underbrace{P^1a^1 \wedge \cdots \wedge P^na^n}_{s_1}, \underbrace{P^ja^j}_{s_2}$
  - $\underbrace{P^ja^j}_{s_1}, \underbrace{P^1a^1 \vee \cdots \vee P^na^n}_{s_2}$
  - $\underbrace{Pa}_{s_1}, \underbrace{\neg\neg Pa}_{s_2}$

- Not entailment
  - $\underbrace{P^ja^j}_{s_1}, \underbrace{P^1a^1 \wedge \cdots \wedge P^na^n}_{s_2}$
  - $\underbrace{P^1a^1 \vee \cdots \vee P^na^n}_{s_1}, \underbrace{P^ja^j}_{s_2}$
  - $\underbrace{Pa}_{s_1}, \underbrace{\neg Pa}_{s_2}$

Where P is a predicate and a is a name. The point here is not to demand that the agent learn complex logical forms, but to recognize which forms are sound and which are not. This should be achieved independently from the content, i.e., the different predicates and names that appear on the sentences. So from the abstract forms above we have only simple examples like:

Ashley is fit

Ashley is not fit

The first sentence implies the second sentence? A: no

Avery is nice and Avery is obedient

Avery is nice

The first sentence implies the second sentence? A: yes

Elbert is handsome or Elbert is long

Elbert is handsome

The first sentence implies the second sentence? A: no

First-Order Quantifiers Task 2 tries to capture the use of some basic *quantifiers*. In this dataset we are trying to predict the entailment relationship between two sentences s_1 and s_2. We say that there is a *entailment* relationship if s_1 implies s_2, we say that there is a *contradiction* if the combination of sentences s_1 and s_2 implies an absurdity and if the combination of sentences does not imply an absurdity we say that they are *neutral*. We have used six forms of logical relations for the quantifiers $\forall$ (for every) and $\exists$ (exists):

- Entailment
  - $\forall x Px, Pa$
  - $Pa, \exists x Px$
- Contradiction
  - $\forall x Px, \neg Pa$
  - $\forall x Px, \exists x \neg Px$
- Neutral
  - Pa, Qa
  - $\forall x Px, \neg Qa$

where P and Q are non-related predicates and a is a name. So we have examples like:

Every person is lively

Belden is lively

What is the semantic relation? A: entailment

Every person is short

There is one person that is not short

What is the semantic relation? A: contradiction

Every person is beautiful

Abilene is not blue

What is the semantic relation? A: neutral

The tasks above force the dialog system to predict entailment independently of the specific meaning of nouns, verbs and adjectives presented on speech. To balance that we intend to design four more tasks centered on *generic semantic knowledge*.

Synonymy This task tests if a dialog system can identify paraphrase caused by synonym use. Two supporting facts are presented, the second differs from the first by one noun, verb or adjective that can be a synonym or not, e.g., "*The girl is talking into the microphone. The girl is speaking. Are the above sentences duplicate?*".

Antinomy This task consists of recognizing contradictions by antinomy use. For example, the question "*John is not an old person. John is young. Are the above sentences a contradiction?*" should be answered "no" and the question "*Susan is happy. Susan is sad and she is crying. Are the above sentences a contradiction?*" should be answered "yes".

Hypernymy When one term is a specific instance of another we say that there is a hypernymy relationship between then. For example, we say that "anthem" is a hyponym and "song" a hypernym, because anthem is a kind of song. Task 5 tests the kind of linguistic entailment caused by the use of hyponyms and hypernyms, e.g., "*A woman is eating an apple. A woman is eating a fruit. Are the above sentences duplicate?*"

Active/Passive voice Finally the last task is centered on the paraphrases originated by the changes from active to passive voice. So two supporting facts are presented, although they have a different syntactic structure, they share the same meaning. For example, "*A man is playing the piano. The piano is being played by a man. Are the above sentences duplicate?*".

It should be noted that we can formulate the notion of paraphrase as an entailment relation: if s_1 is a paraphrase of s_2, it is natural to say that s_1 implies s_2 and vice-versa. This is done in [13]. Although this approach presents no problem it can alienate some people from the machine learning

community: this community normally deals with problems of similarity between sentences, there is very little datasets available centered on the notion of entailment. So although we have mentioned only paraphrase detection in tasks 3–6 the entailment relationship is present.

3.3 Approach

We will only consider *neural network based end-to-end dialog systems*, these are the most important models today [1, 10, 19, 20, 21, 25]. To organize all experiment in an unified framework we decided to use the platform ParlAI [17]. Our criteria for a *semantic robust dialog agent* is not an agent that is only tuned for the Entailment-QA tasks mentioned above, but an agent that performs well across all established tasks (like the bAbI tasks [26]) and performs reasonably on the Entailment-QA tasks 1-6. Here "reasonably" means that the agent should exceed a random agent by a significant margin.

Right now we are in the process of creating the dataset. We have already built tasks 1 (boolean connectives) and 2 (first order quantifiers). They both are composed of 10000 questions for training and 1000 for testing.

We have used the dataset SICK (Sentence Involving Compositional Knowledge) [13] as a proxy for tasks 3-6, since all the structures presented on those tasks are presented in this dataset. The SICK data is composed of a pairs of sentences and a label describing the entailment relation between the sentences. To cast this classification dataset as a question answering problem we have added a question and maintained the labels:

A group of people are marching

A group of people are walking

What is the semantic relation? A: entailment

There is no dog leaping in the air

A dog is leaping high in the air and another is watching

What is the semantic relation? A: contradiction

A man is exercising

A baby is laughing

What is the semantic relation? A: neutral

Some dogs are playing in a river

Some dogs are playing in a stream

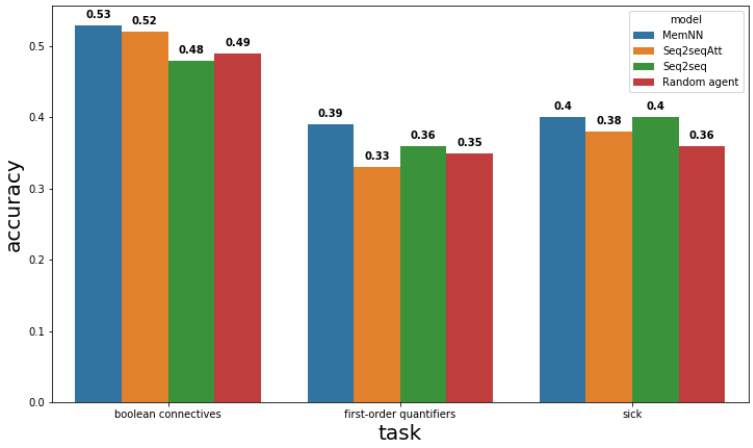
What is the semantic relation? A: entailment

This QA task is composed of 23000 questions for training and 5900 for testing.

3.4 Preliminary results

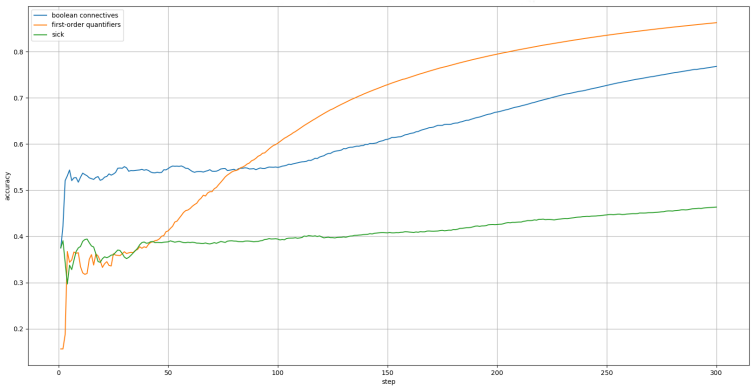
We have performed the first experiments using the sequence to sequence model [22](with and without attention: Seq2seq and Seq2seqAtt, respectively) and the memory network model (MenNN) [27].

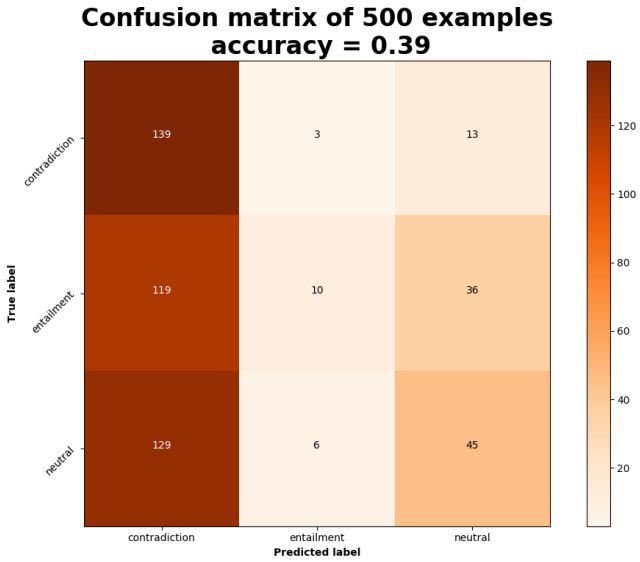
So far these models show unsatisfactory results, as can be seen in Figure 1.



Their overall performance is only slightly better when compared to the random agent. For example, when we look at the memory model, the model that often outperform the seq2seq model on QA tasks [26], we can see that regarding tasks 1 and 2 there is an overfitting problem: the model shows high accuracy on the train dataset (75% accuracy on task 1, and 86% accuracy

on task 2) as can be seen in Figure 2. But when we take a closer look at the confusion matrix of the test data, Figure 3, the results are not impressive.[1]





[1]All the experiments and the respective results are available on GitHub: `https://github.com/felipessalvatore/DialogGym`.

Chapter 4

Project Methodology

science is good

4.1 Work Plan

writing papers is good

Chapter 5

Conclusion

The results so far may seen grim, but they show also a positive side: *the Entailment-QA tasks are not a set of trivial tasks that can be completely solved by the current models*. These results indicated that it is worthwhile to explore this set of task with more detail, either by improving the training using the current models or by exploring new kinds of models. One thing should be clear, we are not concern in obtaining high accuracy on these tasks only (just as a comparison, in [8] is reported an accuracy of 87% on the SICK dataset by using feature engineering techniques). The main idea is to use an end-to-end QA model to obtain good results in the Entailment-QA task without compromising the accuracy on other well established QA tasks.

Future work will explore two branches: on the one hand, we need to finish the Entailment-QA corpus to have a fine grain analysis of the result that we are seeing on the SICK corpus; on the other hand, we need to explore the different extensions for all mentioned models [12, 20] before inferring any kind of limitation of the current set of QA models.

Bibliography

[1] A. Bordes and J. Weston. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683, 2016.

[2] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[3] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2:303–314, 1989.

[4] Y. Goldberg. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726, 2015.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2017.

[6] K. Heafield. Scalable modified kneser-ney language model estimation. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[7] B. Hixon, P. Clark, and H. Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[8] A. Lai and J. Hockenmaier. Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334. Association for Computational Linguistics, 2014.

[9] C. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *CoRR*, abs/1603.08023, 2016.

[10] R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. *CoRR*, abs/1708.07149, 2017.

[11] R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. On the evaluation of dialogue systems with next utterance classification. *CoRR*, abs/1605.05414, 2016.

[12] F. Ma, R. Chitta, S. Kataria, J. Zhou, P. Ramesh, T. Sun, and J. Gao. Long-term memory networks for question answering. *CoRR*, abs/1707.01961, 2017.

[13] M. Marelli, L. Bentivogl, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval 2014*, 2014.

[14] J. McCarthy and P. J. Hayes. Readings in nonmonotonic reasoning. chapter Some Philosophical Problems from the Standpoint of Artificial Intelligence, pages 26–45. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[16] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur. Extensions of recurrent neural network language. *IEEE*, pages 5528–5531, 2011.

[17] A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. Parlai: A dialog research software platform. *CoRR*, abs/1705.06476, 2017.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In *ACL '02: Proceedings of*

the 40th Annual Meeting on Association for Computational Linguistics, pages 311–318. Association for Computational Linguistics, 2001.

[19] I. V. Serban, R. Lowe, L. Charlin, and J. Pineau. Generative deep neural networks for dialogue: a short review. *CoRR*, abs/1611.06216, 2016.

[20] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference of Artificial Intelligence*, AAAI'16, pages 3776–3783. AAAI Press, 2016.

[21] Y. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2200–2209, 2017.

[22] I. Sustskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, 2014.

[23] M. Telgarsky. Benefits of depth in neural networks. *CoRR*, abs/1602.04485, 2016.

[24] A. Turing. Computing machinery and inteligence. *Mind*, pages 433–460, 1950.

[25] T. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. J. Young. A network-based end-to-end trainable task-oriented dialogue system. *CoRR*, abs/1604.04562, 2016.

[26] J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[27] J. Weston, S. Chopra, and A. Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.