“Linguistic and Cognitive Approaches to Dialog Agents”
(LaCATODA 2018)

July 13, 2018
Stockholm, Sweden
# Table of Contents

Sahil Garg, Guillermo Cecchi, Irina Rish, Shuyang Gao, Greg Ver Steeg, Sarik Ghazarian, Palash Goyal and Aram Galstyan  
*Dialogue Modeling Via Hash Functions*  

Daniel Harborne, Dave Braines, Alun Preece and Rafal Rzepka  
*Conversational Control Interface to Facilitate Situational Understanding in a City Surveillance Setting*  

Szymon Malik, Adrian Lancucki, Jan Chorowski  
*Efficient Purely Convolutional Text Encoding*  

Masahiro Mizukami, Hiroaki Sugiyama and Hiromi Narimatsu  
*Event Data Collection for Recent Personal Questions*  

Hiromi Narimatsu, Hiroaki Sugiyama, Masahiro Mizukami  
*Detecting Location-Indicating Phrases in User Utterances for Chat-Oriented Dialogue Systems*  

Maria Skeppstedt and Magnus Ahltorp  
*Towards a structured evaluation of improv-bots: Improvisational theatre as a non-goal-driven dialogue system*  

Yuiko Tsunomori, Ryuichiro Higashinaka and Takeshi Yoshimura  
*Refinement of utterance database and concatenation of utterances for enhancing system utterances in chat-oriented dialogue system*  

Rui Zhao and Volker Tresp  
*Improving Goal-Oriented Visual Dialog Agents via Advanced Recurrent Nets with Tempered Policy Gradient*
Dialogue Modeling Via Hash Functions

Sahil Garg\textsuperscript{1}, Guillermo Cecchi\textsuperscript{2}, Irina Rish\textsuperscript{2}, Shuyang Gao\textsuperscript{1}, Greg Ver Steeg\textsuperscript{1}, Sarik Ghazarian\textsuperscript{1}, Palash Goyal\textsuperscript{1}, Aram Galstyan\textsuperscript{1}

\textsuperscript{1} US Information Sciences Institute, Marina del Rey, CA USA
\textsuperscript{2} IBM Thomas J. Watson Research Center, Yorktown Heights, NY USA

sahil.garg.cs@gmail.com, \{gcecchi.rish\}@us.ibm.com, sgao@isi.edu, \{gregv,ghazarian,goyal,galstyan\}@isi.edu

Abstract

We propose a novel machine-learning framework for dialogue modeling which uses representations based on hash functions. More specifically, each person’s response is represented by a binary hashcode where each bit reflects presence or absence of a certain text pattern in the response. Hashcodes serve as compressed text representations, allowing for efficient similarity search. Moreover, hashcode of one person’s response can be used as a feature vector for predicting the hashcode representing another person’s response. The proposed hashing model of dialogue is obtained by maximizing a novel lower bound on the mutual information between the hashcodes of consecutive responses. We apply our approach in psychotherapy domain evaluating its effectiveness on a real-life dataset consisting of therapy sessions with patients suffering from depression; in addition, we also model transcripts of interview sessions between Larry King (television host) and his guests.

1 Introduction

Dialogue modeling and generation is an area of active research, and of great practical importance, as it provides a basis for building successful conversational agents in a wide range of applications. While an open-domain dialogue remains a challenging open problem, developing dialogue systems for particular applications can be more tractable, due to specific properties of the application domain.

A motivating application for our work is (semi-)automated psychotherapy: easily accessible, round-the-clock psychotherapeutic services provided by a conversational agent. According to recent estimates, mental health disorders affect one in four adult Americans, one in six adolescents, and one in eight children. Furthermore, as predicted by the World Health Organization, by 2030 the amount of worldwide disability and life loss attributable to depression may become greater than for any other condition, including cancer, stroke, heart disease, accidents, and war. However, many people do not receive an adequate treatment; one of the major factors here is limited availability of mental health professionals, as well as limited availability of mental health professionals, as

Table 1: In this table, we show pairs of patient and therapist (Blue) responses. For each response, we obtain a locality sensitive hashcode as its representation. The hashcode of a patient response is used as feature representation to infer the hashcode of the corresponding therapist response.

<table>
<thead>
<tr>
<th>Blue Response</th>
<th>Hashcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;It's all my fault. It's like getting hit in the stomach. And trying to catch your breath. You can't catch it. You struggle, move around, You just can't get your breath. That just really knocks me off balance. You begin to generate this feeling of a kind of negativism. And that’s why it really being hurt. I never allowed myself to think negative, accept everybody else. I am the only one giving back to the &quot;be quiet time.&quot; I had to have something to make do with that.&quot;</td>
<td>01010011100101111111010110110011011100110111001101011</td>
</tr>
<tr>
<td>&quot;OK. Maybe that fits into the whole pattern then.&quot;</td>
<td>011001100101111111100110011110010110011100110</td>
</tr>
</tbody>
</table>

Motivated by above considerations, we introduce here a novel dialogue modeling framework where responses are represented as locality-sensitive binary hashcodes [Kulis and Grauman, 2009; Joly and Buisson, 2011; Garg et al., 2018b]. The motivation behind such approach includes several con-
procedure, we assign artificial labels to other small subsets of re-
a maximum margin boundary, one can learn a kernel or neural lan-
hash function $h$ for assignment of blue labels. Having these 8 responses with artifi-
senting hash bit value 1, and another 4 responses selected randomly
selected randomly for assignment of artificial red color labels, repre-
top left of the figure, for building a hash function, 4 responses are
In the figure, each dialogue response is represented as a dot. In the
Figure 1: An illustration of the locality sensitive hashing approach.

- a novel, generic framework for modeling a dialogue using
  **locality sensitive hash functions** (based on kernel similarities or neural networks, as discussed later) to rep-
  resent textual responses;

- a **novel lower bound on the Mutual Information (MI)** between the hashcodes of the responses from the two
  agents (capturing relevance between the two), used as an optimization criterion to optimize a hashing model;

- an approach using the MI lower bound as a metric for
  joint evaluation of the representation quality of hash-
  codes, and the quality of inference. Also, a tight upper bound on joint entropy is derived, to separately charac-
  terize the quality of hashcodes as representations.

Empirical evaluation of the proposed framework and the op-
timization approach, which uses both kernels and neural net-
works for locality sensitive hash functions, is performed on a
dataset containing textual transcriptions of 400 depression-
therapy sessions conducted by human therapists with real pa-
ients. We find that our approach of optimizing hashing by
maximizing our novel MI lower bound provides a consistent
improvement over alternative types of hashing approaches
(using both kernels and neural nets), in terms of the proposed
evaluation metrics. Further, general applicability of our work
is demonstrated by experiments on another dataset of trans-
scriptions of 75 interview sessions between television host
Larry King and his guests.

2 Related Work
Dialogue agents, e.g. chat bots, are becoming increasingly
popular in various applications, including (semi)-automated
psychotherapy, for example, based on popular techniques
such as CBT (Cognitive Behavioral Therapy [Lewinsohn et al., 1990]); however, these agents have very limited abil-
ities to actually understand free text responses from their
users; instead, they are typically offering a fixed set of pre-
programmed responses to choose from [Fitzpatrick et al., 2017]. See [Jurafsky and Martin, 2014] for an overview.

There are several neural networks based approaches pro-
posed in the recent past for dialogue modeling in general
domain [Serban et al., 2015; 2016; 2017a; 2017b; Asghar et al., 2017; Wu et al., 2017]. However, the setting is somewhat different from the therapy dialogues
where the patient responses can be extremely long (up to tens
of thousands of words). Also, evaluating the effectiveness of
the therapist’s response requires some notion of relevance
which goes beyond the standard measures of its semantic fea-
tures [Papineni et al., 2002; Liu et al., 2016; Li and Jurafsky, 2016; Lowe et al., 2017; Li et al., 2017]; we consider here an
information-theoretic approach to capture this notion. Also,
therapy dialogue modeling and generation has some similar-
ities with the task-driven dialogue [Zhai and Williams, 2014;
Wen et al., 2016; Althoff et al., 2016; Lewis et al., 2017;
He et al., 2017], although evaluating effectiveness of a thera-
peutic dialogue may be more challenging as the effect is not
always immediate. Attention to specific parts of the response,

While the above approach was motivated by psychotherapy
domain, it is generally applicable to a wide range of other do-
mains involving dialogue modeling and generation. The **key contribu-
tions** of this work include:

- a novel, generic framework for modeling a dialogue us-
  ing **locality sensitive hash functions** (based on kernel
  similarities or neural networks, as discussed later) to rep-
  resent textual responses;

- an approach using the MI lower bound as a metric for
  joint evaluation of the representation quality of hash-
  codes, and the quality of inference. Also, a tight upper bound on joint entropy is derived, to separately charac-
  terize the quality of hashcodes as representations.

![Figure 1: An illustration of the locality sensitive hashing approach.](image-url)
3 Dialogue Modeling via Binary Hashing

Here we discuss our novel framework for modeling a dialogue between two agents using binary hash functions. In our formulation below, we refer to the two agents as a patient and a therapist, respectively, although the approach can be clearly applied to a variety of other dialogue settings.

3.1 Problem Formulation

In a dialogue session between a patient and a therapist, we denote \( i_{th} \) response from the patient as \( S_{i}^{pt} \), and the corresponding response from the therapist as \( S_{i}^{t} \); in a session, we have \( S^{pt} = \{ S_{1}^{pt}, \ldots, S_{N}^{pt} \} \), with additional notations for sets, \( S = \{ S_{1}^{pt}, \ldots, S_{N}^{pt}, S_{1}^{t}, \ldots, S_{N}^{t} \} \), \( S^{t} = \{ S_{1}^{t}, \ldots, S_{N}^{t} \} \). In this work, we consider a response \( S_{i} \), be it from the patient or the therapist in a session, as a natural language structure which can be simply plain text, or text along with part of speech tags (PoS), or syntactic/semantic parsing of the text. As per a typical dialogue modeling setup, the task would be to predict \( S_{i}^{t} \) given \( S_{i}^{pt} \), i.e. generating therapist responses. However, we propose a different setup. We propose to encode each response of a patient/therapist \( S_{i}^{pt} \) in a therapy session as a binary hashcode \( (c_{i}^{pt}, c_{i}^{t}) \), and focus upon the problem of inferring the therapist hashcode \( (c_{i}^{t}) \) given the hashcode \( (c_{i}^{pt}) \) of the patient response \( S_{i}^{pt} \), before finally mapping the inferred therapist hashcode \( (c_{i}^{t}) \) to multiple text response choices. Note that \( c_{i}^{t} \) is the hashcode representation of the ground truth therapist response \( S_{i}^{t} \) (so it is the groundtruth itself), whereas \( c_{i}^{pt} \) is the inference given only the knowledge of \( S_{i}^{pt} \), its hashcode \( c_{i}^{pt} \), and no knowledge of \( S_{i}^{t} \); all the hashcodes are generated using the same hashing model \( \mathcal{H} \). See Tab. 1 for examples of pairs of patient/therapist responses and their corresponding hashcode representations.

The idea of maximizing mutual information objective for dialogue modeling has been considered previously by [Li et al., 2015], though it was only used in test setting, rather than as a guide for training.

An evaluation metric such as BLEU score [Papineni et al., 2002] may not be the best for our application, it tries to capture all information in text when comparing an inference w.r.t. the ground truth, rather than evaluate the relevance of one response to the other. We propose several evaluation metrics based on binary hashcodes themselves, since the latter are assumed to serve as efficient representations of text in our dialogue context.

3.2 Locality Sensitive Hashcodes of Responses using Kernel Functions or Neural Language Models

The main idea behind locality sensitive hashing is that data points that are similar to each other should be assigned hashcodes with minimal Hamming distance to each other, and vice versa. A similarity/distance function need not to be defined explicitly; see [Wang et al., 2014] for details.

Since locality sensitive hashing ensures that natural language structures, that are assigned hashcodes with low hamming distance to each other, are similar to each other, locality sensitive hashcodes should serve as generalized representations of language structures (a similarity/distance function implied as per the locality sensitive hashing model learned for a given task) \(^1\), and so for the responses in a dialogue.

Many hash functions have been proven to be locality sensitive, with rigorous mathematical proofs [Wang et al., 2017]. However, the literature on data driven locality sensitive hash functions is recent, such as based on kernel similarity functions [Kulis and Grauman, 2009; Joly and Buisson, 2011; Garg et al., 2018b]; see Fig. 1 for a demonstration of locality sensitive hashing of dialogue responses. Although these

---

\(^1\)A similarity function doesn’t imply a semantic similarity function here. For instance, as per a learned locality sensitive hashing model, the implied (valid) similarity function may account for matching of only certain patterns in textual responses.
works lack theoretical guaranties for locality sensitivity of hash functions, the common intuitive principle in these approaches is to learn a randomized binary hash function, constructed using a very small subset of available data points. For instance, in [Joly and Buisson, 2011], a kernelized hash function is built by learning a (random) maximum margin boundary (SVM) that discriminates between the samples of two small random subsets of a super set; same principle is followed in [Garg et al., 2018b] for constructing a random hash function, i.e. obtaining a k-Nearest Neighbors based discrimination boundary between the two random subsets using kernel functions, instead of a maximum margin boundary. In [Kulis and Grauman, 2009; 2012], a hash function is built using the union of two random subsets, that is an (approximately) random linear hyperplane in the kernel implied feature space. Despite the lack of theoretical guaranties for locality sensitivity of kernel based hash functions, the principle of constructing each random hash function by learning it on a very small random subset of samples works well in practice.

Considering the commonality in these hashing methods due to the principle of random sub-sampling, we present a generalization in Alg. 1 for generating locality sensitive hashcodes, that is applicable for building a random hash function, operating on language structures, either (i) using a convolution kernel, \( K(S_i, S_j) \), defining similarity between two structures \( S_i, S_j \) [Mooney and Bunescu, 2005; Collins and Duffy, 2002; Haussler, 1999], as proposed in the previous works [Kulis and Grauman, 2009; Joly and Buisson, 2011; Garg et al., 2018b], or as we additionally propose in this paper, (ii) using a neural language model such as LSTM, GRU, etc.

In Alg. 1, the pseudo code is generically applicable for obtaining hashcode of any language structure, including responses from a dialogue. We have a set of language structures, \( S^R \) of size \( M \), with no class labels. For building \( j \)th hash function, we take a random subset \( S^R(r^j_i) \) of size \( 2\alpha \), s.t. \( 1 < \alpha \ll M \), from the set \( S^R \); this subset is partitioned into two subsets, each of size \( \alpha \), through the assignment of artificially generated hash bit labels \( z \). Having the two small subsets, we can learn a binary classifier (preferably regularized), based on kernel similarities or a neural language model, that discriminates between the two subsets, acting as a hash function. Following this principle, we can learn \( H \) number of hash functions, with each one taking constant number of computations for learning since the size of the subset \( S^R(r^j_i) \) is a small constant, \( 2\alpha \ll M \).

When building the hash functions using kernel similarity functions, the size \( M \) of the reference set \( S^R \) should be small. This is because generation of a kernelized hashcode for an input structure \( S_i \) using the locality sensitive hashing algorithm, requires computing convolution kernel similarities of \( S_i \) with all the elements of the reference set \( S^R \). A hashcode \( c_i \) for a structure \( S_i \) represents finding important substructures in it related to the set \( S^R \). This means, when generating hashcodes of dialogue responses as representations in our framework, we can control what patterns should be attended by the therapist in the patient responses through the optimization of the reference set itself. This idea can be very powerful when modeling dialogues involving long responses from patients. In Sec. 4, we propose to optimize a reference set using a novel algorithm.

On the other hand, when constructing a parametric hash function by learning a binary classifier based on a neural language model as proposed above, we don’t need to restrict the size of reference set, \( S^R \). Instead, we propose to optimize

2Note that the similarity function, implied from a kernel locality sensitive hashing model, may not be same as the kernel function. While the kernel function may be accounting for similarity of all possible patterns in textual responses, the similarity function would be effectively matching only certain patterns, defined as per the reference set in the hashing model.
the architecture of all the $H$ number of neural models, corresponding to $H$ hash functions, jointly with our novel algorithm in Sec. 4 while learning the weight parameters of the neural models independently of each other.

### 3.3 Mapping Inferred Hashcodes to Textual Responses

While we propose to represent patient/therapist responses as hashcodes, and infer the hashcode for a therapist response instead of directly inferring the textual response, we can map an inferred therapist hashcode to multiple choices for the final textual response. As mentioned previously, one of the main advantages of using binary hash functions to build representations of responses is that one can use the same binary hashing model for indexing a repository of sentences. For the task of building an automated therapist (assisting a human therapist), from a training set of therapy sessions, we can put all the responses from therapists into a single repository. Then, having a hashing model as per Alg. 1, all the responses can be indexed, with binary hashcodes. Now, after inferring a therapist hashcode, we can search in the repository of hashcode-indexed responses, to find multiple responses having minimal hamming distance w.r.t. the inferred hashcode. Even if the number of hashcode bits were large, sublinear time algorithms can be used for efficient similarity search (nearest neighbors) in hamming spaces [Norouzi et al., 2014; Komorowski and Trzczinski, 2017]. For e.g., see Tab. 4.

### 4 Learning Hashing for Dialogue Modeling with Mutual Information Maximization

In the previous section, we introduced a framework for modeling dialogues via hash functions, especially in the context of a therapeutic dialogue between a therapist and a patient. The question is what objective function to use for the optimization of the hashing model? In the following, we introduce an objective function that is suitable for optimizing hash functions for dialogue modeling. The two main criteria for selecting this objective function are as follows: (1) it should characterize the quality of hashcodes as generalized representations of dialogue responses; (2) it should also account for the inference of therapist hashcodes, i.e. therapist dialogue generation. Although the primary motivation of the present work is therapeutic dialogue modeling, we note that the above criteria also apply to more general dialogue modeling problems as well.

### 4.1 Objective Function Formulation to Optimize Hashing for Dialogue Modeling

We propose that maximizing Mutual Information between the hashcodes of patient and the corresponding therapist dialogue responses is suitable to optimize on both the inference accuracy as well as the representation quality.

$$
\arg \max_{M_h} \mathcal{I}(\mathcal{C}_p : \mathcal{C}_t ; M_h) \\
\mathcal{I}(\mathcal{C}_p ; \mathcal{C}_t ; M_h) = \mathcal{H}(\mathcal{C}_t | M_h) - \mathcal{H}(\mathcal{C}_t | \mathcal{C}_p : M_h) 
$$

**Algorithm 2** Algorithm for Optimizing Neural Architecture in a Neural-Hashing Model for Dialogue Modeling by Maximizing Our MI LB

*Require:* Train sets, $\mathcal{S}^p$, $\mathcal{S}^t$; maximum number of layers in neural language models, $L$; the number of samples for computing the MI lower bound, $\gamma$; values for units in a layer, $\{5, 10, 20, 40, \text{None}\}$.

*$\%$ optimizing up to $L$ layers greedily

1: for $j = 1 \rightarrow L$ do
2: $r^j$ ← randomSubset$(N, \gamma)$ % subset of patient/therapist responses pairs for computing the MI LB to optimize $j_{th}$ layer
3: % Number of units in $j_{th}$ layer, None not applicable for $1_{st}$ layer
4: for $l \in u$ do
5: $C^l_j$ ← computeHashcodes($\mathcal{S}^p(r^j_l), n$)
6: $C^t_j$ ← computeHashcodes($\mathcal{S}^t(r^j_l), n$)
7: $m_{lb}(l) ← \text{computeMI} \_\text{lower} \_\text{Bound}(C^p, C^t)$
8: end for
9: $n(j) ← \text{maxMI} \_\text{LB} \_\text{Index}(m_{lb}, u)$ % choose the units with maximum value of MI LB
10: if $n(j)$ is None then break out of loop end if
11: end for
12: return $n$

Herein, $M_h$ denotes a hashing model, such as the one described above; $\mathcal{C}_p$ is the distribution on the hashcodes of patient dialogue responses, and $\mathcal{C}_t$ characterizes the distribution on the hashcodes of the corresponding therapist responses. While minimizing the conditional entropy, $\mathcal{H}(\mathcal{C}_t | \mathcal{C}_p ; M_h)$, is to improve the accuracy for the inference of the therapist response hashcodes, maximizing the entropy term, $\mathcal{H}(\mathcal{C}_t ; M_h)$, should ensure good quality of the hashcodes as generalized representations. See Fig. 2 for an illustration.

Computing mutual information between two high-dimensional variables can be expensive, potentially inaccurate if the number of samples is small [Kraskov et al., 2004]. So, we propose to optimize the hashing model, by maximizing a lower bound on the mutual information criterion.

### 4.2 Information Theoretic Bounds for Optimizing and Evaluating Hashing for Dialogue Modeling

We develop a novel lower bound on the mutual information criterion, that is cheaper and more accurately computable than the criterion itself.

**Theorem 1** (Lower Bound for Mutual Information). Mutual information between two hashcode distributions, $\mathcal{I}(\mathcal{C}_t : \mathcal{C}_p ; M_h)$, is lower bounded as,

$$
\mathcal{I}(\mathcal{C}_t : \mathcal{C}_p ; M_h) \geq \sum_j \mathcal{H}(\mathcal{C}_t(j) | M_h) - \mathcal{T}(\mathcal{C}_t : \mathcal{Y}^*; M_h) + \sum_j \mathcal{q}(\mathcal{C}_t(j) | \mathcal{C}_p ; M_h) \mathcal{H}(\mathcal{C}_t ; \mathcal{Y}^*; M_h) 
$$

(3)

$$
\mathcal{Y}^* \leftarrow \arg \max_{\mathcal{Y}^*} \mathcal{T}(\mathcal{C}_t : \mathcal{Y}; M_h) 
$$

(4)

Herein, $\mathcal{T}(\mathcal{C}_t : \mathcal{Y}; M_h)$ describes the amount of Total Correlations (Multi-variate Mutual Information) $^3$ within $\mathcal{C}_t$ that

$^3$The “total correlation” quantity, also called as multi-variate Mutual Information, or multi-information, was defined in [Watanabe, 1960; Studený and Vejnarová, 1998; Kraskov et al., 2005].
Algorithm 3 Algorithm for Optimizing Reference Set in Kernelized Hashing for Dialogue Modeling by Maximizing Our MI LB

Require: Training sets, \( S^p, S^t, \bar{S} \); initial and final size of reference set, \( I \) and \( M \) respectively; \( \beta \) and \( \gamma \) are the number of samples, as candidates for the reference set, and for computing the lower bound, respectively.

1: \( r^d \leftarrow \text{randomSubset}(2N, I) \) % random subset of indices, of size \( I \), from \( \{1, \cdots, 2N\} \) for initialization of reference set
2: \( \text{for } j = 1 \rightarrow M \) do
3: \( \text{if } j > I \text{ then} \)
4: \( r^d \leftarrow \{r^d, \text{randomSubset}(2N, 1)\} \) % adding one more element in the reference set, that is to be optimized
5: \( \text{end if} \)
6: \( r^{ref} \leftarrow \text{randomSubset}(2N, \beta) \) % subset of the structures as candidates for the reference set
7: \( r^{th} \leftarrow \text{randomSubset}(N, \gamma) \) % subset of patient/therapist responses pairs for computing the MI lower bound
8: \( K^p \leftarrow \text{computeKernel}(S^p(r^{th}), S(r^d)) \) % \( \gamma \times j \) size
9: \( K^t \leftarrow \text{computeKernel}(S^t(r^{th}), S(r^d)) \) % \( \gamma \times j \) size
10: \( K^p \leftarrow \text{computeKernel}(S^p(r^{th}), S(r^{ref})) \) % \( \gamma \times \beta \) size
11: \( K^t \leftarrow \text{computeKernel}(S^t(r^{th}), S(r^{ref})) \) % \( \gamma \times \beta \) size
12: \( p_{\text{mi}} \leftarrow \text{computeMILowerBound}(K^p, K^t, K^p, K^t) \) % compute the MI lower bound for all the candidates \( S(r^{ref}) \), using the kernel matrices, via computation of hashcodes
13: \( r^d(j) \leftarrow \text{maxMILBDataIndex}(p_{\text{mi}}, r^{ref}) \) % choose the index with maximum value of MI lower bound, from the set of candidate indices \( r^{ref} \)
14: \( \text{end for} \)
15: return \( \bar{S}(r^d) \)

can be explained by a latent variables representation \( Y; q(C_t(j)|C_p; M_h) \) is a proposal conditional distribution for the \( j \)-th bit of therapist hashcodes built using a classifier, like a random forest, neural network, etc.

An interesting aspect of the quantity \( TC(C_t; Y; M_h) \) is that one can compute it efficiently for optimized \( Y^* \) that explains maximum possible Total Correlations present in \( C_t \), see [Ver Steeg and Galstyan, 2014] for more details on the optimization; for practical purposes, the dimension of latent representation \( Y \) can be kept much smaller than the dimension of hashcodes, i.e. \( |Y| \ll |C_p| \) for \( |C_p| \gg 1 \).

As we observed for the mutual information criterion above, we can also see different terms in the mutual information lower bound having varying roles for the optimization; for instance, the first two terms in (3) contribute to improve the quality of hashcodes as representations, i.e. maximizing entropy of each hashcode bit while discouraging redundancies between the bits, and the last term of conditional entropies is for improving inference of hashcode bits individually.

We argue to use the proposed MI lower bound, for not only optimizing hashing for dialogue modeling, but also as an evaluation metric quantifying the representation quality as well as the inference of hashcodes; besides, it serves to quantify what is mutually relevant between the responses of two dialog agents.

Further, for evaluating representation quality of hashcodes separately, in Theorem 2 below, we present a novel tight upper bound for joint entropy of hashcodes.

Theorem 2 (Upper Bound for Entropy). Joint entropy of hashcodes, \( \mathcal{H}(C_t; M_h) \), is upper bounded as,

\[
\mathcal{H}(C_t; M_h) \leq \sum_j \mathcal{H}(C_t(j); M_h) - TC(C_t; \mathcal{Y}^*; M_h);
\]

\[
\mathcal{Y}^* \leftarrow \arg \max_{\mathcal{Y};|\mathcal{Y}|=|C_p|} TC(C_t; \mathcal{Y}; M_h)
\]

Note the similarities between the expressions for proposed the mutual information lower bound, and the joint entropy upper bound; the first two terms match between the two expressions.

For \( TC(C_t; \mathcal{Y}^*; M_h) = 0 \), i.e. when a latent representation \( \mathcal{Y}^* \) is learned which explains all the total correlations in \( C_t \), the upper bound becomes equal to the entropy term; practically, for the case of hashcodes, learning such a representation should not be too difficult, so the bound should be very tight. This is quite interesting as we are able to judge the representation quality of hashcodes, that we proposed to quantify via the entropy of hashcodes in the above, through the tight upper bound in Theorem 2 which is much easier and cheap/accurate to compute than the entropy term itself. Besides the mutual information lower bound, for more detailed analysis, we propose to use the upper bound as an evaluation metric for the dialogue problem when modeling via hashing functions.

The proposed information theoretic bounds are generically applicable for any high dimensional variables. For instance, our MI LB and Entropy UB are generically applicable for other paragraph-embedding like representations [Wieting et al., 2015; Arora et al., 2016], though more efficient to compute on binary ones. See Appendix section for derivation of the bounds.

Previously, variational lower bounds on the mutual information criterion have been proposed [Barber and Agakov, 2003; Chalk et al., 2016; Gao et al., 2016; Chen et al., 2016; Alemi et al., 2017; Garg et al., 2018a] for settings where one of the two variables is fixed (say class labels). This simplifies the derivation significantly, compared to our setting.

Next, we discuss details on how to optimize kernel- or neural-hashing (Alg. 1) by maximizing our MI LB.
### 4.3 Details on Optimization of Kernel- or Neural-Hashing with Maximization of MI LB

In the above, we formalized an objective function, i.e. a lower bound on the mutual information criterion, to optimize all the hash functions jointly for dialogue modeling, while keeping each hash function randomized so as to preserve the property of locality sensitive hashing. In this section, we discuss optimization details specific to the kernel- or neural-hashing models as described in Sec. 3.

**Optimizing Kernel-Hashing Models for Dialogue Modeling**

For kernel-hashing in 1, we can optimize the reference set, $S^H$, or the kernel parameters. For the selection of structures in $S^H$, we use a greedy algorithm maximizing the proposed mutual information lower bound (in Theorem 1); see the pseudo code in Alg. 3. We initialize a reference set of small size $I \ll M$, by randomly selecting responses from the training set of patient/therapist responses, i.e. $\tilde{S} = \{S^p_1, \ldots, S^p_N, S^t_1, \ldots, S^t_T\}$; though, as noted before, the superset $\tilde{S}$ for the random selection can be any set of sentences/paragraphs, not necessarily coming from a dataset of patient/therapist responses. First, each element in the initial reference set is optimized greedily, and then more elements are added one by one until the reference set size grows to $M$. When optimizing each element in the set, for computing the MI lower bound, we sample $\gamma$ number of response pairs from the training set of patient/therapist responses pairs, $\{(S^p_i, S^t_j)\}_{i=1}^N$. For computational efficiency, we adopt the idea of sampling for the candidate set as well, in each greedy optimization step, by sampling a subset of candidates of size $\beta$ from the set $\tilde{S}$.

The computation cost in the optimization is dominated by the number convolution kernel similarities, i.e. $O(\gamma(M^2 + M\beta))$. In practice, we can keep low values of $\gamma$ as well as $\beta$; in our experiments, we use $\beta = 1000$, $\gamma = 100$, and vary the value of $M$ from 30 up to 300. A similar procedure can be used to optimize kernel parameters.

**Optimizing Neural Network Architecture in Neural-Hashing Models**

As mentioned previously, if using language neural models for hashing in Alg. 1, we can optimize the number of layers and the units in each layer, by maximizing the proposed MI LB; see pseudo code in Alg. 2.

### 5 Experiments

In this section, we discuss our experimental simulations, primarily on a dataset of textual transcriptions of the audio recordings of 400 depression therapy sessions, with each session conducted by a human therapist with a real patient, overall involving hundreds of patients and therapists, along with an additional dataset of interview transcripts between Larry King (host) and his guests. For the purposes of evaluations in this paper, we put all the pairs of patient/therapist responses from different sessions together in a single set, $\{S^p_{i}, S^t_j\}_{i=1}^N$, of size $N = 42000$. We perform random 90%-10% split of the dataset into a training (38,000 response pairs), and a test set (4200 response pairs), respectively. Similarly, for the second dataset, we put together all pairs of host/guest responses from 75 sessions in a single set of size 8200. Further, we perform 10 trials for obtaining the statistics on evaluation metrics in each experiment, with 95% response pairs sampled from the training subset, and the same percent of pairs from the test subset.

**Hashing Settings**

The number of hashcode bits is hundred, $H = 100$, for both patient and therapist responses. To obtain hashcodes of responses, for each sentence in a textual response of a patient/therapist, we obtain Part of Speech Tags as additional features, known as POS.

In case of kernel-hashing, we use subsequences kernels [Mooney and Bunescu, 2005] for computing similarity

---

4http://transcripts.cnn.com/TRANSCRIPTS/lkl.html

5We use random seed value zero for numpy.random package.
Can you? They're leaving, they're leaving.
Yes.
When I took the 60, I didn't sleep for like two days.
I feel like it is doing nothing.
 Talk to me and listen to me.
Mm-hm.
Uh-huh.
No, I'm not about to tell him. Hm-um.
It was one of the few things in there that I actually bought from her. None of the things I think... She was trying hard to say the right things. Where have I ever heard that from? So I leave the conversation and go, "All right, well, we'll see what happens." And what she asked in return is that when I'm angry at her, tell her. "I'm angry at you. I'm not going to talk to you for a while now." She said, "As long as I know."
Sure. I'll see you day after tomorrow.
You're not going any further in therapy right now because you haven't decided which way you want to go. It really, to me — okay, maybe I'm over playing it but it seems like a parting in the way. Which game are you going to play? Which game do you want to be good at? And you keep pussyfooting saying well, I like a little bit of this and I like a little bit of that.
And I can't breathe. I-I-I can still breathe, you know.
Right.
You're not going to church or nothing —
Yeah, it's basically the, um, the recordings are sort of a $50 subsidy —

Table 3: We show first 15 responses (in raw text format) that are selected greedily for the reference set ($S^R$) in kernel-RMM hashing model, out of 76,000 responses from the train set, with Alg. 3.

between two responses \(^6\), while similarity between a pair of words is computed using wordnet if their POS tags match. For the size of reference set $S^R$, we try values, 30, 100, 300 $(\alpha = 10)$. Reference set is initialized either randomly as a subset of all the responses in the training set (Random Selection), or the sub-selection is optimized with Alg. 3 (MI LB Suboptimal), with configurations $\beta = 1000, \gamma = 100, I = 15$.

See Tab. 3. We explore two different kernel locality sensitive hashing techniques [Joly and Buisson, 2011; Garg et al., 2018b], referred as RMM \(^7\) and RkNN respectively, while considering the latter one as our primary choice for detailed analysis.

For the case of neural-hashing, we use all the responses from the training set as the reference set of size 76,000; since the reference set is very large, we use random subsets of larger size for constructing hash functions, i.e. $\alpha = 50$. We use an LSTM model to construct each hash function, trained using Adams algorithm [Kingma and Ba, 2014], with learning rate $1e^{-3}$, amsgrad=True, and $l_1, l_2$ regularization coefficients of value $1e^{-4}$, and the gradients are computed with back propagation. We initialize a word in a response and its POS tag with a randomly initialized word vector of size 30; for a single time step processing with an LSTM, word vectors of 10 adjacent words, along with their POS tags, are appended into a vector of size 600; this is required to avoid vanishing gradients since patient responses can be of length up to 8,000 words in the training dataset. For the $H$ number of LSTM models as neural-hash functions, same neural architecture, i.e., same number of layers and units, are used in each model. When optimizing the architecture of the LSTM models with Alg. 2 by maximizing our proposed MI LB (MI LB Suboptimal), we add layers one by one greedily up to maximum possible 4 layers ($L = 4, \gamma = 1000$), and try out different possible numbers of normal units in each layer, i.e., 4, 8, 16, 32, 64, \(^8\).

Random Forests for Inferring Therapist Hashcodes

From the above model, we obtain hashcodes for patient and therapist responses in the training and test subsets, serving as ground truth for the task of inferring the hashcode of the therapist response, given the hashcode of a patient response. For inferring each bit of therapist code, we train an individual Random Forest (RF) classifier containing 100 decision trees. All of the hundred RF classifiers (since $H = 100$) share the same features (i.e. the hashcodes of patient responses) though trained independent of each other.

5.1 Evaluation Results

Evaluation metrics Our primary evaluation metric is our proposed MI lower bound (MI LB) in Theorem 1. We also use the proposed upper bound on the entropy of therapist hashcodes in Theorem 2 to evaluate the quality of hashcodes as representations of responses (Entropy UB). In addition, we express the ratio of MI LB and Entropy UB as Normalized MI LB (NMI LB). It is also an interesting to analyze MI LB for random pairings between patient and therapist responses (denoted as “Shuffled”); same applies for NMI LB.

Interpreting the accuracy metric (Accuracy), i.e. the accuracy of inferring each hashcode bit using the RF classifiers, requires a more detailed discussion. Since the class labels (therapist hashcodes) are not fixed, the accuracy is inversely correlated with the Entropy UB metric. We also obtain the baseline accuracy (Baseline), using a trivial classifier that always chooses the most-frequent class label. While analyzing the absolute numbers of accuracy is not meaningful here, we can see the relative improvement of accuracy w.r.t. the accuracy of the baseline dummy classifier. The mean and standard deviation statistics for each metric are computed over 10 runs of the experiment, as mentioned above; in the case of the accuracy metric, the statistics are computed over \(^8\)We keep the number of units small, since $\alpha$ is small.

---

\(^6\) $\lambda = 0.8$, subsequence of length up to 8 are matched

\(^7\) In this hashing technique, an SVM is learned for each hash function. We use $C = 0.1$ in SVM, keeping it highly regularized.
Run away and, and be by myself.

**Um-hum.**
(i) Uh-huh. (ii) Oh. (iii) That doesn’t pan out.

Yeah, we’ll be in touch about all that stuff.

**Okay. So very nice to see you again.**
(i) OK, I see. (ii) Kind of like that- (iii) Good, good. What’s new anything?

Sensitive. Yeah. (inaudible) just like (inaudible) very uncomfortable.

**Uh-huh.**
(i) What’s going on with the moods? (ii) Big people don’t feel this way like? (iii) Scares you how?

Yeah. It just feels moderately depressing. My head spins

**Do you feel depressed?**
(i) So eventually it will die out. (ii) So it may peter out later. (iii) We can stop for now.

Well, am I kidding myself when I’m saying nothing’s bothering me? So that bothers me.

**When you don’t - whether you can really trust your own feelings.**
(i) To be at - all of us. (ii) I’m sure. (iii) That sounds exhausting, yeah.

(laughs) Um, yeah. (laughs)

**Forever.**
(i) OK, I see. (ii) Okay, I understand. (iii) Right all, take it.

But it’s like that’s what I’ve been doing. And I don’t, I can’t scrape up the money to go back to school.

**Yeah.**
(i) Um-hum. (ii) Mm-hmm. (iii) Hmm.

| Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Run away and, and be by myself.

**Um-hum.**
(i) Uh-huh. (ii) Oh. (iii) That doesn’t pan out.

Yeah, we’ll be in touch about all that stuff.

**Okay. So very nice to see you again.**
(i) OK, I see. (ii) Kind of like that- (iii) Good, good. What’s new anything?

Sensitive. Yeah. (inaudible) just like (inaudible) very uncomfortable.

**Uh-huh.**
(i) What’s going on with the moods? (ii) Big people don’t feel this way like? (iii) Scares you how?

Yeah. It just feels moderately depressing. My head spins

**Do you feel depressed?**
(i) So eventually it will die out. (ii) So it may peter out later. (iii) We can stop for now.

Well, am I kidding myself when I’m saying nothing’s bothering me? So that bothers me.

**When you don’t - whether you can really trust your own feelings.**
(i) To be at - all of us. (ii) I’m sure. (iii) That sounds exhausting, yeah.

(laughs) Um, yeah. (laughs)

**Forever.**
(i) OK, I see. (ii) Okay, I understand. (iii) Right all, take it.

But it’s like that’s what I’ve been doing. And I don’t, I can’t scrape up the money to go back to school.

**Yeah.**
(i) Um-hum. (ii) Mm-hmm. (iii) Hmm.

| Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

| Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.

Table 4: Here, we present some examples of textual response choices produced with our hashing based approach (raw text format). For a patient response (Black), we show the ground truth response from a human therapist (Blue), and three textual response choices generated with kernel-RMM model (Magenta) via mapping of the inferred therapist hashcode to hashcode-indexed therapist responses from the training set.
Neural-Hashing

For the optimization of Neural-RLSTM hashing model, Alg. 2 gives a simple LSTM architecture of [16, 64, 16, 8], i.e. four layers, with 16 LSTM units in the first layer (bottom layer receiving the inputs), 64 LSTM units in the second layer, and so on. For this model, we obtain relatively higher values for MI LB (17.2), compared to the kernel-hashing above. Besides the optimized architecture for LSTM models, we also tried manually built architectures; for instance, if using simple LSTM model of single layer with 10 units for each hash function, we obtain MI LB value, 6.7, that is significantly lower compared to the optimized Neural-RLSTM model (17.2), and many of the kernel-hashing models above.

The above results demonstrate that our approach of optimizing hashing functions by maximizing our novel MI LB leads to higher values for the evaluation metrics as desired, across the kernel- as well as neural-hashing models. There is a trade off between kernel-hashing and neural-hashing. In kernel-hashing, as per the selected responses in the optimized reference set, one can get insights on what patterns are relevant for a dialog; for e.g., see Tab. 3. On the other hand with neural-hashing, one can take advantage of the vast variety of neural language models available.

Further, we can map an inferred hashcode to textual responses using a repository of indexed sentences. See Tab. 4.

Larry King Dataset Results

Some selected results for the Larry King dataset are presented in Tab. 5. Herein, it is interesting to see that we get very high value of MI LB, and Entropy UB with our neural hashing approach (optimized architecture is [8,32]), in comparison to our kernel hashing. Also, mean accuracy with neural hashing is significantly higher than the baseline accuracy number. Though, NMI LB values for the two hashing approaches are relatively close.

Table 5: Evaluating the quality inference of dialogue responses (hashcodes), for the Larry King dataset.

<table>
<thead>
<tr>
<th>Hashing Func.</th>
<th>Hash Config.</th>
<th>Model</th>
<th>MI LB (Shuffled)</th>
<th>Entropy UB</th>
<th>NMI LB (Shuffled)</th>
<th>Accuracy (Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel-RKNN</td>
<td>M=100,α=10</td>
<td>Random Sel.</td>
<td>15.1 ± 0.1 (7.3 ± 0.1)</td>
<td>22.6 ± 0.4</td>
<td>0.67 (0.32)</td>
<td>87.7 ± 12.9 (81.8 ± 17.3)</td>
</tr>
<tr>
<td>Kernel-RKNN</td>
<td>M=100,α=10</td>
<td>MI LB Opt.</td>
<td>16.8 ± 0.6 (7.9 ± 0.4)</td>
<td>25.3 ± 0.6</td>
<td>0.66 (0.31)</td>
<td>86.2 ± 13.8 (79.5 ± 17.2)</td>
</tr>
<tr>
<td>Kernel-RKNN</td>
<td>M=300,α=10</td>
<td>Random Sel.</td>
<td>20.4 ± 0.4 (11.7 ± 0.2)</td>
<td>26.9 ± 0.5</td>
<td>0.76 (0.43)</td>
<td>88.3 ± 11.0 (80.7 ± 15.2)</td>
</tr>
<tr>
<td>Neural-RLSTM</td>
<td>M=18000,α=50</td>
<td>MI LB Opt.</td>
<td>49.5 ± 0.3 (28.0 ± 0.4)</td>
<td>60.6 ± 0.2</td>
<td>0.82 (0.46)</td>
<td>69.1 ± 7.8 (55.8 ± 8.3)</td>
</tr>
</tbody>
</table>

6 Conclusions

This paper introduces a novel approach to dialogue modeling based on hash functions, using psychotherapy sessions as a motivating domain. In our framework, responses from both parties (e.g., patient and therapist) are represented by the corresponding hashcodes, capturing certain text patterns. Furthermore, we propose a novel lower bound on Mutual Information in order to characterize the relevance of a therapist’s response to the patient’s text, and vice versa. Moreover, in order to characterize the general quality of hashcodes as response representations, we propose a tight upper bound on the joint entropy of hashcodes. We performed empirical evaluation of the proposed approach on the dataset containing depression therapy sessions between real patients and therapists. We optimized locality sensitive hashing models, based on kernel functions or neural language models, by maximizing the proposed MI lower bound as an objective function. Our results consistently demonstrate superior performance of the proposed approach over several alternatives, as measured by several evaluation metrics.

References


A Derivations of the Information Theoretic Bounds

Before the discussion of our novel lower bound, we introduce the information-theoretic quantity called Total Correlation ($TC$), which captures non-linear correlation among the dimensions of a random variable $X$, i.e.,

$$\text{TC}(X; M_h) = \sum_j H(X(j); M_h) - H(X; M_h);$$

$$\text{TC}(X : Y; M_h) = \text{TC}(X; M_h) - \text{TC}(X|Y; M_h).$$  \hspace{1cm} (5)

Intuitively, (5) describes the amount of information within $X$ that can be explained by $Y$.

Along these lines, the mutual information quantity between the hashcodes can be decomposed as in Lemma 1 below.

Lemma 1 (Mutual Information Decomposition). Mutual Information between $C_t$ and $C_p$ is decomposed as follows:

$$\mathcal{I}(C_t : C_p; M_h) = \sum_j \mathcal{I}(C_t(j) : C_p; M_h) - \text{TC}(C_t : C_p; M_h).$$ \hspace{1cm} (6)

Looking at the first term of RHS in (6), it is the mutual information between a one-dimensional and multi-dimensional random variable.

For these terms, since one of the variables is only 1-D, we can use the existing technique of variational bounds for an approximation [Gao et al., 2016], as in Lemma 2 below.

Lemma 2. Marginal mutual information for each bit in therapist hashcodes, $\mathcal{I}(C_t(j) : C_p; M_h)$, is lower bounded as,

$$\mathcal{I}(C_t(j) : C_p; M_h) \geq H(C_t(j); M_h) + \log q(C_t(j)|C_p; M_h) p(C_t(j), C_p; M_h).$$ \hspace{1cm} (7)

Herein, $H(C_t(j); M_h)$ is easy to compute because $C_t(j)$ is one-dimensional. For each of the proposal distributions $q(C_t(j)|C_p; M_h)$, we propose to use a Random Forest (RF) classifier [Garg et al., 2018a].

In reference to the second term of RHS in (6), it is computationally intractable to compute the total correlation expression $\text{TC}(C_t : C_p; M_h)$, which denotes the total correlations between bits of $C_t$, explainable by $C_p$. So, we would also like to obtain an upper bound of $\text{TC}(C_t : C_p; M_h)$, which is cheap to compute, that would give us a lower bound for the second term in (6) because of the negative sign.

Lemma 3. $\text{TC}(C_t : C_p; M_h)$ can be upper bounded as:

$$\text{TC}(C_t : C_p; M_h) \leq \text{TC}(C_t : Y^*; M_h);$$ \hspace{1cm} (8)

wherein $|\cdot|$ denotes the dimensionality of a random variable.

Although it is intractable to compute the original term $\text{TC}(C_t : C_p; M_h)$, it is possible to compute $\text{TC}(C_t : Y^*; M_h)$ for a latent variable representation $Y^*$ of $C_t$ that maximally explains the Total Correlations in $C_t$.

We can think of the computation of the upper bound as an unsupervised learning problem. We propose to use an existing algorithm, CorEx, for the unsupervised learning of latent
random variables representation $\mathbf{Y}^*$ [Ver Steeg and Galstyan, 2014].

It is important to note some practical considerations about the upper bound. In the case of a suboptimal solution to the maximization of $\mathcal{T}C(\mathbf{C}_t; \mathbf{Y}; M_h)$ above, the optimized quantity may not be an upper bound of $\mathcal{T}C(\mathbf{C}_t; \mathbf{C}_p; M_h)$, but rather an approximation. Also, the upper bound would not be tight if $\mathbf{C}_p$ doesn’t explain much of total correlations in $\mathbf{C}_t$. Further, for even more computation cost reductions during the learning, the dimension of the latent representation $\mathbf{Y}$ can be kept much smaller than the dimension of hashcodes, i.e. $|\mathbf{Y}| \ll |\mathbf{C}_p|$ for $|\mathbf{C}_p| \gg 1$; this is because even a small number of latent variables should explain most of the total correlations for practical purposes as demonstrated in [Ver Steeg and Galstyan, 2014], and observed in our experiments on hashcodes as well.

Combining (7) and (8) into (6), we get the lower bound in (3) in Theorem 1.

Along same lines, we derive the tight upper bound on joint entropy of hashcodes in Theorem 2. From the definition of Total Correlation above (5), we have the following,

$$
\sum_j H(\mathbf{C}_t(j); M_h) - \mathcal{T}C(\mathbf{C}_t; M_h) = H(\mathbf{C}_t; M_h),
$$

$$
\mathcal{T}C(\mathbf{C}_t; M_h) = \mathcal{T}C(\mathbf{C}_t; \mathbf{Y}^*; M_h) + \mathcal{T}C(\mathbf{C}_t|\mathbf{Y}^*; M_h),
$$

and finally the expression below.

$$
\sum_j H(\mathbf{C}_t(j); M_h) - \mathcal{T}C(\mathbf{C}_t; \mathbf{Y}^*; M_h)
= H(\mathbf{C}_t; M_h) + \mathcal{T}C(\mathbf{C}_t|\mathbf{Y}^*; M_h)
$$

From this derived expression, we can simply obtain the upper bound and the corresponding gap.

**Previous Lower Bounds for Mutual Information:** Variational lower bounds on the mutual information criterion have been proposed in the past [Barber and Agakov, 2003; Chalk et al., 2016; Gao et al., 2016; Chen et al., 2016; Alemi et al., 2017; Garg et al., 2018a]. Their lower bounds works only when one of the variables is fixed, say if $\mathbf{C}_t$ were fixed. In our objective, not only $\mathbf{C}_t$ is a functional of the hashing model that we are learning, it is high dimensional. Unless we have a lower bound for the entropy term $H(\mathbf{C}_t; M_h)$ as well, which should be hard to obtain, we can not use the above mentioned variational lower bounds for our problem as such. Besides, it is also non-trivial to find an appropriate proposal distribution $q(\mathbf{C}_t|\mathbf{C}_p; M_h)$. Therefore, we adopt a different approach for obtaining a novel lower bound on the mutual information quantity, as described above.
Conversational Control Interface to Facilitate Situational Understanding in a City Surveillance Setting

Daniel Harborne1, Dave Braines2, Alun Preece1, Rafal Rzepka3
1 Crime and Security Research Institute, Cardiff University, Cardiff, UK
2 IBM Emerging Technology, Hursley, UK
3 Graduate School of Information Science and Technology, Hokkaido University, Japan
4 RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan
harborned@cardiff.ac.uk

Abstract
In this paper we explore the use of a conversational interface to query a decision support system providing information relating to a city surveillance setting. Specifically, we focus on how the use of a Controlled Natural Language (CNL) can provide a method for processing natural language queries whilst also tracking the context of the conversation with relation to past utterances. Ultimately, we propose our conversational approach leads to a versatile tool for providing decision support with a low enough learning curve such that untrained users can operate it either within a central command location or when operating within the field (at the tactical edge). The key contribution of this paper is an illustration of applied concepts of CNLs as well as furthering the art of conversational context tracking whilst using such a technique.
Keywords: Natural Language Processing (NLP), Conversational Systems, Situational Understanding

1 Introduction
With the continued improvements made to machine-based analytics tools and techniques (such as the rise of Deep Learning), there has been an increase in the extent to which data can be processed autonomously by machines to provide actionable intelligence. We now can harness broader datasets, existing in many modalities and that are collected from many sources. Furthermore, the capability for a system to perform this collection and analysis in real time is increasingly common. For tactical decision makers, such as emergency service incident commanders, this means at the point of formulating a decision, the quantity of information feeds and the variety of the information within those feeds has vastly grown. Having access to the right information at the right time is a key aspect to making the right decision and being overloaded by information can inhibit forming a decision entirely. Due to this change in the information landscape, novel approaches to capitalizing on the vast information available need to be explored. To fulfill this need, we have seen the increased innovation and adoption of novel interaction methods, such as conversational interfaces [Mctear et al., 2016], to access and manipulate information. In this paper we explore an approach to a conversational interface that takes advantage of a Controlled Natural Language (CNL), specifically ITA Controlled English [Mott, 2010]. We first outline the key characteristics of this technology and then move on to discuss the benefits it provides. Finally we include an approach for tracking the context of user queries, furthering the capabilities of the framework. To demonstrate these factors, we use a hypothetical scenario of city-wide surveillance, where data feeds such as traffic cameras, tweets concerning the local area and reports from agents on the ground could be used to build an awareness for the state of the city. This could grant insights with regard to congestion, crimes in progress or emergencies that require response. In this work, we focus on traffic camera data feeds and on the information a surveillance system could plausibly generate when processing such data.

2 Situational Understanding and Decision Support Systems
Situational awareness (SA) [Endsley, 1995] is the ability to build an accurate model of the state of a system, with situational understanding (SU) [Smart et al., 2009] being the ability to reason about it’s current and future states. Decision support systems attempt to augment a human user’s ability to perform one or both of these tasks. These systems can offer simple aggregation of data and information sources in to a more comprehensible channel and/or can bring together services that can process such data and make inferences from the available information, providing insights, predictions and recommendations to the decision maker.

2.1 Conversational Interfaces
It is not uncommon for a decision maker’s primary skill set to be outside the realm of computer science or data analysis. Instead, they take advantage of domain knowledge and related intuition to make decisions within a given scenario, making use of information provided to them on request or preemptively by human or machine analysts. By offering, a conversational interface to a decision support system, decision makers can request information, perform reasoning and action their decisions using natural language rather than through a traditional software interface. Firstly, this can minimize the learning curve for using the system and can speed up the decision
making process. In addition, when combined with speech-to-text and text-to-speech technology, can remove the need for conventional input devices. This move away from mice, keyboards and even screens to voice input/output mechanisms not only can increase efficiency by allowing a wide range of actions to be available without using menus but also can often free the user from the requirement for a desk-based system or mobile computational device (such as a laptop or smart device). Instead, the decision maker can form requests and receive information, with minimal change to their operational behavior, including while operating at the tactical edge.

In earlier work the concept of conversational interactions to support casual system users without specific ontology or knowledge engineering capabilities has been explored. In [Pizzocaro et al., 2013], the concept of a conversational interaction to support the tasking of sensing assets within a coalition context, in a constrained environment was constructed. The work brought together earlier task-oriented matching of assets and capabilities to requirements, and placing the power of the system and all the complexities within it, behind a simple conversational interface. In [Preece et al., 2014] the work was extended further into the intelligence analysis domain, and articulated using a simple intelligence gathering and tracking scenario with various human and sensor assets providing information relevant to the task. We also formally defined the underlying conversational model using the CE language, enabling formal definition of different speech acts and the pre-defined ways in which conversational interactions can logically flow. In this work the human users were able to provide their "local knowledge" as new facts via the conversational interaction, as well as ask questions using the same interface. As a result of reasoning and other task-oriented activities the machine agents within the system were able to raise alerts and start conversations with the human users via the same conversational interface as well. Finally in [Braines et al., 2016] we extended the conversational interaction to enable development and extension of the underlying models (also known as ontologies) that underpin the system. Through these capabilities we have been able to show support for question answering interactions as well as the addition of local knowledge and the extension of the underlying models, all through natural language conversational interfaces using the Controlled English language.

3 City Surveillance

In this work, we use a scenario based on city surveillance to explore how a dialogue system can facilitate conversational control of many services and how a decision maker can use natural language to make queries and perform reasoning across the range of available information. We imagine hypothetical tasks the agent may need to perform that relate to the monitoring of traffic volumes and assisting the location and tracking of specific vehicles to assist law enforcement.

3.1 Resources Available

In our system we focus on information provided by traffic cameras. Specifically, traffic camera locations, video and imagery available via Transport for London’s Jam Cams Application Programming Interface (API)\(^1\). In our scenario, we imagine the type of services that could be available to process this data, some of which we have been using in related work [Harborne et al., 2018] and others are proposed as hypothetical services that realistically could exist. Using these services would generate information relating to detecting cars in video and imagery as well as refining the car detections to a specific make and color. For the purpose of this paper, we use pre-generated information, rather than that generated from live services as the integration of such services is outside of the scope of our work.

4 Controlled Natural Language and Controlled English

Controlled English (CE) is an example of a controlled natural language (CNL) which aim to reduce the complexity of natural language (NL) to allow for easier human-machine interaction. The benefits of CNL is that by reducing the grammar to a confined subset, the information retains a machine-readable structure whilst also being naturally readable by humans. This is converse to unstructured data, such as natural conversation, typically difficult for machines to process and highly structured data, such as XML, which are less human readable. In previous work, we have shown that controlled English can help a user control smart devices within their home [Braines et al., 2017]. In that work we outline many of the principles of CE, in this paper we will recap the fundamentals and relate them to the functionality required for this specific piece of work. It is recommended to read the previous work for a thorough explanation of CE.

4.1 Concepts, instances, rules

Controlled English allows the maintaining of a knowledge base via concepts and instances and allows for automatic inferences using rules. All three of these can be created before a support system goes live or can be created as part of the operation of the system. Concepts outline the classes of entities, instances are representations of specific known entities and rules allow for the system to perform reasoning with these items. In Figure 1 and Figure 2 we show the definition of the traffic camera concept along with some parent concepts it inherits from in order to further facilitate inference and reasoning. In Figure 3, we show an instance of a traffic camera. Both the concept and instance definition can be automatically generated from the result of querying the traffic camera API.

\(^1\)https://data.london.gov.uk/dataset/tfl-live-traffic-cameras
4.2 CE Hudson and Custom Answerers

When a user submits a query to the interface, it is sent to Hudson - an API that interprets natural language into recognized CE components². This interpretation is returned as a JSON output which the interface can use to provide an appropriate response to the user. In CE terminology, an application that reacts to Hudson API output is called a “custom answer”. This approach has both benefits and costs when compared with other machine learning approaches to NLP. A key characteristic is that the interpretation comes from the CE knowledge base, thus interpretations can’t be learned based on sentence structure or patterns (like with a deep learning approach). This can make the space of interpretable input sentences smaller. However, CE does allow for synonyms to be assigned to concepts and the approach of CE concept matching is usually powerful and robust enough to recognize user requests in a closed domain (as shown in previous work [Preece et al., 2017b]). To counter this limitation, a hybrid approach that uses deep learning for interpreting from natural language to CE concepts could be used but exploring this is outside the scope of this work.

In our use case, where we explore a control system within a closed domain, CE’s power outweighs this drawback. Unlike a user interacting with a general purpose chatbot, a tactical decision maker often will have a higher requirement for consistent and reliable answers and information. Thus, a robust interface with a slightly higher learning curve is more important than covering all possible utterances to achieve a certain goal. This increased learning curve is likely to be quickly overcome during the decision maker’s initial interaction with the system and the knowledge base can be designed in such a way that interaction is still intuitive (also shown in [Preece et al., 2017b]). In addition to this consistency of response, this approaches also allows users to update the knowledge base via the conversational interface and for these updates to immediately be utilized in input processing and output generation. This is discussed further in Section 6.

5 Rules Inferencing

Rules in CE are used to provide inherent inferencing that can take place upon the information within the knowledge base. For example, the rule shown in Figure 5 can take advantage of the properties of the region and location concepts (Figure 4) to allow the system to infer which locations are located within defined regions of the city based on the geo-position properties of the location and the boundary of the region. In addition, the rule shown in Figure 6 allows for the system to infer that if a displayable thing (such as a video source) can show a road and that road is in a region, then that camera also shows that region.

²In this work we used a publicly available open source implementation of Controlled English, named ce-store which implements a number of generic APIs for simple usage. One set of APIs, known as Hudson, enables natural language text processing in the context of a CNL model, returning an “interpretation” of the specified natural language as matches to concepts, properties, instances and more within the CNL model(s) loaded within the ce-store. ce-store, available online at http://github.com/ce-store/ce-store
This inferencing takes place as the knowledge base is updated and so new information provided to the system can lead to further inferences to be made. Like concept definitions and instances, rules can be added by users during standard usage of the interface. This is discussed further in Section 6

\[
\text{conceptualise a } \text{region } \text{REG that has the value XONE as } \text{x1} \text{ and has the value XTWO as } \text{x2} \text{ and has the value YTWO as } \text{y2} \text{ and has the value YONE as } \text{y1} \text{ and is a geolocation source.}
\]

\[
\text{conceptualise a } \text{location } \text{LOCA that is a geolocation source and } \text{is located in } \text{the region REG.}
\]

\[
\text{conceptualise a } \text{road } \text{ROAD that is a location and has the value NAME as } \text{road name }.
\]

Figure 4: Controlled English concept definitions for regions, locations and roads. These concepts, via inheritance, create a specialisation of the geolocation source concept defined earlier in figure 2

\[
\text{[DisplayableInRegion]}
\]

if (the location L has the value X as longitude) and (the location L has the value Y as latitude) and (the region R1 has the value X1 as x1) and (the region R1 has the value X2 as x2) and (the region R1 has the value Y1 as y1) and (the region R1 has the value Y2 as y2) and (the value X <= X1) and (the value X >= X2) and (the value Y <= Y1) and (the value Y >= Y2) then (the location L is located in the region R1).

Figure 5: Example of a CE rule which infers the city regions that locations are found in based on the location’s geolocation data and region boundaries.

\[
\text{[ShowRegion]}
\]

if (the displayable thing C can show the location R) and (the location R is located in the region REG) then (the displayable thing C can show the region REG).

Figure 6: CE rule that allows the system to infer that if a displayable thing instance can show a location and that location is within a region, the instance can also show the region.

6 Tellability

As outlined in previous work [Preece et al., 2017a], a system’s tellability describes how easy it is for a user to inject new or updated information in to a system during operation. This is a strength of a CE solution as not only can a user inject new instances or update those instances through the conversational interface but they can define new entirely new concepts. This does require some level of familiarity with the system but requires no coding and the interface can take advantage of the new information immediately. This is in contrast with deep learning techniques, where the creation of a new class, feature or query type will often require retraining of the model backing the interface, this can require the knowledge of a trained engineer and time before the new information is accounted for within the interface.

To illustrate this, Figure 7 and Figure 8 show the interfaces being used to request a view of a region of the city. As outlined in Section 5, this region (named, 'test region') is defined with a geospatial boundary, and rules are used to infer that any instance of a displayable thing that can show a location within that boundary (such as a traffic camera the can show a road) can show that region. In Figure 8, we see a hypothetical scenario where the user knows that a camera that is marked in the API as viewing a certain road (not in the test region area) also can indirectly view another road (one that is in the boundary of the test region). The user, via the interface, can tell the system that the camera can show the second road, the knowledge base is updated instantly and future queries will take this in to account, including when answering queries correctly requires rule inferences.

7 Actions and Query Types

To provide decision support, a system must allow a decision maker to query the data and information available within the system. Sometimes, the user simply needs to see a selection of the information or data sources for manual inspection (discussed in Section 7.1). In addition, the decision maker may want to ask a question about the state of the world and receive a computed or inferred answer. In this work, we explore three forms of query response: confirmation of the existence of entities matching desired criteria, a count of entities matching desired criteria and listing all entities matching desired criteria (these response types are detailed in Section 7.2). Identifying the required response type to appropriately answer a user’s query is achieved by detecting instances of question phrases, examples of which are shown in Figure 9.

To process user queries that contain filter criteria (such as car color), the CE knowledge base contains definition of property categories (Figure 10) which represent attributes that instances can be filtered by (these mirror a subset of the properties found on concepts defined in the knowledge base). Instances of these properties are then created reflecting possible values that can be filtered by (Figure 11). The purpose of these property and value definitions is to allow the Hudson API to identify them within an utterance from the user. These properties can be combined with the detection of other criteria such as concepts (e.g. "car"), instances (e.g. "Romford Road") or being interested in a specific relationships (e.g. "is driving on" [road])—. The detection of these filter criteria, allows the custom answerer to form a query to be sent to the knowledge base, the response of which can then be formatted appropriately and displayed to the user. It is worth noting
that the creation of these property and value definitions can be automated based on the concept and instance definitions or from another source and can also be injected by the user during operation.

A further benefit of defining properties and their possible values as concepts is that it makes context tracking possible. This is discussed further in Section 8.

7.1 Actions: "Show me..."

One important benefit to decision support systems is that they can offer one interface for accessing a range of data sources and pieces of information. By offering an efficient and easy to use method for filtering and viewing desired content, a system can ensure a decision maker is not overloaded by being presented with all available data sources simultaneously, instead requesting to see specific resources when they wish to make use of them. As seen in Figure 2, in our system, a concept exists ("displayable thing") that indicates that instances of that concept can, in some way, be presented to the user. This parent concept is inherited into further concepts such as "video source" which indicate to the interface how the data can be displayed. If the user has screens available a "video source" can be shown, if the interface includes a map a "geolocation source" can be zoomed in on and become the focus of it. These concepts also allow the user to request to view a list of sources of particular modalities or that feature particular aspects of interest. For example, requesting videosources that can show a specific location.

7.2 Query Types: "...exists?", "Count...", "List..."

Another important feature that can be offered by a decision support system is for a user to be able to ask questions of the information available. This information may have been present in the initialization (such as the location of traffic cameras) of the knowledge base or may have been generated by services processing data sources over time (such as cars within traffic camera video). To do this, filter criteria are identified as outlined at the beginning of this section, a query is formed that will filter to instances that meet the criteria. The result of this query is then returned in three possible formats based on the nature of the questions asked by the user. This response format is based on the detection of question phrases by Hudson API and the custom answerer’s reaction to those detected phrases. Examples of these query types are show in Figures 12, 13 and 14.

8 Tracking Context

Tracking context within conversational interfaces can be considered a challenging but important task. The ability for a decision maker to ask a query of the system and then subsequently refine that query creates a much more efficient work
there is an action named 'show'.
there is a question type named 'exists'.
there is a question type named 'count'.
there is a question type named 'list'.
there is a question phrase named 'are there any' that refers to the question type 'exists'.
there is a question phrase named 'are there' that refers to the question type 'exists'.
there is a question phrase named 'how many' that refers to the question type 'count'.

there is a color named 'black'.
there is a color named 'white'.
there is a color named 'red'.
there is a color named 'green'.
there is a color named 'blue'.
there is a model named 'Toyota'.
there is a model named 'BMW'.
there is a model named 'Ford'.
there is a model named 'Range Rover'.
there is a model named 'Renault'.
there is a model named 'Mazda'.
there is a direction named 'North'.
there is a direction named 'South'.
there is a direction named 'East'.
there is a direction named 'West'.

conceptualise a "property category" PROPC.
conceptualise a "color" COL that is a property category.
conceptualise a "model" MODEL that is a property category.
conceptualise a "direction" DIR that is a property category.

Figure 8: Example of updating the knowledge base via the conversational interface. Once updated with the knowledge a particular camera can indirectly show a road located within test region, the "show test region" request now shows the new camera feed due to inference.

Figure 9: Controlled English instance definitions for phrases indicating the aim of a user’s query — (defined as the question type).

Figure 10: CE concept definition for the concept "property category" and child concepts that facilitate instance filtering.

Figure 11: CE instance definitions for values the property categories can take.

Figure 12: Example of using the interface to check if any instances of a specified criteria exist.
flow in contrast to forcing the user to re-ask the same question and adding the desired additional parameters. In our approach to adding this functionality to a CE-based interface, the first step is to define query expansion phrases. These are phrases that are used in user utterances that indicate the intent to use the previous query as a base for building the next query (definition for these phrases are shown in Figure 15). The second step is to maintain a store within the custom answerer of the values for properties, the concepts, the instances and the relationships involved in the most recent query. With this information, a query can be formed using the following approach:

- If a query expansion phrase is found, use all parameters from the last query. If a value in the current utterance is from a property category that had a value(s) stipulated in the previous query, then use the "and" operator for the values.
- If no query expansion phrase is found and the current utterance contains only property values (no concepts, instances or relationships), use the previous query’s concepts, instances and relationships.
- Finally, if neither of the above are true, clear the query store and form a new query.

The query store can also be used with actions. In Figure 17, we see the conversation ends with "show me" without any stipulation of what to show. Here the custom answerer is able to infer that it should show the *displayable things* from the results of the last query.

**Figure 15:** CE definitions for the concept and instances of query expansion phrases.

```plaintext
Figure 16: An example of using the interface to make an initial query, then filtering the result rather with additional perimeters and then make a new query.
```

**Figure 17:** An example of using the interface to make an initial query, then filtering the result rather with additional parameters and finally using "show me" to display the results from the previous query without having to re-specify and of it's parameters.
9 Conclusion

In this work we have outlined characteristics and methodology for a conversational interface backed by the controlled natural language, ITA Controlled English. We have shown the benefits such an interface can provide to a decision maker and discussed the implications of using this approach over other techniques. In this work, we have also proposed a method for context tracking using a controlled language to allow for intuitive and efficient query of the system’s knowledge base. We have also identified the possibility of future work, that explores the integrating of deep learning techniques performing natural language processing from user utterances to Controlled English. This may lead to an increase in versatility and robustness of interpretation, whilst maintaining the consistency of response, tellability and inferencing provided by CE that is important for tactical decision makers.

Acknowledgments

This research was sponsored by the U.S. Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the UK Ministry of Defence or the UK Government. The U.S. and UK Governments are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. We’d like to thank Cardiff University Global Opportunities Centre for partially funding the collaboration between Crime and Security Research Institute (Cardiff University) and Graduate School of Information Science and Technology (Hokkaido University).

References


Efficient Purely Convolutional Text Encoding

Szymon Malik*, Adrian Lancucki*, Jan Chorowski
Institute of Computer Science
University of Wrocław
szymon.w.malik@gmail.com, {alan, jch}@cs.uni.wroc.pl

Abstract
In this work, we focus on a lightweight convolutional architecture that creates fixed-size vector embeddings of sentences. Such representations are useful for building NLP systems, including conversational agents. Our work derives from a recently proposed recursive convolutional architecture for auto-encoding text paragraphs at byte level. We propose alternations that significantly reduce training time, the number of parameters, and improve auto-encoding accuracy. Finally, we evaluate the representations created by our model on tasks from SentEval benchmark suite, and show that it can serve as a better, yet fairly low-resource alternative to popular bag-of-words embeddings.

1 Introduction
Modern conversational agents often make use of retrieval-based response generation modules [Ram et al., 2018], in which the response of the agent is retrieved from a curated database. The retrieval can be implemented as similarity matching in a vector space, in which natural language sentences are represented as fixed-size vectors. Cosine and Euclidean distances typically serve as similarity measures. Such approaches have been applied by participants of recent chatbot contests: The 2017 Alexa Prize [Pichl et al., 2018; Liu et al., 2017; Serban et al., 2017], and The 2017 NIPS Conversational Intelligence Challenge [Chorowski et al., 2018; Yusupov and Kuratov, 2017]. Retrieval-based modules are fast and predictable. Most importantly, they enable soft matching between representations. Apart from this straightforward application in dialogue systems, sentence embeddings are applicable in downstream NLP tasks relevant to dialogue systems. Those include sentiment analysis [Pang and Lee, 2008], question answering [Weissenborn et al., 2017], censorship [Chorowski et al., 2018], or intent detection.

Due to the temporal nature of natural languages, recurrent neural networks gained popularity in NLP tasks. Active research of different architectures led to great advances and eventually a shift towards methods using the transformer architecture [Vaswani et al., 2017] or convolutional layers [Bai et al., 2018; van den Oord et al., 2017a; van den Oord et al., 2017b] among researchers and practitioners alike. In this work, we focus on a lightweight convolutional architecture that creates fixed-size representations of sentences.

Convolutional neural networks have the inherent ability to detect local structures in the data. In the context of conversational systems, their speed and memory efficiency eases deployment on mobile devices, allowing fast response retrieval and better user experience. We analyze and build on the recently proposed Byte-Level Recursive Convolutional Auto-Encoder (BRCA) for text paragraphs [Zhang and LeCun, 2018], which is able to auto-encode text paragraphs into fixed-size vectors, reading in bytes with no additional preprocessing.

Based on our analysis, we are able to explain the behavior of this model, point out possible enhancements, achieve auto-encoding accuracy improvements and an order of magnitude training speed-up, while cutting down the number of parameters by over 70%. We introduce a balanced padding scheme for input sequences and show that it significantly improves convergence and capacity of the model. As we find byte-level encoding unsuitable for embedding sentences, we demonstrate its applicability in processing sentences at word-level. We train the encoder with supervision on Stanford Natural Language Inference corpus [Bowman et al., 2015; Conneau et al., 2017] and investigate its performance on various transfer tasks to assess quality of produced embeddings.

The paper is structured as follows: in Section 2 we introduce some of the notions that appear in the paper. Section 3 discusses relevant work for sentence vector representations. Details of the architecture can be found in Section 4. Section 5 presents the analysis of the auto-encoder and motivations for our improvements. Section 6 demonstrates supervised training of word-level sentence encoder, and evaluates it on tasks relevant to conversational systems. Section 7 concludes the paper.

2 Preliminaries
An information retrieval conversational agent selects a response from a fixed set. Let \( D = \{(i_k, r_k)\}_{k} \) be a set of conversational input-response pairs, and \( q \) be a current user’s input. Two simple ways of retrieving a response \( r_k \) from available data are [Ritter et al., 2011; Chorowski et al., 2018]:
• return \( r_k \) most similar to user’s input \( q \).
• return \( r_k \) for which \( i_k \) is most similar to \( q \).
Utterances \( i_k, r_k, q \) may be represented (embedded) as real-valued vectors.

Many NLP systems represent words as points in a continuous vector space using word embedding methods [Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016]. They are calculated based on co-occurrence of words in large corpora. The same methods were applied to obtain sentence embeddings only with partial success, due to the combinatorial explosion of all possible word combinations, which make up a sentence. Instead, Recurrent Neural Networks (RNNs), autoregressive models that can process input sequences of an arbitrary length, are thought to be a good method for handling variable-length textual data, with Long Short-Term Memory network (LSTM) [Hochreiter and Schmidhuber, 1997] being the prime example of such.

Recently, RNNs have been reported to be successfully replaced by convolutional architectures [Bai et al., 2018; van den Oord et al., 2017a; van den Oord et al., 2017b]. Convolutional neural networks are traditionally associated with computer vision and image processing [Krizhevsky et al., 2012; Redmon et al., 2015]. They primarily consist of convolutional layers that apply multiple convolutions to the input, followed by pooling layers that are used for reducing the dimensionality of the hidden state. Convolutional networks are efficient during training and inference: they utilize few parameters and do not require sequential computations, making hardware parallelism easy to use. Due to their popularity in image processing, there are efficient implementations that scale well.

Residual connection [He et al., 2015] is a connection that adds an unchanged input to the output of the layer or block of layers. During the forward pass it provides upper layers with undistorted signal from the input and intermediate layers. During the backward pass it mitigates vanishing and exploding gradient problems [Hochreiter et al., 2001].

Batch Normalization [Ioffe and Szegedy, 2015] (BN) applies normalization to all activations in every minibatch. Typical operations used in neural networks are sensitive to changes in the range and magnitude of inputs. During training the inputs to upper layers vary greatly due to changes in weights of the model. Normalization of the signal in each layer has the potential to alleviate this problem. Both BN and residual connections enable faster convergence by helping with forward and backward flow of information. They were also crucial in training our models.

3 Related Work

There are many methods for creating sentence embeddings, the simplest being averaging word-embedding vectors of a given sentence [Joulin et al., 2016; Le and Mikolov, 2014]. SkipThought [Kiros et al., 2015] generalizes idea of unsupervised learning of word2vec word embeddings [Mikolov et al., 2013]. It is implemented in the encoder-decoder setting using LSTM networks. Given a triplet of consecutive sentences \( (s_{i-1}, s_i, s_{i+1}) \), the encoder creates a fixed-size embedding vector of the sentence \( s_i \), and the decoder tries to generate sentences \( s_{i-1} \) and \( s_{i+1} \) from this representation. SkipThought vectors have been shown to preserve syntactic and semantic properties [Kiros et al., 2015], so that their similarity is represented in the embedding space.

InferSent model [Conneau et al., 2017] shows that training embedding systems with supervision on a natural language inference task may be superior to an unsupervised training. Recently, better results were obtained by combining supervised learning on an auxiliary natural language inference corpus with learning to predict the correct response on a conversational data corpus [Yang et al., 2018].

4 Model Description

Our model builds on the Byte-Level Recursive Convolutional Auto-Encoder (BRCA) and our model. Dark boxes indicate input padding. BRCA pads the input form the right to the nearest power of two. We pad the input evenly to a fixed-size vector. Our model does not have postfix/prefix groups with linear layers and uses Batch Normalization (BN) after every layer.

![Figure 1: Structural comparison of Byte-Level Recursive Convolutional Auto-Encoder (BRCA) and our model. Dark boxes indicate input padding. BRCA pads the input form the right to the nearest power of two. We pad the input evenly to a fixed-size vector. Our model does not have postfix/prefix groups with linear layers and uses Batch Normalization (BN) after every layer.](image)
consists of \(N\) temporal convolutional layers followed by a max-pool layer with kernel size 2. For each sequence, the prefix group is applied only once, while the recursive group is applied multiple times, sharing weights between all applications. All convolutional layers have \(d = 256\) channels, kernels of size 3 and are organized into residual blocks, with 2 layers per block, ReLU activations, residual connections, and Batch Normalization (see Section 5.5 for details).

The encoder of our model reads in text, by sentence or by paragraph, as a sequence of discrete tokens (e.g. bytes, characters, words). Each input token is embedded as a fixed-length \(d\)-dimensional vector. Unlike [Zhang and LeCun, 2018], where input sequence is zero-padded to the nearest power of 2, we pad the input to a fixed length \(2^K\), distributing the characters evenly across the input. We motivate this decision by the finding that the model zero-padded to the nearest power of 2 does not generalize to sentences longer than those seen in the training data (see Section 5.2).

First, the prefix group is applied, which retains the dimensionality and the number of channels, i.e., sequence length and embedding size. Let \(d \cdot 2^r\) be the dimensionality of the latent code output by the encoder. The encoder then applies the recursive group \(K - r\) times. Note that with \(r\) one may control size of a latent vector (the level of the compression). With the max-pooling, every application halves the length of the sequence. Weights are shared between applications. Finally, the encoder outputs a latent code of size \(d \cdot 2^r\). Unlike [Zhang and LeCun, 2018], we do not apply any linear layers after recursions. Our experiments have shown that they slightly degrade the performance of the model, and constitute the majority of its parameters.

### 4.2 Decoder

The decoder acts in reverse. First, it applies a recursive group consisting of a convolutional layer which doubles the number of channels to \(2d\), which is followed by an expand transformation [Zhang and LeCun, 2018] and \(N - 1\) convolutional layers. Then it applies a postfix group of \(N\) temporal convolutional layers. Similarly to the encoder, the layers are organized into residual blocks with ReLU activations, Batch Normalization, and have the same dimensionality and kernel size. We double the size of input in the residual connection, which bypasses the first two convolutions of the recursive group, by stacking it with itself. We found it crucial for convergence speed to use only residual blocks in the network, also in the expand block.

The decoder applies its recursive group \(K - r\) times. Each application doubles the numbers of channels, while the expand transformation reorders and reshapes the data to effectively double the length of the sequence while the number of channels is unchanged and equals \(d\). The postfix group processes a tensor of size \(2^K \times d\) and retains its dimensionality. The output is interpreted as \(2^K\) probability distributions over possible output elements. Adding an output embedding layer, either tied with input embedding layer or separate, slowed down training and did not improve the results. At the end, a Softmax layer is used to compute output probabilities over possible bytes. Note that output probabilities are independent from one another conditioned on the input.

### 5 Model Analysis

In this section we justify our design choices through a series of experiments with the BRCA model and report their outcomes.\(^1\)

#### 5.1 Data

In order to produce comparable results, we prepared an English Wikipedia dataset with similar sentence length distribution to [Zhang and LeCun, 2018]. Namely, we took at random 11 million sentences from an English Wikipedia dump extracted with WikiExtractor\(^2\), so that their length distribution would roughly match that of [Zhang and LeCun, 2018] (Table 1). In experiments with random data, we generate random strings of a-zA-Z0-9 ASCII characters.

<table>
<thead>
<tr>
<th>Length</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-63 B</td>
<td>35%</td>
</tr>
<tr>
<td>64-127 B</td>
<td>14%</td>
</tr>
<tr>
<td>128-255 B</td>
<td>20%</td>
</tr>
<tr>
<td>256-511 B</td>
<td>18%</td>
</tr>
<tr>
<td>512-1023 B</td>
<td>14%</td>
</tr>
</tbody>
</table>

#### 5.2 Model Capacity

Natural language texts are highly compressible due to their low entropy, which results from redundancy of the language [Levitin and Reingold, 1994]. In spite of this, the considered models struggle to auto-encode 1024-byte short paragraphs into 1024-float latent vectors, which are 4096-byte given their sheer information content. Transition from discrete to continuous representation and inherent inefficiency of the model are likely to account for some of this overhead.

One can imagine an initialization of weights that, given the over-capacitated latent representation, would make the network perform identity for paragraphs up to 128 bytes long\(^3\). We confirmed those speculations experimentally, training models on paragraphs of random printable ASCII characters, namely random strings of a-zA-Z0-9 symbols (Table 2). The empirical capacity of our model is 128 bytes, which sheds light on the amount of overhead. This model has to be trained on paragraphs longer than 512 bytes in order to learn useful, compressing behavior given a 1024-float latent representation.

#### 5.3 Generalization to Longer Sequences

Auto-encoding RNN models such as LSTM are known to deteriorate gradually with longer sequences [Cho et al., 2014; Chorowski et al., 2015]. We trained a BRCA model \((N = 2)\) and an LSTM encoder-decoder network with hidden size 256.

---

\(^1\)Source code of our models is available: https://github.com/smalik169/recursive-convolutional-autoencoder

\(^2\)https://github.com/attardi/wikiextractor

\(^3\)When max-pooling is replaced by convolution with stride 2 and kernel size 2
Table 2: Learning identity by training on random sequences of ASCII characters of different length. Accuracy is presented for BRCA (N=8) model.

<table>
<thead>
<tr>
<th>Training Lengths</th>
<th>Test Length</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 – 128</td>
<td>128</td>
<td>99.81%</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>60.79%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>22.99%</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>9.81%</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the ability of BRCA and LSTM encoder-decoder to learn an identity function and generalize to unseen data. Values represent byte-level decoding accuracy. Note that the LSTM decoder has the advantage of always being primed with the correct prefix sequence.

<table>
<thead>
<tr>
<th>Lengths (bytes)</th>
<th>BRCA (N=2)</th>
<th>LSTM-LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>9-16</td>
<td>97.06%</td>
<td>91.17%</td>
</tr>
<tr>
<td>17-32</td>
<td>97.96%</td>
<td>90.20%</td>
</tr>
<tr>
<td>33-64</td>
<td>97.45%</td>
<td>91.72%</td>
</tr>
<tr>
<td>65-128</td>
<td>83.56%</td>
<td>86.34%</td>
</tr>
<tr>
<td>129-256</td>
<td>11.66%</td>
<td>72.88%</td>
</tr>
<tr>
<td>257-512</td>
<td>8.05%</td>
<td>58.80%</td>
</tr>
</tbody>
</table>

Both models were trained on sentences of length up to 128 bytes and evaluated on unseen data. The LSTM model did not perfectly learn the identity function, even though it was solving an easier task of predicting the character given the correct prefix. However, the LSTM model generalized much better on longer sequences, where performance of BRCA deteriorated rapidly (Table 3).

5.4 Balanced Padding of Input Sequences

We found BRCA difficult to train. The default hyperparameters given by the authors [Zhang and LeCun, 2018] are single-sample batches, SGD with momentum 0.9, and a small learning rate 0.01 with 100 epochs of training. In our preliminary experiments, increasing the batch size by batching paragraphs of the same length improved convergence on datasets with short sentences (mostly up to 256 bytes long), but otherwise deteriorated on the Wikipedia dataset, where roughly 50% paragraphs are longer than 256 bytes. We suspect that the difficulty lies in the difference of the underlying tasks: long paragraphs require compressive behavior, while short ones merely require learning the identity function. Updating network parameters towards one task hinders the performance on the others, hence the necessity for careful training.

In order to blend in both tasks, we opted for padding input sequences into fixed-length vectors. We find it sensible to fix maximum length of input sentence, since the model does not generalize to unseen lengths anyway. Variable length of input in BRCA does save computations, however we found fixing input size to greatly improve training time, despite the overhead.

In order to make the tasks more similar, we propose balanced padding of the inputs (Figure 2). Instead of padding from the right up to $2^k$ bytes, we pad to the nearest power of 2 and distribute the remaining padding equally in between the bytes. We hypothesized that it could free convolutional layers from the burden of propagating the signal from left to right in order to fill the whole latent vector, as it would be the case, e.g., when processing a 64-byte paragraph padded with 960 empty tokens from the right to form 1024-byte input. Empirically, this trades additional computations for better convergence characteristics.

5.5 Batch Normalization

Fixed-length, balance-padded inputs allow easy mixing of paragraphs of different lengths within a batch, in consequence allowing raising the batch size, applying Batch Normalization and raising the learning rate. This enables a significant speed-up in convergence and better auto-encoding accuracy (see Section 5.6). However, the statistics collected by BN layers differ during each of the $K-r$ recursive steps, even though the weights of convolutions in the recursive layers are shared. This breaks auto-encoding during inference time, when BN layers have fixed mean and standard deviation collected over a large dataset. We propose to alleviate this issue by either: a) collecting separate statistics for each recursive application and each input length separately, or b) placing a paragraph inside a batch of data drawn from the training corpus during inference and calculating the mean and the standard deviation on this batch. We also experimented with the instance normalization [Ulyanov et al., 2016], which performs the normalization of features of a single instances, rather than of a whole minibatch. We have found that the instance normalization improved greatly upon the baseline model with no normalization, but performed worse than batch normalization.

BRCA has been introduced with linear layers in the postfix/prefix groups of the encoder/decoder. In our experiments, removing those layers from the vanilla BRCA lowered accuracy by a few percentage points. Conversely, our model benefits from not having linear layers. We observed faster convergence and better accuracy without them, while reducing the number of parameters from 23.4 million to 6.67 million.

5.6 Auto-Encoding Performance

Our training setup is comparable with that of BRCA [Zhang and LeCun, 2018]. In each epoch, we randomly select 1 million sentences from the training corpus. We trained using SGD with momentum 0.5 in batches of 32 paragraphs of random length, balanced padded to $2^r = 1024$ tokens, including
5.7 Generalization

We investigated which inputs influence correct predictions of the network using the method of Integrated Gradients [Sundararajan et al., 2017]. We have produced two heatmaps of input-output relationships for short (128 bytes) and long (1024 bytes) paragraphs in our best model (Figure 3). In theory, a model performing identity should have a diagonal heatmap. Our model finds relations within bytes of individual words, rarely crossing word and phrases boundaries. In this sense, it fails to exploit the ordering of words. However, the order is mostly preserved in the latent vector.

Early in the training the model learns to output only spaces, which are the most common bytes in an average Wikipedia paragraph. Later during training, it learns to correctly rewrite spaces, while filling in the words with vowels, which are the most frequent non-space characters. Interestingly, the compressing behavior seems to be language-specific and triggered only by longer sequences. Figure 5 presents input sentences...
Figure 5: Auto-encoding capabilities of the model with errors marked in **bold red**. The model was trained only on English Wikipedia paragraphs. On short sequences, our model performs close to an identity function. On longer ones, it seems to correctly auto-encode only 64 pre-trained GloVe vectors applied K times where $m$ is the length of the sentence, $w_i$ is its $i$-th word, and $e(w)$ is the GloVe embedding of the word $w$. Final embedding is the sum $x = v + u$.

Table 4 presents results for word-level recursive convolutional encoder (WRCE), word-level model with fixed balanced padding (Ours), and an ensemble of our model and an average embedding of the input sequence (Ours + BoW). We compare them with a baseline model (BoW - average of GloVe vectors for words in a sentence) on SNLI and other classification tasks, SICK-Relatedness [Marelli et al., 2014], and STS[12-16] tasks. The SentEval tool was used for these experiments.

For certain tasks, especially those measuring textual similarity, which are useful in retrieval-based response generation in dialogue systems, presented models perform better than bag-of-words. However, they are still not on par with LSTM-based methods [Conneau et al., 2017; Kiros et al., 2015] that generate more robust embeddings. LSTM models are autoregressive and thus require slow sequential computations. They are also larger, with the InferSent model [Conneau et al., 2017] having over 30 times more parameters than convolutional encoders presented in this section. In addition, our architecture can share word embedding matrices with other components of a conversational system, since word embeddings are ubiquitous in different modules of NLP systems.

In order to qualitatively assess how the results for those tasks transfer to the actual dialogue system, we compared some retrieved responses of a simple retrieval-based

---

Figure 5: Auto-encoding capabilities of the model with errors marked in **bold red**. The model was trained only on English Wikipedia paragraphs. On short sequences, our model performs close to an identity function. On longer ones, it seems to correctly auto-encode only 64 pre-trained GloVe vectors applied $K$ times where $m$ is the length of the sentence, $w_i$ is its $i$-th word, and $e(w)$ is the GloVe embedding of the word $w$. Final embedding is the sum $x = v + u$.

Table 4 presents results for word-level recursive convolutional encoder (WRCE), word-level model with fixed balanced padding (Ours), and an ensemble of our model and an average embedding of the input sequence (Ours + BoW). We compare them with a baseline model (BoW - average of GloVe vectors for words in a sentence) on SNLI and other classification tasks, SICK-Relatedness [Marelli et al., 2014], and STS[12-16] tasks. The SentEval tool was used for these experiments.

For certain tasks, especially those measuring textual similarity, which are useful in retrieval-based response generation in dialogue systems, presented models perform better than bag-of-words. However, they are still not on par with LSTM-based methods [Conneau et al., 2017; Kiros et al., 2015] that generate more robust embeddings. LSTM models are autoregressive and thus require slow sequential computations. They are also larger, with the InferSent model [Conneau et al., 2017] having over 30 times more parameters than convolutional encoders presented in this section. In addition, our architecture can share word embedding matrices with other components of a conversational system, since word embeddings are ubiquitous in different modules of NLP systems.

In order to qualitatively assess how the results for those tasks transfer to the actual dialogue system, we compared some retrieved responses of a simple retrieval-based

---

**6 Word-Level Sentence Encoder**

Following the methods and work of [Conneau et al., 2017], we apply our architecture to a practical task. Namely, we train models consisting of the recursive convolutional word-level encoder and a simple three-layer fully-connected classifier on Stanford Natural Language Inference (SNLI) corpus [Bowman et al., 2015]. This dataset contains 570k sentence pairs, each one described by one of three relation labels: entailment, contradiction, and neutral. Then we test encoders on various transfer tasks measuring semantic similarities between sentences.

The encoder of each model has a similar architecture to the previously described byte-level encoder. However, instead of bytes it takes words as its input sequence. Our best encoder has $N = 8$ layers in each group. The recursive group is applied $K$ times where $2^K$ is length of a padded input sequence, so that the latent vector is of the size of a word vector. We use pre-trained GloVe vectors\(^4\) and we do not fine-tune them. We compared both fixed-length balanced, and variable length input paddings. In fixed-length padding, up to first 64 words are taken from each sentence. We also compare ensemble of our best trained model and bag-of-words as a sentence representation. Let $v$ be the output vector of the encoder, and $u = \frac{1}{m} \sum_{i=1}^{m} e(w_i)$ be the average of word vectors of the sentence, where $m$ is the length of the sentence, $w_i$ is its $i$-th word, and $e(w)$ is the GloVe embedding of the word $w$. Final embedding is the sum $x = v + u$.

Table 4 presents results for word-level recursive convolutional encoder (WRCE), word-level model with fixed balanced padding (Ours), and an ensemble of our model and an average embedding of the input sequence (Ours + BoW). We compare them with a baseline model (BoW - average of GloVe vectors for words in a sentence) on SNLI and other classification tasks, SICK-Relatedness [Marelli et al., 2014], and STS[12-16] tasks. The SentEval tool was used for these experiments.

For certain tasks, especially those measuring textual similarity, which are useful in retrieval-based response generation in dialogue systems, presented models perform better than bag-of-words. However, they are still not on par with LSTM-based methods [Conneau et al., 2017; Kiros et al., 2015] that generate more robust embeddings. LSTM models are autoregressive and thus require slow sequential computations. They are also larger, with the InferSent model [Conneau et al., 2017] having over 30 times more parameters than convolutional encoders presented in this section. In addition, our architecture can share word embedding matrices with other components of a conversational system, since word embeddings are ubiquitous in different modules of NLP systems.

In order to qualitatively assess how the results for those tasks transfer to the actual dialogue system, we compared some retrieved responses of a simple retrieval-based

---

**6 Word-Level Sentence Encoder**

Following the methods and work of [Conneau et al., 2017], we apply our architecture to a practical task. Namely, we train models consisting of the recursive convolutional word-level encoder and a simple three-layer fully-connected classifier on Stanford Natural Language Inference (SNLI) corpus [Bowman et al., 2015]. This dataset contains 570k sentence pairs, each one described by one of three relation labels: entailment, contradiction, and neutral. Then we test encoders on various transfer tasks measuring semantic similarities between sentences.

The encoder of each model has a similar architecture to the previously described byte-level encoder. However, instead of bytes it takes words as its input sequence. Our best encoder has $N = 8$ layers in each group. The recursive group is applied $K$ times where $2^K$ is length of a padded input sequence, so that the latent vector is of the size of a word vector. We use pre-trained GloVe vectors\(^4\) and we do not fine-tune them. We compared both fixed-length balanced, and variable length input paddings. In fixed-length padding, up to first 64 words are taken from each sentence. We also compare ensemble of our best trained model and bag-of-words as a sentence representation. Let $v$ be the output vector of the encoder, and $u = \frac{1}{m} \sum_{i=1}^{m} e(w_i)$ be the average of word vectors of the sentence, where $m$ is the length of the sentence, $w_i$ is its $i$-th word, and $e(w)$ is the GloVe embedding of the word $w$. Final embedding is the sum $x = v + u$.

Table 4 presents results for word-level recursive convolutional encoder (WRCE), word-level model with fixed balanced padding (Ours), and an ensemble of our model and an average embedding of the input sequence (Ours + BoW). We compare them with a baseline model (BoW - average of GloVe vectors for words in a sentence) on SNLI and other classification tasks, SICK-Relatedness [Marelli et al., 2014], and STS[12-16] tasks. The SentEval tool was used for these experiments.

For certain tasks, especially those measuring textual similarity, which are useful in retrieval-based response generation in dialogue systems, presented models perform better than bag-of-words. However, they are still not on par with LSTM-based methods [Conneau et al., 2017; Kiros et al., 2015] that generate more robust embeddings. LSTM models are autoregressive and thus require slow sequential computations. They are also larger, with the InferSent model [Conneau et al., 2017] having over 30 times more parameters than convolutional encoders presented in this section. In addition, our architecture can share word embedding matrices with other components of a conversational system, since word embeddings are ubiquitous in different modules of NLP systems.

In order to qualitatively assess how the results for those tasks transfer to the actual dialogue system, we compared some retrieved responses of a simple retrieval-based

---

\(^4\)https://nlp.stanford.edu/projects/glove/

---

\(^5\)https://github.com/facebookresearch/SentEval
Table 4: Results for word-level sentence encoders. We compare bag-of-words (BoW), i.e. averaged word embeddings, WRCE - the encoder from Zhang and LeCun’s model on word-level, our word-level model with balanced padding to 64 elements (Ours), and an ensemble of our model and BoW (Ours + BoW) for various supervised (classification accuracy) and unsupervised (Pearson/Spearman correlation coefficients) tasks.

<table>
<thead>
<tr>
<th>Model</th>
<th>SNLI (dev/test acc%)</th>
<th>CR (dev/test acc%)</th>
<th>MR (dev/test acc%)</th>
<th>MPQA (dev/test acc%)</th>
<th>SUBJ (dev/test acc%)</th>
<th>SST Bin. Class. (dev/test acc%)</th>
<th>SST Fine-Grained Class. (dev/test acc%)</th>
<th>TREC (dev/test acc%)</th>
<th>MRPC (dev/test acc%)</th>
<th>SICK-E (dev/test acc%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoW</td>
<td>67.7 / 67.5</td>
<td>79.7 / 78.0</td>
<td>77.7 / 77.0</td>
<td>87.4 / 87.5</td>
<td>91.8 / 91.4</td>
<td>74.5 / 82.2</td>
<td>45.1 / 44.4</td>
<td>74.4 / 73.2</td>
<td>74.4 / 73.2</td>
<td>79.8 / 78.2</td>
</tr>
<tr>
<td>WRCE</td>
<td>82.0 / 81.3</td>
<td>78.0 / 77.3</td>
<td>72.9 / 72.4</td>
<td>85.9 / 85.6</td>
<td>86.1 / 85.4</td>
<td>67.0 / 72.4</td>
<td>38.3 / 40.5</td>
<td>72.4 / 71.1</td>
<td>73.7 / 72.5</td>
<td>82.6 / 82.8</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>83.1</strong></td>
<td><strong>83.1</strong></td>
<td><strong>73.1</strong></td>
<td><strong>86.0</strong></td>
<td><strong>87.2</strong></td>
<td><strong>69.2</strong></td>
<td><strong>40.5</strong></td>
<td><strong>71.1</strong></td>
<td><strong>73.7</strong></td>
<td><strong>83.6</strong></td>
</tr>
<tr>
<td>Ours + BoW</td>
<td>83.2 / 82.6</td>
<td>83.1 / 82.6</td>
<td>75.3 / 74.8</td>
<td>89.0 / 88.9</td>
<td>89.0 / 88.9</td>
<td>71.0 / 77.4</td>
<td>41.9 / 41.4</td>
<td>74.1 / 73.3</td>
<td>75.3 / 74.8</td>
<td>83.2 / 83.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task (correlation)</th>
<th>BoW</th>
<th>WRCE</th>
<th>Ours</th>
<th>Ours + BoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>SICK-R</td>
<td>0.80 / 0.72</td>
<td>0.85 / 0.78</td>
<td><strong>0.87 / 0.80</strong></td>
<td>0.86 / 0.80</td>
</tr>
<tr>
<td>STS12</td>
<td>0.53 / 0.54</td>
<td>0.56 / 0.57</td>
<td>0.60 / 0.60</td>
<td><strong>0.62 / 0.61</strong></td>
</tr>
<tr>
<td>STS13</td>
<td>0.45 / 0.47</td>
<td>0.55 / 0.54</td>
<td>0.53 / 0.54</td>
<td><strong>0.57 / 0.58</strong></td>
</tr>
<tr>
<td>STS14</td>
<td>0.53 / 0.54</td>
<td>0.65 / 0.63</td>
<td>0.68 / 0.70</td>
<td><strong>0.69 / 0.66</strong></td>
</tr>
<tr>
<td>STS15</td>
<td>0.56 / 0.59</td>
<td>0.68 / 0.69</td>
<td>0.70 / 0.70</td>
<td><strong>0.71 / 0.72</strong></td>
</tr>
<tr>
<td>STS16</td>
<td>0.52 / 0.57</td>
<td>0.69 / 0.70</td>
<td>0.70 / 0.72</td>
<td><strong>0.71 / 0.73</strong></td>
</tr>
</tbody>
</table>

agent, which matches user utterance with a single quote from Wikiquotes [Chorowski et al., 2018]. We present a comparison of our word-level sentence encoder with the bag-of-word method in response retrieval task (Figure 6). Human utterances from the training data of NIPS 2017 Conversational Challenge have been selected as input utterances. We match them with the closest quote from Wikiquotes, using a method similar to the one used in Poetwannabe chatbot [Chorowski et al., 2018]. All utterances have been filtered for foul speech (for details see [Chorowski et al., 2018]), tokenized using Moses tokenizer, and embedded as vectors. For every user input utterance, we return the closest matching quote from Wikiquotes with respect to the cosine similarity.

7 Discussion and Future Work

The problem of efficiently producing good and robust sentence embeddings seems far from being solved. In this work, which we consider a step forward in exploration of possible tools and solutions, we analyzed and proposed improvements to the BRCA model by Xiang Zhang and Yann LeCun [2018]. With extensive usage of convolutions, our model is efficient in terms of computation and memory usage. By analyzing BRCA we were able to diagnose problems with its training, such as exploding gradients, and understand the difficulty in auto-encoding of long paragraphs, especially in the initial stage of training. Furthermore, we showed how to successfully apply batch normalization with recursive layers and investigate input-output relations with Integrated Gradients method.

The recursive convolutional architecture benefits from the ease of training and low number of parameters. Due to our realization that in the current byte-level setting, input-output relations rarely cross word boundaries, we demonstrate applicability of the architecture in a word-level setting as a sentence embedder. Furthermore, a good performance on semantic similarity tasks while using little resources demonstrates its practical usefulness for dialog systems.

Acknowledgments

The authors thank Paweł Rychlikowski and Michał Zapotoczny for fruitful discussions, and Xiang Zhang for help towards a better understanding of the original model. The research was partially supported by National Science Center (Poland) grant Sonata 8 2014/15/D/ST6/04402.
User: Hello, have you studied algebra?
BoW: I never failed in mathematics. Before I was fifteen I had mastered differential and integral calculus.
Ours+BoW: I never failed in mathematics. Before I was fifteen I had mastered differential and integral calculus.
User: From what I understand it’s a video game.
BoW: Life is like a piano. What you get out of it depends on how you play it.
Ours+BoW: From now on, I control all aspects of the game.
User: God
BoW: Knowledge of God is obedience to God.
Ours+BoW: God and all attributes of God are eternal.
User: Have you been to South Asia?
BoW: We do not want chaos in South Africa.
Ours+BoW: There is a wide knowledge gap between us and the developed world in the West and in Asia.
User: How do you like Hackathon?
BoW: Any thing you can do I can do better.
Ours+BoW: Discover the things that you do that make you proud as hell.
User: How do you think ?
BoW: Any thing you can do I can do better.
Ours+BoW: How you think matters more than what you think.
User: I just didn’t catch what was context message about.
BoW: I read it because I was asked to explain what the truth is.
Ours+BoW: I spend so much time thinking about all the answers to the problem... that I forget what the problem actually was.
User: I’m an idiot
BoW: I am an Agnostic because I am not afraid to think.
Ours+BoW: I wish I could say I was sorry.
User: It’s classics!
BoW: I love musical theatre and my dream is to do Once On This Island.
Ours+BoW: No work which is destined to become a classic can look like the classics which have preceded it.
User: So, start talking.
BoW: Oh, ok, ok... Fair enough, yeah, rage it up. Rage all you want. Good things are coming. Good things.
Ours+BoW: Many people talk much, and then very many people talk very much more.
User: Technically correct
BoW: Surely only correct understanding could lead to correct action.
Ours+BoW: Where an opinion is general, it is usually correct.
User: Thats why I play computer games alone.
BoW: I have no time to play games.
Ours+BoW: The only legitimate use of a computer is to play games.
User: Well, can you dance?
BoW: If I can mince , you can dance.
Ours+BoW: Ah, so you wish to dance.
User: What about ivy league?
BoW: Ah wonder if anybody this side of the Atlantic has ever bought a baseball bat with playing baseball in mind.
Ours+BoW: This is so far out of my league.

Figure 6: Sample answers of retrieval-based agents which embed sentences as either BoWs, or BoWs combined with our method

References


[van den Oord et al., 2017b] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals,


Event Data Collection for Recent Personal Questions

Masahiro Mizukami, Hiroaki Sugiyama, Hiromi Narimatsu
NTT Communication Science Laboratories
{mizukami.masahiro, sugiyama.hiroaki, narimatsu.hiromi}@lab.ntt.co.jp

Abstract
In human-human conversation, people frequently ask questions about a person with whom to talk. Since such questions also asked in human-agent conversations, previous research developed a Person DataBase (PDB), which consists of question-answer pairs evoked by a pre-defined persona to answer user’s questions. PDB contains static information including name, favorites, and experiences. Therefore, PDB cannot answer questions about events that occurred after it was built. It means that this approach does not focus on answering questions about more recent things (recent personal questions), e.g., Have you seen any movies lately? In contrast, since recent questions are frequently asked in a casual conversation, conversational agents are required to answer recent questions for maintaining a conversation. In this paper, we collect event data that consist of a large number of experiences and behaviors in daily lives, which enables to answer recent questions. We analyze them and show that our data is effective for answering recent questions.

1 Introduction
Questions about a conversational partner are called “personal question,” which are an essential factor for expressing interest in conversational partners. Such questions frequently occur in casual human-human conversations. Nishimura et al. showed that such questions occurred in both human-human and human-agent conversations [Nishimura et al., 2003]. Adequately answering them is an essential factor in the development of conversational agents [Sugiyama et al., 2017].

To answer personal questions, previous works developed Person DataBase (PDB), which consists of question-answer pairs evoked by a pre-defined persona [Batacharia et al., 1999; Sugiyama et al., 2014]. Although their approach covers a wide variety of personal questions, developing a high-quality PDB is too expensive. The cost problem makes it difficult to update constantly; consequently, PDB usually contains only static information that rarely changes over time. Therefore, conversational agents using PDB cannot answer questions about recent events such as What did you have for dinner yesterday?. Also, it is easy to imagine that immutable responses to recent personal questions make conversational agents unnatural; therefore, conversational agents have to spend different days that like people spend different days, and it is more natural to return different answers to recent personal questions. To solve this problem, preparing other kinds of data which expresses recent experiences helps conversational agents to answer such questions about recent things (recent personal questions).

One simple idea is to collect data that express such recent events as a diary that is updated by the user. Previous work on response generation leveraged diaries or microblogs as a corpus that includes people’s recent personal information [Li et al., 2016]. Even though this approach seems reasonable, handcrafted-data-driven approach such as PDB has practical advantages in controllability and reliability. In this paper, we collected event data from participants who take part in short- and long-term periods. This collected data is hand-crafted, high-quality and easy to update (adding new day’s data). We clarified the potential of event data to answer questions about recent behaviors/experiences in casual conversations through analysis.

2 Related Works
As mentioned in the introduction, PDB is the most closely related research to answer user questions. Batacharia et al. developed PDB about Catherine, a 26-year-old female living in New York City [Batacharia et al., 1999]. To cover more questions and with different personas, Sugiyama et al., developed a PDB with six personalities such as a 20-year-old female, a 50-year-old male, and robots [Sugiyama et al., 2014]. Both PDBs contain only static information; therefore, they cannot answer recent questions. If we want to answer recent questions by PDB, we have to update PDB’s contents constantly; however, updating PDBs constantly causes too expensive costs. The difficulty of updating PDB is the relationship of questions and answers (QA); for example, when the content of a base QA changes (e.g., Question: Do you have any pets?, Answer: Yes, I have a dog. change to new Answer: No, I don’t have.), related contents of QAs should be changed depending on a changed content of base QA (e.g., Question: Do you have a dog?, Answer: Yes, I have a dog. should be changed to new Answer: No, I don’t have.). PDB has many complicated relations of QAs, it makes updating
PDB difficult and expensive. A PDB’s merit, which is found in handcrafted-data-driven methods, is the ability to generate answers based on facts and consistency from the data. Such handcrafted-data-driven approaches answer questions with consistent replies and without a lie. The consistency of the responses based on facts has the potential for improving the performance of conversational agents.

Although there are many studies on conversational agent’s response generation [Ritter et al., 2011; Inaba and Takahashi, 2016], few studies focus on the consistency depending on an agent’s personality. Persona-based conversation models treat personality as speaker-embedding to increase the sentence quality [Li et al., 2016]. This model is the state-of-the-art model to generate conversational agent’s responses using an embedding vector that expresses agent’s personality. This approach has potential to answer recent personal questions; however, it indicates two critical problems. One is that this approach cannot promise to answer without lies: this problem is strongly related with research of PDB. Hand-crafted database approach such as PDB can answer responses that reflected a right personality unless it gets wrong matching of questions. In contrast, neural network based approaches often answer questions with response sentences that do not exist in training data; since such models are optimized only for maximizing the naturalness of response sentences. Even though this approach has the potential to answer recent personal questions, it can offer no guarantee that the answers exist in training data.

Another problem is that this model does not consider the past consistency depended on the day, time, and past events. When we asked a question such as What did you eat last night? to conversational agents, this approach always replies the same response such as I ate ramen. This QA pair is natural when we check only this one pair; however, eating the same food every dinner is too unnatural in the daily life of conversational agents. Therefore, to establish a long-term conversation with human-agent and to make conversational agents more natural, we must solve this invariance problem of responses. Using information of date and time to train speaker-embedding vector, it may help to solve this problem. However, we can imagine easily that this model requires much training data which is insufficient with the amount we have now.

In this paper, we created event data for answering recent personal questions in casual conversations. This approach by the created event data is identical with PDB as the handcrafted-data-driven approach and is essential to verify answering based on facts, and we use it as the first step to develop a function that answers questions about recent experiences based on facts.

### 3 Data Collection

To answer recent personal questions that ask about recent experiences and behaviors, we collect the consistent data from humans as events that express experiences and behaviors. This event data has to be collected from participants with low-costs; because we need to update it constantly. Besides, we have to collect data from various participants because we do not know what kind of persona influences events.

We recruited 62 Japanese-speaking participants of roughly equal numbers of both genders whose ages ranged from 10s to 60s and collected daily experiences and behaviors as event data. They wrote down 20 events every day and at least two events every four hours. We collect event name, reasons, time, and impressions for each event; because these aspects are asked in casual conversation frequently. Participants were indicated not to write any descriptions including privacy. Such diary-like method to write down like a diary is low-cost compare than the PDB’s collecting method. Specifically, we prepare an Excel file and ask participants to write four aspects such as name, reasons, time, and impressions,

<table>
<thead>
<tr>
<th>Event name</th>
<th>Event reason</th>
<th>Event time</th>
<th>Event impressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Played a mobile game game</td>
<td>Habit before going to bed, To get daily bonus</td>
<td>08:00-12:00</td>
<td>Happy</td>
</tr>
<tr>
<td>Read a novel by a mobile phone</td>
<td>Habit before going to bed, To induce sleep</td>
<td>09:00-12:00</td>
<td>Fun, Sleepy</td>
</tr>
<tr>
<td>Got up</td>
<td>To prepare a lunch box</td>
<td>16:00-18:00</td>
<td>Sleepy, Tired</td>
</tr>
<tr>
<td>Went to sleep</td>
<td>To rest before going to office</td>
<td>20:00-22:00</td>
<td>Sleepy</td>
</tr>
<tr>
<td>Got up</td>
<td>To go to office</td>
<td>22:00-24:00</td>
<td>Sleepy</td>
</tr>
<tr>
<td>Ate breakfast and made up</td>
<td>To go to office, Hungry</td>
<td>09:00-12:00</td>
<td>Delicious, Tired</td>
</tr>
<tr>
<td>Drove a car while listening to musics</td>
<td>To go to office, To motivate</td>
<td>10:00-12:00</td>
<td>Happy, Fun</td>
</tr>
<tr>
<td>Worked</td>
<td>I’m worker, To get a salary</td>
<td>12:00-16:00</td>
<td>Delicious</td>
</tr>
<tr>
<td>Ate lunch</td>
<td>Recess</td>
<td>12:00-16:00</td>
<td>Fun, Sleepy</td>
</tr>
<tr>
<td>Listened to music</td>
<td>To relax</td>
<td>12:00-16:00</td>
<td>Sleepy</td>
</tr>
<tr>
<td>Worked</td>
<td>I’m worker, To get a salary</td>
<td>12:00-16:00</td>
<td>Sleepy</td>
</tr>
<tr>
<td>Worked overtime</td>
<td>To send mail</td>
<td>16:00-20:00</td>
<td>Tired</td>
</tr>
<tr>
<td>Drove a car</td>
<td>To go shopping</td>
<td>16:00-20:00</td>
<td>Sleepy</td>
</tr>
<tr>
<td>Went shopping</td>
<td>Shop received reservation products</td>
<td>20:00-24:00</td>
<td>Happy, Fun</td>
</tr>
<tr>
<td>Took a bath</td>
<td>To refresh oneself</td>
<td>20:00-24:00</td>
<td>Warm, Sleepy, Pleasant</td>
</tr>
<tr>
<td>Ate dinner</td>
<td>Prepared for me</td>
<td>20:00-24:00</td>
<td>Delicious</td>
</tr>
<tr>
<td>Did travel preparations</td>
<td>To go for a trip tomorrow</td>
<td>20:00-24:00</td>
<td>Tired, Pleasure</td>
</tr>
<tr>
<td>Looked for things</td>
<td>I have lost bought one</td>
<td>20:00-24:00</td>
<td>Sad, Laughing</td>
</tr>
<tr>
<td>Played a mobile phone game</td>
<td>Habit before going to bed</td>
<td>20:00-24:00</td>
<td>Happy, Sleepy</td>
</tr>
<tr>
<td>Went to bed</td>
<td>To prepare tomorrow</td>
<td>20:00-24:00</td>
<td>Sleepy</td>
</tr>
</tbody>
</table>

Table 1: Examples of collected event data
for each column. The format of this Excel file is simple; one line is for one event, one sheet is for one day, one file is for one participant.

An event includes four aspects:

1. Event name: What is happened? What did you do?
2. Event reasons: Why did it happen? What did you do?
3. Event time: Selected from the following four-hour time blocks: 0:00-4:00, 4:00-8:00, 8:00-12:00, 12:00-16:00, 16:00-20:00, or 20:00-24:00.
4. Event impressions: How did you feel?

For the aspects of reasons and impressions, participants can write more than one sentence with a space between phrases. We define two groups for collecting data. One is the long-term group which takes data with many days from a few participants, and this facilitates the comparison between participants. Another one is the short-term group which takes data for a few days from a lot of participants; this is necessary to collect various event data. Five participants wrote 20 events per day for 30 days (long-term group), and 57 participants wrote 20 events per day for seven days (short-term group); finally, we collected a total of 10,980 events. Table 1 shows examples of events collected from a participant who belongs to the short-term group. The example shows that we obtain a variety of events even if the only one participant wrote.

4 Data analysis

We analyze next two viewpoints to show that our collected data helps to answer recent personal questions that related to personality and date. First, the tendency of events was varying among participants; it shows that we have to reflect participant’s characteristics to answer recent personal questions. Second, the tendency of events was varying according to a day of the week; it shows that we have to reflect a day of the week and update event data constantly.

To analyze the tendency of events, we categorized the collected event data since they have slightly different event names, with which we cannot count the occurrence of each event. For example, we wish to handle two events such as Went to school and Went to high school as the same event. To collect such similar events as the same event, we perform the word-based hierarchical clustering using word2vec that trained from Wikipedia data.

Next, we highlight the difference between event’s tendencies among participants and days. We calculate frequency distributions of events for each participant and each day, and compare a JS divergence of these frequency distributions. This comparison clarifies two relationships of event tendencies: Distributions of event frequency depend on each participant and Participants have different distributions of event frequency depending on each day.

4.1 Event clustering

We performed hierarchical clustering to find similar events in the collected data [Larsen and Aone, 1999]. This clustering is both analysis and a necessary procedure to compare events among participants or days by collecting clusters. Before clustering, we trained word2vec [Mikolov et al., 2013] from Wikipedia articles; word2vec is useful to convert event names to a word embedding. In clustering, we tokenize event names by mecab [Kudo, 2006], and restore tokenized words to original forms. Next, we calculate vectors by adding together word2vec of each tokenized words, and cluster these calculated vectors with Ward’s method [Szekely and Rizzo, 2005]. Figure 1 shows a dendrogram and a heatmap of each vector. To confirm the difference of clustering results by the number of clusters, we respectively show the hierarchical clustering results of ten clusters and 30 clusters. Table 2 and Table 3 are ten and 30 lists of events. Event names are the nearest event to the center of each collected cluster, and cluster sizes are

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Event name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Cleaned up (掃除をした)</td>
<td>2480</td>
</tr>
<tr>
<td>E2</td>
<td>Got up (起床した)</td>
<td>1238</td>
</tr>
<tr>
<td>E3</td>
<td>Drove a car (車を運転した)</td>
<td>565</td>
</tr>
<tr>
<td>E4</td>
<td>Took a meal (ご飯を食べた)</td>
<td>1478</td>
</tr>
<tr>
<td>E5</td>
<td>Drank drink (飲み物を飲んだ)</td>
<td>1095</td>
</tr>
<tr>
<td>E6</td>
<td>Watched TV (テレビを見た)</td>
<td>633</td>
</tr>
<tr>
<td>E7</td>
<td>Took a bath (お風呂に入った)</td>
<td>436</td>
</tr>
<tr>
<td>E8</td>
<td>Went to a toilet (トイレに行った)</td>
<td>911</td>
</tr>
<tr>
<td>E9</td>
<td>Ate lunch (昼食を摂った)</td>
<td>1356</td>
</tr>
<tr>
<td>E10</td>
<td>Went to bed (寝た)</td>
<td>788</td>
</tr>
</tbody>
</table>

Table 2: List of 10 cluster’s representative events

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Event name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Looked SNS by PC (PCでSNSを閲覧した)</td>
<td>117</td>
</tr>
<tr>
<td>E2</td>
<td>Played a game (ゲームをした)</td>
<td>232</td>
</tr>
<tr>
<td>E3</td>
<td>Read mails (メールをチェックした)</td>
<td>269</td>
</tr>
<tr>
<td>E4</td>
<td>Cooked dinner (夕食の支度をした)</td>
<td>581</td>
</tr>
<tr>
<td>E5</td>
<td>Worked (仕事した)</td>
<td>260</td>
</tr>
<tr>
<td>E6</td>
<td>Did the laundry (洗濯をした)</td>
<td>1021</td>
</tr>
<tr>
<td>E7</td>
<td>Got up (起床した)</td>
<td>876</td>
</tr>
<tr>
<td>E8</td>
<td>Going to bed (就寝する)</td>
<td>146</td>
</tr>
<tr>
<td>E9</td>
<td>Worked (仕事した)</td>
<td>216</td>
</tr>
<tr>
<td>E10</td>
<td>Took the train (電車に乗った)</td>
<td>202</td>
</tr>
<tr>
<td>E11</td>
<td>Came back home by car (車で帰宅した)</td>
<td>131</td>
</tr>
<tr>
<td>E12</td>
<td>Drove a car (車を運転した)</td>
<td>232</td>
</tr>
<tr>
<td>E13</td>
<td>Ate breakfast (朝食を食べた)</td>
<td>992</td>
</tr>
<tr>
<td>E14</td>
<td>Took a meal (ご飯を食べる)</td>
<td>486</td>
</tr>
<tr>
<td>E15</td>
<td>Drank Coffee (コーヒーを飲んだ)</td>
<td>518</td>
</tr>
<tr>
<td>E16</td>
<td>Washed dishes (食器を洗った)</td>
<td>577</td>
</tr>
<tr>
<td>E17</td>
<td>Watched a video (動画を見た)</td>
<td>233</td>
</tr>
<tr>
<td>E18</td>
<td>Watched TV (テレビを見た)</td>
<td>274</td>
</tr>
<tr>
<td>E19</td>
<td>Watching TV (テレビを見る)</td>
<td>126</td>
</tr>
<tr>
<td>E20</td>
<td>Took a bath (お風呂に入った)</td>
<td>436</td>
</tr>
<tr>
<td>E21</td>
<td>Went to a toilet (トイレに行った)</td>
<td>222</td>
</tr>
<tr>
<td>E22</td>
<td>Went shopping (買い物に行った)</td>
<td>689</td>
</tr>
<tr>
<td>E23</td>
<td>Read a newspaper (新聞を読んだ)</td>
<td>218</td>
</tr>
<tr>
<td>E24</td>
<td>Tidying up dishes (食器を片づける)</td>
<td>540</td>
</tr>
<tr>
<td>E25</td>
<td>Ate lunch (昼食を摂った)</td>
<td>138</td>
</tr>
<tr>
<td>E26</td>
<td>Talked with guests (来客と話した)</td>
<td>143</td>
</tr>
<tr>
<td>E27</td>
<td>Sent a child to sleep (子供を寝かせた)</td>
<td>317</td>
</tr>
<tr>
<td>E28</td>
<td>Woke up (起きた)</td>
<td>143</td>
</tr>
<tr>
<td>E29</td>
<td>Slept in bed (ベッドで寝た)</td>
<td>300</td>
</tr>
<tr>
<td>E30</td>
<td>Slept (寝た)</td>
<td>345</td>
</tr>
</tbody>
</table>

Table 3: List of 30 cluster’s representative events
numbers of events included in each cluster.

From Table 2, we obtained common events which seem to happen to anyone such as *Got up*, *Took a meal*, *Drank drink*, *Took a bath*, *Ate lunch* and more. In contrast, from Table 3, we obtained detailed events which seem to happen to specific personas such as *Look SNS by PC*, *Played a game*, *Watched a video* and more. Such events which indicates participant’s characteristic are important to highlight the difference between participants; therefore we use the 30 lists of events as clustering result to compare events between participants and days in following analysis sections.

Note that we defined size of clusters based on a few preliminary experiments. Proposing the clustering method that determines a size of clusters based on clusters variances or entropy has a potential to improve clustering performance; therefore, we will tackle defining a better model to handle event data in future work.

From Figure 1, we can find a few particularly bright clusters that include very similar events. In contrast, some clusters with less brightness include various events that are not so similar.

Since this hierarchical clustering successfully makes clusters, we can benefit by using clustering results for data analyses. This clustering method that considers word meaning as word2vec, could make clusters which gathered almost same meaning events.

### 4.2 Event analysis for each participant

First, we analyzed events among participants. To highlight the differences between participants, we calculate the distribution on clusters of every participant. To calculate it, we used the 30 clusters in Table 3. We compare these cluster distributions between each participants using JS divergence.

The averaged JS divergence of every participant was 0.39. The minimum JS divergence is 0.063, and the maximum JS divergence is 0.77, these scores were found among participants who are the short-term group. Averaged JS divergence is not close to 0; it means that distributions of event frequency are different on each participant.

To analyze details of event tendencies, we show counts of event cluster assignment in each participant who is a long-term group, in Table 4. Most participants have different distributions of events, but E4, E6, E7, E9( and E5), E13, E14, E15, E16, E20, E22, E23, E24, E26, and E27 occurred more than once in all participants of the long-term group. E4, E6, E16, E22, and E24 are clusters containing mainly housework such as *washing*, *cleaning*, *cooking*, *shopping* and more. E7, E13, E14, E15, and E20 are clusters that indicate physiological desires such as *eating*, *drinking*, *sleeping*, *taking a bath* and more. E9 (and E5) is the cluster that indicates mainly *working*, E23 indicates *reading*, and E26 indicate *talking*. The last E27 is a cluster included various events such as *child-rearing*, *one’s hobby*, and *school life*. Such basic events that are related to living were observed in almost participants. In contrast, we obtained that events which relate to entertainment such as *Played a game* were observed in a specific participant such as P3.

These results show that participants have different event tendencies. This indicates that we should collect data which depends on each persona to answer recent personal questions.
4.3 Event statistics for each day

Second, we analyzed events among days. Like Section 4.2, we counted the clusters of every participant with 30 clusters in Table 3. We compare cluster counts of all participants in a total at a day of the week. We show JS divergences between days of the week in Table 5.

We focus on JS divergences between weekdays and weekends. These high JS divergences (We showed it as bold in Table 5) show the difference between weekdays and weekends; furthermore, the small JS divergence scores are concentrated in between weekdays and between weekends. This result shows that participants spend different life between weekdays and weekends. Such result that we can imagine easily lets us reconfirm the importance to answer depending on a day. Therefore, we need data which depend on each day to answer questions that ask about events.

5 Discussion

In this section, we discuss the potential to answer recent personal questions by our collected data. Our discussion follows the “comparison with the conversation corpus” in Sugiyama et al. [Sugiyama et al., 2014], whose PDB covers 41.3% of questions in real conversations and explains why other questions were excluded. The top reason that questions were excluded is “limited by specific words, date, or time” such as What did you eat for lunch today? or Where did you go this summer vacation? such questions are about 71.2% of a whole of excluded questions. We mainly focus on these excluded questions and show case studies which can answer by our collected data. Tackling to answer such questions helps to solve future works of the previous research.

First of all, we collect 286 questions that are the same as excluded questions by Sugiyama et al. [Sugiyama et al., 2014], and extract 204 questions that were excluded by “limited by specific words, date, or time.” In previous works, they said that these questions are difficult to maintain consistency with 5W1H answers in particular. We focus on these questions and find questions that can answer questions if we use event data. We show examples of such question which can answer based on an event in Table 6.

From Table 6, some questions which ask about speaker’s recent behaviors can answer by our collected data. For example, we can answer a question such as What did you eat for lunch today?, an answer is Yes, I ate a curry and rice by using an event such as Ate a curry and rice. In this manner, we can make an answer utterance that based on an event matched with a question. These results show us the possibility to answer a part of questions that were unsolved future works of PDB with our collected data.

We can also answer questions that ask opinions. Such questions frequently occurred after disclosure or an answer that replied to first questions. To answer with opinions, we use aspects of event impressions. We show examples of questions which ask opinions about events in Table 7. Specifically, when a conversational agent say I watched a movie, as disclosure, and the user asks Do you like it? that asks conversational agent’s opinion, a conversational agent can answer It is fun. by using an aspect of event impression from our collected data. In question-answering based on the conventional PDB, we cannot handle such kind of questions which continued the same topic as the previous turn. Answering questions about the details of the same one event, it shows the potential which improves the question-answering function to talk deeper.

However, we obtain some questions that we could not answer by our collected data; there are Questions that ask about agent’s past custom and Questions that ask about agent’s future. To answer questions that ask about agent’s future, we have to prepare the other data such as plans made by agents. These plans may need the approach such as the belief-desire-intention model that is different from our event data. To answer questions that ask about agent’s past custom, we need data which indicates habitual events and experiences. Such data seem closely related to our event data, because habitual events and experiences may be made by the accumulation of recent events. We clarify the relationship between past custom and events, and will propose a method that generates past custom based on accumulated recent events in future work.

From analyses and case studies, we showed the potential of answering for recent personal questions that cannot be answered by the previous PDB. Our collected data helps to answer not only asking events but also asking opinions. However, we obtain some problems that remain about questions which ask about past custom and future. In future work, we tackle to answer questions that ask past custom such as habitual events using our event data. Furthermore, clarifying volumes and frequency to collect enough event data; these are How many times do we ask to write per one day?, How many times do we ask to write events?, and How many days do we ask to write events?. Besides the data collection, to develop conversational agents that answer recent personal questions using event data, we have to propose a method that finds events which match with user’s recent personal questions.

6 Conclusion

In this paper, we collect 10,980 events which express recent experiences and behaviors to help conversational agents answer questions about recent experiences. First of all, we analyze collected data to highlight the tendencies of events based on each participant and each weekday, and we show the necessity of our event data that make conversational agents more natural. Our analysis shows that event data reflect participant’s characteristics and dependencies on weekdays, and we show two knowledge about tendencies of events. One, event tendencies are depending on each participant; we
should collect event data which depends on each conversational agent’s persona. Another one, event tendencies are depending on each weekday; we should collect event data which depends on each day to make conversational agent’s answers more natural.

In the discussion, we followed the previous works and obtained case studies that can answer by our collected event data. Our event data helps to answer recent personal questions such as What did you eat for lunch today? that asks about doing conversational agent’s events; therefore, results show potential to achieve our first purpose that answers a part of questions that cannot be answered by the previous PDB. Furthermore, aspects of event impressions help to answer questions that ask opinions such as Do you like it?. This continuous question-answering shows the potential which improves the question-answering function to talk deeper.

In future work, we clarify volume and frequency to collect enough event data, and develop conversational agents that answer recent personal questions by collected event data.

Table 6: Examples of question which can answer based on an event

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did you eat for lunch today?</td>
<td>Yes, I ate.</td>
<td>Ate a lunch</td>
</tr>
<tr>
<td>What did you eat for lunch today?</td>
<td>I ate a curry and rice.</td>
<td>Ate a curry and rice</td>
</tr>
<tr>
<td>Did you play video games lately?</td>
<td>Yes, I played video games.</td>
<td>Played video games</td>
</tr>
<tr>
<td>What did you play video games lately?</td>
<td>I played smartphone games.</td>
<td>Playing smartphone games</td>
</tr>
<tr>
<td>Did you watch a movie?</td>
<td>Yes, I watch a movie.</td>
<td>Watched a movie</td>
</tr>
<tr>
<td>Where did you go out somewhere recently?</td>
<td>I went to the nearby French restaurant.</td>
<td>Went to the neighboring French restaurant</td>
</tr>
</tbody>
</table>

Table 7: Examples of questions which ask impression or evaluation

<table>
<thead>
<tr>
<th>First question</th>
<th>First answer / Disclosure</th>
<th>Question</th>
<th>Opinions</th>
<th>Event</th>
<th>Event impressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did you go to the theater recently?</td>
<td>Yes, I watched “The Dark Night Rising” and “Library war”</td>
<td>How was it?</td>
<td>It is fun.</td>
<td>Watched a movie</td>
<td>Fun</td>
</tr>
</tbody>
</table>

References


Detecting Location-Indicating Phrases in User Utterances for Chat-Oriented Dialogue Systems

Hiromi Narimatsu, Hiroaki Sugiyama, Masahiro Mizukami
NTT Communication Science Laboratories
{narimatsu.hiromi, sugiyama.hiroaki, mizukami.masahiro}@lab.ntt.co.jp

Abstract
This paper establishes a method that detects words or phrases that indicate location in Japanese spoken language for a chat-oriented dialogue system. Although conventional methods for detecting words or phrases focus on named entities (NE)s, humans frequently use non-NE words to signify locations. For example, we can say “I went to that famous tower in Paris” instead of “I went to the Eiffel Tower” if we forget its proper name. Since conventional NE recognizers extract only Paris as a location from the utterance, they cannot correctly understand because the phrase “that famous tower in Paris” denotes the location in this utterance. Such insufficient understanding may allow a system to ask “Where did you go in Paris?” next, and easily result in dialogue breakdown.

To correctly understand location phrases, we focused on conditional random field (CRF)-based model as a representative method for NE extraction. Since there is no chat corpus that such location-indicating phrases are annotated, we firstly created a corpus by annotating location-indicating phrases to actual human-human chat-oriented dialogues. Then, we evaluated with the corpus how the model works.

The evaluation shows that human utterances include various location phrases except for NEs. It also shows that a CRF-based model trained a new annotated corpus detects the target phrases with high accuracy.

1 Introduction
Recently, chat-oriented dialogue systems have been attracting attention for social and entailment aspects [Bickmore and Picard, 2005; Ritter et al., 2011; Higashinaka et al., 2014; Otsuka et al., 2017]. In chatting situation, there is a significant problem that systems precisely understand users’ utterances. Although the systems need to grasp the meaning of words or phrases in utterances [Higashinaka et al., 2015], it is difficult because the domain in not limited in chats.

In this study, we focused on the understanding of location phrases. Locations are frequently used as background of a dialogue, which should be shared between talkers. In addition, location phrases are important in a slot filling-based conversational agents [Han et al., 2013]. An example system is that uses 5W1H (who, what, when, where, why, how) slots for filling by conversation. The target words or phrases are extracted from user utterances. Since the targets of when and where slots particularly appear in the beginning of dialogue, the system needs to detect whether they are included in the utterance.

For the purpose of detecting location in sentences and documents, previous work has been adopted named entity (NE) recognition. However, we human often use and understand location words or phrases except for NEs in chatting situation. We describe two cases using Figure 1 and Figure 2.

First case is that we human use and understand a common word as location. In the example shown in Figure 1, a park represents location but it is not a named entity. If the system takes 5W1H information extraction strategies, it is important to detect it as location. However, NE recognizers usually undetect it as location, and it leads a dialogue breakdown.

The second case is that humans use various words to tell a location. For instance, the following two utterances “I went to Paris” and “I went to the capital of France,” have identical meaning. However, conventional NE recognizers correctly extract Paris as the location in the first utterance, but they only extract France as the location while whole the phrase “the capital in France” is the correct location phrase in the second. Such insufficient detection also results in a dialogue breakdown.
breakdown, as shown in Figure 2.

The simplest way to detect these phrases as location is that developing a location phrase list as a dictionary and matching the target phrase against the list, but it is possible to lead misdetection such as park in “Can I park my car?” for the first case. Moreover, location phrases include not only words but also phrases like “the capitol in France,” and “the electricity shop near XX station” as shown in the second case. Therefore, simply adding these location phrases to a list is not effective.

To overcome the difficulties, we conduct this research as follows. First, we newly annotated such location-indicating phrases to human-human chat-oriented dialogues because there is no such corpus available. Then, we evaluated the location phrase detection accuracy using the chat corpus. We focused on CRF-based model that is a representative method for NE extraction, and compared three models; one is trained only NEs, another is trained the chat corpus, and the other is combined the above two models. The evaluation results show that human represents location with various phrases except for NEs, and training the chat corpus with CRF-based model is effective for detecting them.

2 Related Work

For the purpose of grasping the meaning of words or phrases, there are two types of related work. The first type is a named entity task initiated by the Defense Advanced Research Projects Agency (DARPA) [DARPA, 1995] at the Sixth Message Understanding Conference (MUC-6). It identified seven types of NEs: person, organization, location, and numeric expressions such as date, time, and money. Sekine et al. proposed an extended named entity [Sekine et al., 2002]. There are many NE recognition approaches [Sekine et al., 1998], and the scheme using conditional random fields (CRF) [Lafferty et al., 2001] has been the primary one [Nadeau and Sekine, 2007]. The characteristics using CRF is that it can estimate the sequence probability dealing with relations between n-th prior and posterior words and their features, i.e., part-of-speech (POS) tags and character types. For this task, approaches using Bi-directional LSTM, RNN have also been proposed [Chiu and Nichols, 2015; Lample et al., 2016; Wang et al., 2017]. They obtain higher performance than CRF-based methods, but they need a certain amount of training datasets to obtain stable results. Although these approaches detect NEs with high accuracy, the target location phrases are different from location phrases in chats.

The second type is an information extraction task for task-oriented dialogue systems [Lee et al., 2010; Eric et al., 2017; Bordes et al., 2017]. Basically, this is a slot-filling task, which assumes that the target words or phrases that fill the slots are predefined. For example, in a restaurant reservation task, slots are prepared for date, location, and the number of people, and they are filled through a dialogue by checking words in user’s utterances against words and phrases list that are predefined. Although this approach is effective if the words and phrases list is prepared in advance, they are unsuitable for chatting situation such that target words or phrases cannot be predefined.

3 Location Phrase Dataset

To examine what kinds of words or phrases except for NE are used as locations, we analyze human utterances in chats. Since there is no available chat data with location phrase annotations, we create a corpus by annotating location words or phrases in human-human chat-oriented dialogues.

3.1 Location phrase annotation

We use chat dialogues collected by human-human text-based chats, and annotated location words or phrases to them. The dialogue data are collected by the previous study [Meguro et al., 2009] and the dialogues are conducted without limiting the topic or contents. We use 600 dialogues and 24,888 utterances in the dataset. Each dialogue consists of about 40 utterances.

Then, we extract location-indicating phrases by manual annotations. To define the instructions for the annotation, we examined 10 chat dialogues including about 400 utterances and extracted the features of location phrases. These are example location phrases:

Example 1
I went to the capital of France yesterday.
I ate at a ramen shop near my office.

In the examples, the underlined phrases the capital of France and a ramen shop near my office, were the target locations of the utterance. Although France and ramen shop are also location words or phrases, they are partial phrases of the target locations. Therefore, we assumed a whole phrase that indicates a location is extracted as a single location.

Then, we determined the instructions as follows:

1. Annotate a sequence of words (including modifiers) as a single location, such as the capital in France instead of France.
2. Annotate words or phrases that can identify a location, such as the area around the tower and the place where I ate ramen.
3. Regard words or phrases that evoke “location” even if only slightly, as annotation target. (This definition helps to avoid overlooking any words.)
4. Clarify the ambiguity of the annotation, by attaching one of the labels shown in Table 1. (It helps to omit superfluities that may be occurred by the third instruction.)

We assumed that most of the location phrases can be intuitively understood as location, but it is possible that human cannot decide whether the phrase is location, and where the phrase is segmented. Therefore, we decided the ambiguity labels as shown in Table 1. These labels help to precisely measure the system performance by removing phrases which human cannot simply decide.

To decide the number of annotators, we firstly verified the annotation agreement using the first 30 dialogues. We employed two annotators and gave them the above instructions and the entire sequential dialogues.
Table 1: Ambiguity label.

<table>
<thead>
<tr>
<th>Label</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>The words/phrase that annotated without any hesitation.</td>
</tr>
<tr>
<td>L2</td>
<td>The words/phrase that annotated without a certain about segmentation.</td>
</tr>
<tr>
<td>L3</td>
<td>The words/phrase that the annotator annotated but had no confidence that it is a location.</td>
</tr>
<tr>
<td>L4</td>
<td>Applies both labels 2 and 3.</td>
</tr>
</tbody>
</table>

Table 2 shows the annotation agreement results. We calculated agreement score \( v \) by

\[
v = \frac{\text{Number of phrases detected by both annotators}}{\text{Number of phrases detected by the reference annotator}}.
\]

The score using all the detected phrases is shown as all and that using only label L1 is shown as L1. The agreement scores using L1 data exceeded 0.89 in both evaluations. Since the 0.89 score is high enough to use the data of a single annotator, one annotator worked on the remaining 570 dialogues in accordance with the above instructions.

Table 2: Annotation agreement.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Detector</th>
<th>L1 (all)</th>
<th>L1 (L1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotator 1</td>
<td>Annotator 2</td>
<td>0.87</td>
<td>0.89</td>
</tr>
<tr>
<td>Annotator 2</td>
<td>Annotator 1</td>
<td>0.83</td>
<td>1.0</td>
</tr>
</tbody>
</table>

3.2 Dataset analysis

We analyzed the annotated data by counting the number of ambiguity labels. The total number of location words or phrases annotated by this work was 4,202. Table 3 shows the number and the ratio of the ambiguity labels annotated to these phrases. The L1 results show that 70% of the location phrases were annotated without any ambiguity. The L2 results show that 25% were annotated with segmentation ambiguity. The other labels were much less than L1 and L2.

Table 3: Number of phrases with ambiguity labels.

<table>
<thead>
<tr>
<th>Label</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number (Ratio)</td>
<td>2914 (0.69)</td>
<td>1025 (0.24)</td>
<td>216 (0.05)</td>
<td>47 (0.01)</td>
<td>4202</td>
</tr>
</tbody>
</table>

Then we analyze the feature of sentences in each ambiguity labels by taking some representative examples. Figure 3, Figure 4, and Figure 5 are the example three sentences assigned into each label L1, L2, and L3 respectively. For label L1 that human understand the words or phrases as locations without any ambiguity, there were many location phrases except for NEs such as general nouns and the phrases including modifiers. For label L2 that human uncertainly annotated the words in regard to the segmentation place, there were words used to ambiguate the locations for example around and about. For label L3 that human annotated the words with less confidence, there were words that it is difficult to identify the unique location, and words included in other phrases that represent other entities except for location. From the results, we focused on detecting location phrases assigned L1 because it is not a big difference that understanding only Kyoto as location and area around Kyoto as location phrases. In addition, the location phrases assigned into L3 are different from others because they are some parts of other entities. Since such phrases are understood as other entities, we assumed that it is not necessary to detect them as location. Furthermore, although the location phrases assigned into L3 include phrases that cannot identify the location as fast food restaurants, human does not always understand them as location. Therefore, we use the location phrases assigned L1 as evaluation target.

4 Location Phrase Detection using Annotated Dataset

To detect target location phrases except for NE, we develop a new model using the dataset that is newly annotated in Section 3. We used CRF [Lafferty et al., 2001] to detect location phrases by training word sequences with their features and tags. Since the performance of CRF-based approach is stable and it can work with less datasets than neural network based methods, we take CRF-based approach.

We use grammatical and superficial features: the original words, the POS tags for each word estimated a priori, and five character types: hiragana, katakana, kanji, mark, and tag.
Table 4: Features and LOC-tag that is an estimation target where the input sentence is [JP]: 昨日、エッフェル塔に登ったよ。 [EN]: I went to the Eiffel Tower yesterday. The underlined words represent location.

<table>
<thead>
<tr>
<th>Word</th>
<th>Char type</th>
<th>POS</th>
<th>LOC-tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;S&gt;</td>
<td>tag</td>
<td>bos</td>
<td>O</td>
</tr>
<tr>
<td>昨日 (yesterday)</td>
<td>kanji</td>
<td>noun</td>
<td>O</td>
</tr>
<tr>
<td>, ()</td>
<td>mark</td>
<td>noun</td>
<td>O</td>
</tr>
<tr>
<td>エッフェル (Eiffel)</td>
<td>katakana</td>
<td>noun</td>
<td>B-LOC</td>
</tr>
<tr>
<td>塔 (Tower)</td>
<td>kanji</td>
<td>noun</td>
<td>I-LOC</td>
</tr>
<tr>
<td>に (to)</td>
<td>hiragana</td>
<td>pp</td>
<td>O</td>
</tr>
<tr>
<td>登った (go)</td>
<td>kanji</td>
<td>verb</td>
<td>O</td>
</tr>
<tr>
<td>た (-ed past)</td>
<td>hiragana</td>
<td>verb</td>
<td>O</td>
</tr>
<tr>
<td>に (expression)</td>
<td>hiragana</td>
<td>sep</td>
<td>O</td>
</tr>
<tr>
<td>。()</td>
<td>mark</td>
<td>sep</td>
<td>O</td>
</tr>
<tr>
<td>&lt;S&gt;</td>
<td>tag</td>
<td>eos</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 4 shows the example features where the input sentence is “[JP]: 昨日、エッフェル塔に登ったよ。 ([EN]: I went to the Eiffel Tower yesterday.) The underlined words represent location.” First, the sentence is split into words using a Japanese morphological analyzer, JTAG [Fuchi and Takagi, 1998], and POS tags were estimated simultaneously. Char type represents the character type that is determined by its unicode symbols. The LOC-tags are labeled using BIO-tags that B-LOC is attached to the first word of location phrase, I-LOC is attached to its intermediate words, and O is attached to the other words that are not location words or phrases. BIO-tags are the estimation targets. Here, the i-th word is represented as \( x_i \). To train and estimate the tag of \( i \)-th word \( x_i \), we used the features of \( x_{i-2}, \ldots, x_{i+2} \).

5 Evaluation

We evaluated the performance of the location phrase detection using the new model described in Section 4 comparing with conventional models trained only NEs.

5.1 Experimental setup

We compared the following three models:

NE CRF trains the NE location tags annotated to 1995 Mainichi newspapers.

Dial CRF trains the location tags newly annotated to our text dialogue data.

NE+Dial CRF trains both NE and Dial dataset.

For NE evaluation, we only used B-LOC, I-LOC, and O location tags instead of all NE-tags in this experiment. All the 24,888 annotated utterances were used as test data for the evaluation. For Dial evaluation, we calculated the evaluation scores by 5-fold cross-validation. For NE+Dial evaluation, we combined both of the above dataset and trained them using CRF. We evaluated the detection performance using precision, recall, and \( f\)-measure, which is the harmonic mean of the precision and recall. If the detected phrase partially matched the annotated one, it was counted as incorrect because extracting partially matched phrase such as Paris in “that famous tower in Paris” easily leads dialogue breakdown.

5.2 Results

Table 5 shows the results. Score all represents the detection performance for the annotated location phrases in all the utterances. Score L1 represents the performance using only the utterances that are annotated L1 ambiguity labels. The results of recall scores using all labels indicate that only 22% of location phrases in human-human chat dialogue are NEs, and Dial can detect non-NE location phrases by training the suitable dataset. Then, the results of precision scores show that the correctness of detected phrases using Dial are improved 0.33 points over NE. Therefore, the overall score \( f\)-Measure is improved 0.47 points. The results of label L1 remarkably indicate that human use various phrases except for NEs as location in chatting situation. Finally, combined models NE+Dial reached 0.80 for all, and 0.87 for L1 label.

To demonstrate the effectiveness of training the newly annotated data, we analyzed the detected location phrases and compared the results of the two models; NE and Dial. Figure 6 shows the example phrases of Dial successfully detected utterances and NE undetected utterances. The underlined words or phrases represent the location phrases. Although humans understand sea, mountain, and home as locations, these terms are undetected by NE because they are not location NEs. However, these words were correctly detected as locations by training the chat corpus annotated in this study.

Figure 7 shows example phrases of Dial undetected and NE successfully detected utterances. The underlining is represented as well in Figure 6. The words Florence, Palma and Bologna are named locations. Famous place names are of course included in the data of NE. However, Dial includes some famous place names only in the annotated dialogue data. Therefore, combining the training data of NE and Dial is effectively improved the detection performance. However, some named locations that are not so famous cannot be detected by both NE and Dial. Therefore, adding some named...
locations may be necessary in case that further higher accuracy is required.

From these results, Dial extracts location words and phrases that are not named entities, and a group of phrases such as the library in my neighborhood by training features of words and words’ sequence. Since the detected phrases from NE and Dial are different each other, the combined model NE+Dial is effective for detecting them. The results also show that CRF trained NE with small dialogue dataset is effective for detecting location phrase in chat-oriented dialogues.

6 Conclusion

We addressed the importance of understanding location phrases in chatting situations. To verify the performance of conventional CRF models of NE extraction for phrases that indicate locations in chatting situation, we created a new corpus of annotated location phrases in a textualized human-human chat-oriented dialogue. Our evaluation using the corpus shows that the conventional NE recognizer is insufficient for understanding location phrases in chatting situation, but the conventional method CRF is effective for detecting location-indicating phrases in chats by training the target words and phrases that are newly annotated in this studies.

In future work, we will further annotate an essential location phrase in phrases assigned to L2, L3, and L4 ambiguity labels, and evaluate the performance in detail. Then, we will implement the detection function in 5WH based chat-oriented dialogue systems, and evaluate the effectiveness. Some dialogue examples using this location-phrase detection are described in Section A. Finally, we will extend this work to other targets of 5WH except for locations.

A Appendix

We show some dialogue examples using the location phrases detection. In the case of Figure 8, the system conducts dialogue by choosing one sentence from many options. Although the similarity score between the user utterance and the option sentences is high, the system can filter the options with different locations.

In the case of Figure 9, the system correctly understands mountain as location and asks “which mountain” to identify the location in detail. Actually, the mountain near by Mt. Fuji easily makes a system misunderstand only Mt. Fuji as location. Therefore, showing the correct understanding to users may look smarter than ever.

In case of Figure 10, the system rephrases the location phrase to a correct NE. Detecting location phrase that is not NE may be used for identifying the location and rephrasing it as smart agents. These rephrasing may makes us feel the intelligence of the system.

References


System: お昼はどこで食べましたか？ (Where did you eat lunch today?)
User: 東京行きの新幹線で食べました。 (I ate lunch in the bullet train bound for Tokyo.)
System: Thinking as follows...
These options are similar to user’s utterance, but they should be filtered because the location is not the bullet train. Option1: Lunch at Tokyo is great! Option2: There are many restaurants in the Tokyo station.
System: 新幹線、ってことは駅ですか？ (Bullet train! So, you mean you ate Ekiben (box lunch)??)
User: そうそう。京都駅で駅弁買ったんだよ。 (That’s right! I bought the box lunch at Kyoto station.)

Figure 8: Example dialogue using the location phrase detection.

System: 夏休みにどこへ行きましたか。 (Where did you go last summer?)
User: 山に行きました。 (I went to a mountain.)
System: Thinking as follows... Understand a mountain as location, but that is not named entity.
System: 山行ったんですよね。どちらの山に行かれたんですか？ (Oh, sounds good! Which mountain did you go?)
User: 富士山の近くの山に登りました。 (I climbed the mountain near by Mt. Fuji.)
System: Thinking as ... Understand the mountain near by Mt. Fuji as location.
System: 富士山じゃないんですね。 (Oh, that's not Mt. Fuji.)

Figure 9: Example dialogue using the location phrase detection.

User: 先週末、パリにあるタワーよりに登りました。 (I went to the tower in Paris last weekend.)
System: Thinking as follows... Understand the tower in Paris as location and identify it as Eiffel Tower.
System: あって、エッフェル塔ですね。どうでしたか。 (Oh, that’s the Eiffel Tower, isn’t it? How was that?)
User: そうそう。すごく眺めが良かった。 (Yes! The view was great!)

Figure 10: Example dialogue using the location phrase detection.


Towards a structured evaluation of improv-bots: Improvisational theatre as a non-goal-driven dialogue system

Maria Skeppstedt¹,², Magnus Ahltorp³

¹ Computer Science Department, Linnaeus University, Växjö, Sweden
² Applied Computational Linguistics, University of Potsdam, Potsdam, Germany
³ Magnus Ahltorp Datakonsult, Stockholm, Sweden
maria.skeppstedt@lnu.se, magnus@ahltorpdata.se

Abstract
We have here suggested a structured procedure for evaluating artificially produced improvisational theatre dialogue. We have, in addition, provided some examples of dialogues generated within the evaluation framework suggested. Although the end goal of a bot that produces improvisational theatre should be to perform against human actors, we consider the task of having two improv-bots perform against each other as a setting for which it is easier to carry out a reliable evaluation. To better approximate the end goal of having two independent entities that act against each other, we suggest that these two bots should not be allowed to be trained on the same training data. In addition, we suggest the use of the two initial dialogue lines from human-written dialogues as input for the artificially generated scenes, as well as to use the same human-written dialogues in the evaluation procedure for the artificially generated theatre dialogues.

1 Introduction
Improvisational theatre (or impro/improv) is an art form in which unscripted theatre is performed. Dialogue, characters and actions are typically created spontaneously. Through collaboratively creating a story, the actors can make a new scene evolve in front of the audience. [Wikipedia contributors, 2018].

Seen from the perspective of artificial intelligence research, improvisational theatre is a sub-genre of human interaction that is more forgiving than interaction in general. Errors made in general interaction are typically seen as a failure, and in the case of a dialogue system, errors might lead to a dialogue breakdown. In contrast, errors made within an improvisational theatre scene are encouraged, and can form an input to how the scene evolves. It might, therefore, be interesting to find out how artificially constructed improvisational theatre bots, which are likely to make errors to a larger extent than a human, are perceived in this special setting.

Although there is previous work on the construction of artificially generated improvisational theatre, there are, to the best of our knowledge, no descriptions of structured methods for the evaluation of the dialogues created, and thereby no method for comparing different approaches for dialogue generation. According to Serban et al. [2016], even the more general question of which evaluation method to use for non-goal-driven dialogue systems (for which improvisational theatre could be claimed to be a sub-category), is an open one.

The aim of this paper is therefore to i) provide a suggestion for a structured procedure for evaluating artificially produced improvisational theatre dialogue, and ii) give some examples of dialogues generated within the evaluation framework suggested.

2 Previous work
Creating artificially generated human dialogue is a classical task within the research field of artificial intelligence, with the ultimate aim of a bot being able to pass the Turing test. Dialogue could either be created in the form of a goal-driven dialogue system, i.e., a system that is meant to be used to perform a specific task, such as booking a ticket, or in the form a non-goal-driven system, for which no such task is given.

2.1 Conversational modelling and dialogue systems
One implementation method for the task of generating dialogue is to use actual lines (possibly slightly modified) from an existing dialogue corpus. This approach was, for instance, applied by Banchs and Li [2012]. They constructed a vector space model-vector from the previous lines in the dialogue, i.e., lines either automatically generated or provided by the human dialogue participant, and measured its distance to vectors constructed in the same fashion from the dialogue corpus. The closest corpus dialogue which had the closest vector representation was then retrieved, and the dialogue line from the corpus, which was given in response to the ones retrieved, was returned as the next line in the dialogue.

Another solution is to generate new sentences, that do not necessarily have to have been present in the corpus used for training. For this task, neural network techniques are typically applied [Vinyals and Le, 2015; Li et al., 2016; Serban et al., 2016]. For instance, the seq2seq architecture (perhaps best known for its ability to carry out machine translation [Sutskever et al., 2014; Luong et al., 2017]), has been applied for conversational modelling/dialogue generation.

The second approach is intuitively more appealing, since it gives more flexibility to what kinds of lines that can be gen-
erated. Previous studies have, however, shown examples of the generative approach resulting in utterances that are fairly general, as well as examples of that the same utterances are often repeated. That is, the content that is most commonly occurring in the training corpus is that which is most typically being generated, and the potential for flexibility does not automatically lead to a larger creativity. Instead, dialogue lines that are generated mainly on the basis of what is very representative to the corpus might thus be boring in the context of improvisational theatre (and possibly also in most other applications of non-goal-driven dialogue systems). It has been possible to solve the problem of repeated lines, through the application of reinforcement learning that rewards diversity, but the examples provided in the paper describing this approach still include dialogue lines that are rather generic [Li et al., 2016].

In addition, we suspect that the generative approach might require larger dialogue corpora to give usable results, despite that out-of-domain resources, such as large external monologue corpora to initialise word embeddings, have been shown useful [Serban et al., 2016]. Li et al., for instance, used the OpenSubtitles parallel corpus, which consists of around 80 million source-target pairs, for their generative approach.

Since it is relevant to be able to provide automatically generated improvisational dialogues also for languages for which there does not exist a large dialogue corpus and possibly not even a large out-of-domain corpus, or for sub-genres within a language (e.g., improvised Shakespeare [The Improvised Shakespeare Company, 2018] or Strindberg [Strindbergs intima teater, 2012]), it is also important to explore the performance of methods that are less resource demanding. Therefore, along with exploring generative approaches, it might also be relevant to compare these (for different in-domain or out-of-domain training data sizes) to methods that create dialogues through the use of existing dialogue lines.

### 2.2 Artificial improvisational theatre

The use of artificial intelligence as a part of improvisational theatre has recently been explored by Mathewson and Mirowski [2017]. Their work included the creation of a dialogue system that allowed a human improvisation actor to communicate with a robot that produced lines in response to lines uttered by the human actor. Two versions of the robot dialogue were constructed, one version that selected existing lines in the training corpus, and one version that relied on text generation techniques.

The ambitious approach by Mathewson and Mirowski thus included the use of speech recognition and a text-to-speech system, which functioned in real-time in front of an audience. We believe that this set-up is an appropriate goal for artificial intelligence-powered improvisational theatre, in particular their choice of including a human actor as one of the participants in the dialogue. We suspect, albeit without being able to provide any substantial basis for this suspicion, that watching a human produce lines in real time is one of the main fascinations of improvised theatre, and that many audience members would quickly lose interest in a play if they were aware of that it only included artificial actors and artificially generated dialogue.

We do, however, not consider this ambitious approach to be appropriate for the goal of objectively evaluating, and thereby in the long run improving, the generation of improvised dialogue. The main reason for this is that the competence of the human actor impacts the quality of the resulting dialogue, since skilful improvisers have a larger ability to fit strange utterances from a co-actor into an improvised scene. There is, for instance, an improvisational theatre game [improviki, 2018b], where the actors are given a set of pre-written, out-of-context lines, which they are to incorporate in a natural way into the scene. A human actor in an improvisational theatre dialogue is thus very different from a human interacting with a standard, task-oriented dialogue system. In addition, the quality of the text-to-speech system and the speech recognition might influence the audience’s perception of the dialogue, and thereby their evaluation of the quality of the dialogue content.

### 3 Evaluation procedure suggested

Given the problems of including a human actor in a more structured evaluation, we suggest the following procedure for evaluating automatically generated improvisational theatre, in which the task is narrowed down to the generation of dialogue and in which the dialogue is initialised in a manner which increases the possibilities to carry out a reliable evaluation.

#### 3.1 Interaction between two bots

A more reliable evaluation method needs to remove the human influence, and the easiest approach for achieving that would be to replace the human actor with another improvisation bot, i.e., the set-up would be two improvisation bots talking to each other. However, since the end goal is to construct a bot that is able to act against a human actor, the functionality of the bots should not be allowed to be dependent on any one of the bots having full knowledge of the other bot. Instead, the shared knowledge between the two bots should aim to approximate the shared knowledge between two human improvisational actors.

To approximate that level of shared knowledge, we suggest that the two bots that are to be evaluated should not be allowed to be trained on the same training data. The data could be taken from the same text genre, but is should not be the exact same data. That is, in the same manner as two humans that learn the same language are exposed to text from the same genre, i.e., the very wide genre of utterances from many different registers in the language, but are not exposed to the exact same utterances.

#### 3.2 Starting the improvised scene

Improvisational theatre is often carried out with the use of a set of constraints, typically in the form of an input that the actors can use as a starting point for their scene. For instance, the audience could provide an input in the form of a suggestion for a location at which the scene is to take place. Another example is input in form of body postures that the actors use as the starting point for a scene [Johnstone, 1999, pp. 186–187].
The evaluation method we suggest is to use the two initial dialogue lines from human-written dialogues as input for the scene. This is a form of input that can be easily automated on a larger scale (as opposed to using non-textual input such as body postures). In addition, the two initial lines provide background data that the dialogue bots can use for generating new lines, which simplifies the task somewhat.

Most importantly, however, using the first two lines of human-written dialogue as input, will result in that the artificially generated dialogue has a comparable human-written dialogue against which it can be evaluated. To make them as comparable as possible, the improvisation bots could be instructed to generate approximately the same number of dialogue lines as the number of lines included in the human dialogue.

3.3 Evaluating the scene from the perspective of its likelihood of having been produced by humans

With the use of this set-up for dialogue generation, for which there will be comparable human-written and automatically generated texts available, the evaluation can be carried out as follows:

The two initial dialogue lines are randomly sampled from a set of (preferably short) human-written dialogues, and one or several pairs of bot systems use these two initial lines to produce a generated dialogue.

A human evaluator is then presented with a set of short dialogue texts, of which some (e.g., half of them) have been selected from the human-written dialogues from which the two initial starter-lines were sampled, and some from the automatically generated dialogues. The task of the human evaluator is then to, for each text, decide whether the dialogue has been generated by a machine or produced by humans. The same human evaluator should not be presented with a human-written dialogue and an automatically generated one that begins with the same two initial lines. With this restriction, the situation that the evaluator carries out a direct comparison between the two texts is avoided. An evaluation through comparison would be a less realistic task, since the final aim is to produce a dialogue that could pass as human-produced, not a dialogue that is more human-like than a text that has actually been produced by a human. Employing at least three human evaluators, would be a prerequisite for all automatically generated texts being shown to a human, and that enough texts are annotated to allow for inter-annotator agreement calculations.

Naturally, the set of dialogues from which the two initial dialogue lines are sampled to use as evaluation data, can not be allowed to be included in the data sets used for training the improv-bots.

3.4 Evaluating the scene from other perspectives

There are, of course, other aspects than the resemblance to a human-produced dialogue that the dialogues generated should be evaluated for. Two parameters, mentioned in the background, are the level of diversity among the lines generated and how general the lines produced are. Repetitive and generic dialogue lines are both examples of phenomena that might produce a boring scene, and these two parameters might therefore be combined into a metric in the form of the entertainment value of the dialogues. The evaluator should, therefore, when estimating whether a dialogue has been produced by a human, also assess how entertaining the dialogue is. This is likely to be a more subjective measure. However, given a hypothetical situation in which the artificially generated dialogue often is perceived as being generated by a human, but these dialogues are consistently being given a lower entertainment value score than the human-written dialogues, then this would give an indication of that there is something important missing in the dialogues generated. The easiest solution is, probably, to use a binary score, e.g., to let the evaluator determine whether the dialogue was boring or not.

There are also other types of measures that could be applied for evaluating generated dialogues, e.g., measures that are related to techniques taught within improvisational theatre. An actor should, for instance, aim to be collaborative, e.g., give offers to and accept offers from the co-actors [Johnstone, 1987, pp. 94–108]. To help the audience follow a scene, what roles the actors play, what their relationship is, where the scene is played and what the objectives of the characters are, should also be established early on in a scene [improviki, 2018a]. It would be a very interesting task to construct an improvisational theatre bot that could achieve such improv-theatre tasks. With these more specific tasks, however, the system is perhaps no longer a non-goal-driven dialogue system, but starts to resemble a goal-driven system. Creating such a system is thus a separate task, for which a separate framework for evaluation should be developed.

4 Implementation

In the long run, we aim to implement and evaluate a resource-intensive method as well, e.g., a method that uses seq2seq to generate new text. However, to illustrate the evaluation method, we here implemented a dialogue creation strategy built on selecting the most appropriate line from a dialogue corpus. This method uses i) a moderate-size dialogue corpus, and ii) a distributional semantics space that is constructed from a very large out-of-domain corpus. We apply a dialogue generation method that is built on several different sub-ideas, which we hope might serve as inspiration for future work, but an evaluation of the contribution of each idea is not within the scope of this paper.

As corpus, we used the Cornell movie-dialogues corpus [Danescu-Niculescu-Mizil and Lee, 2011], and as distributional semantics space we use the word2vec space that has been pre-trained on a very large corpus of Google News and which has been made available by Mikolov et al. [2013; 2013].

Due to the spontaneous and collaborative nature of improvisational theatre, we believe that each dialogue line in this genre in average is likely to be shorter than lines in scripted theatre. We, therefore, extracted a subset of dialogue line triplets from the Cornell movie-dialogues corpus, where each of the lines had to conform to the following set of length criteria: A line was allowed to contain a maximum of two sentences, and in case it contained two sentences, the first of
these two sentences was allowed to contain a maximum of two tokens. The last sentence (that is, the only sentence for one-sentence lines and the second for two-sentence lines) was allowed to contain a maximum of twelve tokens. Sentence splitting and tokenisation was carried out with NLTK [Bird, 2002].

In the Cornell movie-dialogues corpus, there were only 262 dialogues that contained at least six dialogue lines and for which all of the lines conformed to the length criteria we had established for the experiment. These 262 dialogues were, therefore, saved to use as the set of evaluation data, i.e., data which could be used in the evaluation of the automatically generated dialogues. Line triplets from the rest of the corpus were divided into two groups, one group to use as training data for Actor A and another group to use as training data for Actor B. We divided the triplets film-wise, so that all triplets from the same film were assigned either as training data to Actor A or to Actor B. In addition, 100 of the dialogues were not added to the training data set, but were used for an informal evaluation during the development, i.e., used as the two first input lines to run the dialogue generation during development. A total of 10,322 line triplets were used to train the functionality for Actor A and a total of 10,884 line triplets for the functionality of Actor B.

A context in the form of the line most recently uttered in the dialogue and the line before that was used as input data for predicting the next line in the dialogue. The first two lines of each training data triplet were used to represent these two most recent lines, and the third line to represent the line to be predicted. The core of the method for prediction was thus to retrieve the training data triplet for which the two first lines were most similar to the two most recent lines in the generated dialogue, and to use the third line in the triplet as the next line in the generated dialogue. Similarity of dialogue line pairs was determined through converting the two lines into a semantic vector representation, and using the Euclidean distance between the vectors as the similarity measure.

The vector representation for the previous, and the most recently uttered line in the generated dialogue (as well as for the first and second lines in the training data triplets), were constructed as follows: For the previous line, the average of the word2vec vectors representing the tokens in the line were used as the line representation. Tokens present in a standard English stop word list were removed before creating the average vector. For the most recently uttered line, the same representation was used, except that stop words were retained. We believe that also words that are normally considered as stop words are important when interpreting the exact content of the most recently uttered dialogue line, while they might be less important for the content of an earlier line which we included to provide a topical context.

In addition to the averaged vectors, we used the word2vec representation of the three first tokens in the most recently uttered line, as well as the three last tokens in the line, as we believe that these might be more important than the other words for capturing the surface form of the conversation. All of these six vector representations were then concatenated into one long vector. The averaged vectors were slightly down-weighted, to give more importance to the vector representa-

tions for the three initial and ending tokens of the most recent line (the weights were determined by inspecting the output of the algorithm on the development data). Vector elements were also added to indicate whether a line contained any of the question words who, where, when, why, what, which, how or a question mark.

When there were several dialogue line pairs in the training data that matched the lines in the generated dialogues equally well (allowing for a maximum Euclidean distance difference of 0.08 between different candidates), and which resulted in many candidates for the next line, we applied an unsupervised outlier detection to this set of candidates, using scikit-learn’s OneClassSVM [Pedregosa et al., 2011]. The set of outliers were then removed from the candidate list.

For the number of candidates that were still present in the candidate list after outliers had been removed, we tried to incorporate the co-operative spirit of improvisational theatre for selecting which of them to use. This was accomplished by selecting the candidate line, for which, when this line (together with its preceding line) was submitted as input to the algorithm, the closest neighbour was found. The motivation for this was that when a line was selected to which the co-actor would be more likely to find a good answer, the dialogue would run more smoothly, i.e., just as in real improvisational theatre.

We also applied two simple rules to improve the dialogues, i) to avoid to end a dialogue with a line ending with a question mark, ii) and to avoid repeating a line in the dialogue. These rules were, however, not strictly enforced, and when there were no other candidates of approximately the same quality as a line ending with a question mark or as a repeated line, these were still used.

Word2vec vectors were accessed through the Gensim library [Rehůřek and Sojka, 2010]. The search for dialogue line pairs in the training data, i.e., the dialogue line pairs that were closest to the data given when constructing new dialogues, was sped up by training a scikit-learn OneClassSVM [Pedregosa et al., 2011].

5 Example output

In Table 1, we present 6 generated dialogues, which were randomly sampled from the set of 262 dialogues that had been set aside as evaluation data. The first two lines are given from the corpus dialogue, and the left-hand column presents the generated version while the right-hand column presents the human-written corpus version. The last two examples show the output of our algorithm and the output presented by Li et al. [2016]. Similarly as when generating lines starting from human-written dialogue, we provided the first two lines in the dialogues published by Li et al. as input to our system.

Our suggested formal evaluation of these dialogues would thus be to present half of the dialogues in Table 1 to Evaluator 1 and the other half to Evaluator 2, who are to determine i) whether the dialogue is produced by a human or not, and ii) whether the dialogue is boring. When informally evaluating these dialogues, we would say that most dialogues in the right-hand column would pass as human made, except the strange dialogue 2, while hardly any of the dialogues in the left-hand column would be classified as produced by humans.
Table 1: The automatically generated dialogues compared to the human-written dialogues, and (for the two last examples), compared to the output of previously published generated dialogue examples. The same human evaluator would either be shown the text in the left-hand column or the text in the right-hand column, and determine i) whether it has been produced by a human, and ii) whether it is boring.

<table>
<thead>
<tr>
<th>Computer-generated</th>
<th>Human-written</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>A: I was first to respond.</td>
<td>A: I was first to respond.</td>
</tr>
<tr>
<td>B: What were you doing out here?</td>
<td>B: What were you doing out here?</td>
</tr>
<tr>
<td>A: My debrief –</td>
<td>A: Drive by patrol.</td>
</tr>
<tr>
<td>B: Change?</td>
<td>B: How is she?</td>
</tr>
<tr>
<td>A: With a vengeance, I hated them.</td>
<td>A: She’s tough.</td>
</tr>
<tr>
<td>B: She was the Queen of the Netherlands.</td>
<td>B: Have to be. The shit she’s gone through.</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A: I NEED SOME</td>
<td>A: I NEED SOME</td>
</tr>
<tr>
<td>B: YOUNG BLOOD</td>
<td>B: YOUNG BLOOD</td>
</tr>
<tr>
<td>A: On your sleeve.</td>
<td>A: I NEED SOME</td>
</tr>
<tr>
<td>B: Well, I’m sure she kept her word and left you satisfied.</td>
<td>B: YOUNG BLOOD</td>
</tr>
<tr>
<td>A: No. No you're not.</td>
<td>A: I NEED IT NOW I NEED SOME</td>
</tr>
<tr>
<td>B: Yes, Mr. Hillyer, I am.</td>
<td>B: YOUNG BLOOD</td>
</tr>
<tr>
<td>A: Guess we better get going, huh?</td>
<td>A: I NEED SOME</td>
</tr>
<tr>
<td>B: Just a little somethin’ somethin’.</td>
<td>B: YOUNG BLOOD</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>A: I’d better go in.</td>
<td>A: I’d better go in.</td>
</tr>
<tr>
<td>B: I only live up there and across the roof.</td>
<td>B: I only live up there and across the roof.</td>
</tr>
<tr>
<td>A: Did you find something?</td>
<td>A: Thanks anyway.</td>
</tr>
<tr>
<td>B: Who?</td>
<td>B: Listen, Edie, am I going to see you again?</td>
</tr>
<tr>
<td>A: If he’s not out, why are they here?</td>
<td>A: What for?</td>
</tr>
<tr>
<td>B: I have no idea.</td>
<td>B: I don’t know.</td>
</tr>
<tr>
<td>A: Why not?</td>
<td>A: I really don’t know.</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A: This is ridiculous.</td>
<td>A: This is ridiculous.</td>
</tr>
<tr>
<td>A: Office work?</td>
<td>A: Why can’t we do it through the mail?</td>
</tr>
<tr>
<td>B: You bet it did.</td>
<td>B: We missed the deadline.</td>
</tr>
<tr>
<td>A: You drive away business roarin’ in like that.</td>
<td>A: Can’t we do it on the phone?</td>
</tr>
<tr>
<td>B: Bravo!</td>
<td>B: I don’t think so.</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>B: That’s a fine-looking ring.</td>
<td>B: That’s a fine-looking ring.</td>
</tr>
<tr>
<td>A: Busy tomorrow?</td>
<td>A: A lady threw it to me.</td>
</tr>
<tr>
<td>B: Yes. Things are pretty hectic right now, but thanks.</td>
<td>B: Most surely that was a nice lady to give a ring to a little girl.</td>
</tr>
<tr>
<td>A: I know!</td>
<td>A: It’s a pretty ring.</td>
</tr>
<tr>
<td>B: I’m gonna have to ask you to leave.</td>
<td>B: I wouldn’t be surprised if it were a true wishing ring.</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>A: You all right?</td>
<td>A: You all right?</td>
</tr>
<tr>
<td>B: Yes, thank you.</td>
<td>B: Yes, thank you.</td>
</tr>
<tr>
<td>A: Hello, Helen.</td>
<td>A: What was that all about?</td>
</tr>
<tr>
<td>B: I’m afraid I spoke out of turn last night.</td>
<td>B: He wants me to perform for him.</td>
</tr>
<tr>
<td>A: From whom?</td>
<td>A: Perform?</td>
</tr>
<tr>
<td>B: The main guy, Dieter--</td>
<td>B: You know.</td>
</tr>
<tr>
<td>A: I don’t have time.</td>
<td>A: You on the game?</td>
</tr>
<tr>
<td>B: I can see why.</td>
<td>B: God no. I’m a hairdresser.</td>
</tr>
</tbody>
</table>

Computer-generated here

Computer-generated in previous studies

| 7                   |               |
| A: How old are you? | A: How old are you? |
| B: I’m 16. Why are you asking? | B: I’m 16. Why are you asking? |
| A: Why are you smiling? | A: I thought you were 12. |
| B: To tell you that we’re interested in the property. | B: What made you think so? |
| A: Oh, Dmitri.I wish that I could. | A: I don’t know what you are talking about. |
| B: He didn’t say we need guns. | B: You don’t know what you are saying. |
| 8                   |               |
| A: Where are you going? | A: Where are you going? |
| B: I’m going to the police station. | B: I’m going to the police station. |
| B: I’ll scream. | B: No, no, no, no, you’re not going anywhere. |
| A: I mean, we get paid tonight, don’t we? | A: Why? |
| B: What the fuck is that supposed – | B: I need you to stay here. |
| A: What? | A: I don’t know what you are talking about. |
None of the dialogues would, however, be classified as boring, except maybe the first of the two dialogues provided by Li et al. [2016], as it starts to generate very generic lines towards the end of the dialogue.

6 Conclusion and outlook

The generated dialogues presented here portray a collection of somewhat strange exchanges, and would not be useful in the context of simulating a real conversation. They might, however, function as absurd dialogues that, for instance, could be used as improvised scene starters. We believe, however, that the more structured form of evaluating a non-goal-driven dialogue system that we present and exemplify could be generally useful. The evaluation structure might be possible to apply in the setting of a shared task, in which the participants not only produce dialogues of this type, but also participate in the evaluation by classifying the dialogues produced by other participating groups.

The next step is to implement a more resource-intensive method, e.g., a method built on seq2seq or some other neural network-based technique. We also intend to extend our initial attempts of achieving dialogue generation with the help of a moderately sized dialogue corpus. We have, for instance, not yet attempted any post-processing of the selected lines to make them fit better into the dialogue, e.g., to make the pronoun gender and number agree between the lines, or to match the use of helper verbs.

Although the ultimate goal would be to achieve an improvisation bot that could act seamlessly with a human actor, it would also be interesting to explore the suspicion we introduced in the background, i.e., that an audience would quickly lose interest in a play if they were aware of that it consisted solely of artificially generated dialogue. For instance, if two puppets were given two starting lines by the audience, and from these starting lines played a scene with automatically generated human-like dialogues, would the audience still find it interesting?

Acknowledgements

We would like to thank Jonas Sjöbergh, as well as the anonymous reviewers, for valuable input to the content of this paper.

References


2012.


Refinement of utterance database and concatenation of utterances for enhancing system utterances in chat-oriented dialogue systems

Yuiko Tsunomori¹, Ryuichiro Higashinaka², Takeshi Yoshimura¹
¹NTT DOCOMO
²NTT Corporation
¹{yuiko.tsunomori.fc, yoshimurat}@nttdocomo.com,
²higashinaka.ryuichiro@lab.ntt.co.jp

Abstract
We have been using an utterance database created from a massive amount of predicate-argument structures extracted from the web for generating utterances of our commercial chat-orientated dialogue system. However, since the creation of this database involves several automated processes, the database often includes non-sentences (ungrammatical or uninterpretable sentences) and utterances with inappropriate topic information (called off-focus utterances). Also, utterances tend to be monotonous and uninformative because they are created from single predicate-argument structures. To tackle these problems, we propose methods for filtering non-sentences by using neural-network-based methods and utterances inappropriate for their associated foci by using co-occurrence statistics. To reduce monotony, we also propose a method for concatenating automatically generated utterances so that the utterances can be longer and richer in content. Experimental results indicate that our non-sentence filter can successfully remove non-sentences with an accuracy of 95% and that we can filter utterances inappropriate for their foci with high recall. We also examined the effectiveness of our filtering and concatenation methods through an experiment involving human participants. The experimental results show that our methods significantly outperformed the baseline in terms of understandability and that the concatenation of two utterances leads to higher familiarity and content richness while retaining understandability.

1 Introduction
Chat-oriented dialogue systems have become increasingly popular [1; 2; 3; 4; 5]. Such systems need to generate a wide variety of utterances to cope with the many topics contained in user utterances. Although rule-based methods have typically been used to generate system utterances, the topics that appear in chats are diverse, and it is extremely expensive to create rules with adequate coverage [6].

To overcome this weakness, Higashinaka et al. [7] proposed a method of using a large volume of text data on the web to extract predicate-argument structures (PASs) and convert them into utterances. The result of this method is a database of utterances with their associated topics (called foci) (see Section 3 for details). We are using the utterance database created by this method in our commercial chat-oriented dialogue system¹ [1].

Although this method can generate utterances corresponding to a variety of foci by exploiting the richness of the web, system utterances have the following problems:

- Because of errors resulting from automatic analysis of PASs and their automatic conversion into utterances, non-sentences (ungrammatical or uninterpretable sentences) and utterances inappropriate for their associated foci (called off-focus utterances) can sometimes be generated.
- The system utterances tend to be monotonous and uninformative because they are created from single PASs.

In this paper, we propose methods for improving the quality of the utterance database created by using Higashinaka et al.’s method [7] and for reducing the monotony of system utterances. In particular, our methods filter non-sentences and off-focus utterances using neural-network-based methods and co-occurrence statistics. We also propose a method of reducing monotony by concatenating pairs of automatically generated utterances about the same focus so that the utterances can be longer and richer in content. We verified the effectiveness of our methods through an experiment involving human participants. Our contributions are as follows:

- We successfully created non-sentence and off-focus filters that can greatly refine the utterance database created from PASs on the web. In terms of theutterance quality, we observed significant improvements regarding familiarity, understandability, and content richness in subjective evaluations. By using our methods, the utterances of the database can be safely used by chat-oriented dialogue systems.
- We found that, by concatenating two utterances about the same focus from the utterance database, we can create utterances that are significantly better in terms of familiarity and content richness. We confirmed that this effect is brought about only when we use the utterance

¹https://dev.smt.docomo.ne.jp/
database refined by the non-sentence and off-focus filters. We believe our proposed methods can especially contribute to commercial chat-oriented dialogue systems, in which the quality of utterances is critical.

The paper is structured as follows. In Section 2, we cover related work. In Section 3, we explain our PAS-based utterance database and examine the proportions of non-sentences and utterances inappropriate for their associated foci. In Section 4, we explain our proposed methods for filtering inappropriate utterances and our utterance-concatenation method. In Section 5, we explain our experiment involving human participants. Finally, we summarize the paper and discuss future work in Section 6.

2 Related work

Various methods have been proposed to generate utterances in chat-oriented dialogue systems, such as rule-, retrieval-, and generation-based methods.

Rule-based methods generate system utterances on the basis of hand-crafted rules. Representative systems that use such rules are ELIZA [8] and A.L.I.C.E. [9]. However, the topics that appear in chat are diverse, and it is extremely expensive to hand-craft rules with wide coverage [6].

Retrieval-based methods have been proposed to improve coverage. The recent increase in web data has propelled the development of methods that use data retrieved from the web for open-domain conversation [10; 11; 2]. The advantage of such retrieval-based methods is that, owing to the diversity of the web, systems can retrieve at least some responses for user input, which can solve the coverage problem. However, this comes at the cost of utterance quality. Since the web is inherently noisy, it is, in many cases, difficult to sift out appropriate sentences from retrieval results.

Recently, generation-based methods based on neural networks have been extensively researched. However, these methods generally tend to generate utterances with little content, although there has been research on improving the diversity in generated utterances [12; 13]. We acknowledge that current neural-network-based methods are yielding promising results. However, we use an utterance database created from PASs on the web [7] because it is guaranteed to output system utterances with content related to the focus of the conversation and because system utterances can be more controllable, which is particularly important for commercial applications.

The detection of inappropriate utterances including non-sentences is related to the detection of grammatical errors made by second-language learners. Imaeda et al. [14] proposed a dictionary-based method for detecting case particle errors by using a lexicon. Oyama et al. [15] proposed a support vector machine (SVM)-based method for detecting case particle errors in documents created by non-native Japanese speakers, and Imamura et al. [16] proposed a method for detecting all types of particle errors. However, these methods cannot be directly applied to the utterances of dialogue systems since the error tendency of automatically generated utterances differs from that of second-language learners. The detection of inappropriate utterances has also been tackled in dialogue breakdown detection challenges (DBDCs) [17; 18]. However, the main focus is on detecting inappropriate utterances in the context of dialogue, whereas we focus on refining an utterance database. Inaba et al. [19] proposed a monologue-generation method for non-task-oriented dialogue systems by concatenating sentences extracted from Twitter. This is similar to our concatenation method in that it concatenates utterances to reduce monotony but different in that it targets monologues rather than dialogues.

3 PAS-based utterance database

We first describe the construction and details of the utterance database of our chat-oriented dialogue system. Then, to illustrate the problems with the database, we examine the proportions of non-sentences and off-focus utterances.

3.1 Creation of the utterance database

We use the utterance database created by using the method described by Higashinaka et al. [7]. The method uses PAS analysis [16] to extract PASs with their foci from a large amount of text data. To extract high-quality PASs and their foci, the method extracts predicates with just two arguments explicitly marked with particles ‘wa’ and ‘ga.’ ‘Wa’ is a topic marker and ‘ga’ is a nominative case marker in Japanese. This way, a subject and a predicate can be extracted as constituents of a PAS together with a focus.

Since PASs cannot be uttered as they are, they need to be converted into utterances. Given a PAS and a dialogue-act type (we need this as input because utterances require underlying intentions; dialogue-act types are described below), an utterance is automatically created. The PASs are first converted into declarative sentences using a simple rule. Then, their sentence-end expressions (NB. In Japanese, modalities are mostly expressed by sentence-end expressions) are swapped with those matching the target dialogue-act type. The sentence-end expressions used are those automatically mined from dialogue-act-annotated dialogue data. The details of the method of obtaining and swapping sentence-end expressions are given by Miyazaki et al. [20].

From the list of 32 dialogue-act types [21] 21, which are mainly related to self-disclosure and question, are used for conversion. From blog data (about three years’ worth of blog articles) and by the combination of the extracted PASs and the dialogue-act types, the resulting utterance database contains 7,116,597 utterances associated with 204,497 foci.

3.2 Quality of utterance database

Since the PASs are extracted and converted into utterances automatically, errors in the resulting utterances are inevitable, affecting the quality of the utterance database. From our observation, there can be two types of erroneous utterances: non-sentence and off-focus utterances.

Non-sentence Sentences that we cannot understand due to grammatical errors or a strange combination of words. Non-sentences are generated mainly in the conversion of sentence-end expressions; some propositions cannot be uttered with certain sentence-end expressions in Japanese (see [22] for such examples).
Table 1: Examples of non-sentence annotation (0: non-sentence, 1: valid-sentence). A1 and A2 indicate the labels given by the two different annotators. Utterances were originally in Japanese. English translations in parentheses were done by authors.

<table>
<thead>
<tr>
<th>Focus</th>
<th>Utterance</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>秋冬 (Fall &amp; winter)</td>
<td>どんな人が流行りますよね (What kind of types is popular, isn’t it?) (NB. This sentence sounds odd because its subject is an interrogative while the sentence is declarative.)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>秋冬 (Fall &amp; winter)</td>
<td>レギンスがつけますよね (Boys wearing leggings are increasing, aren’t they?)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>秋冬 (Fall &amp; winter)</td>
<td>空気が乾燥してるだろ ((Air is dry and so on.)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Statistics of non-sentence annotation (0: non-sentence, 1: valid-sentence)

<table>
<thead>
<tr>
<th></th>
<th># of utterances</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 annotators labeled 0</td>
<td>23,052</td>
<td>12%</td>
</tr>
<tr>
<td>2 annotators labeled 1</td>
<td>150,955</td>
<td>75%</td>
</tr>
<tr>
<td>1 annotator labeled 0, other labeled 1</td>
<td>25,993</td>
<td>13%</td>
</tr>
<tr>
<td>Total</td>
<td>200,000</td>
<td>100%</td>
</tr>
</tbody>
</table>

Off-focus utterances  Utterances inappropriate for their associated foci. Although the utterances in the database are created from PASs in which the focus and subject are explicitly marked by the topic marker and case marker, respectively, the focus and content of an utterance are often not closely associated. This occurs when there is an error in the PAS analysis or when the meaning of the focus is just too broad or vague.

We investigated the current quality of the database in terms of how many non-sentences and off-focus utterances are contained. For this purpose, we performed annotations regarding non-sentence and off-focus utterances, which are described below.

Non-sentence annotation  We randomly sampled 200,000 utterances from the utterance database. The annotators labeled each utterance with the following instructions:

- If you think the utterance is a non-sentence, label it 0.
- If you do not think the utterance is a non-sentence (i.e., it is a valid-sentence), label it 1.

A total of 24 annotators participated; two annotators were randomly assigned to each utterance. Cohen’s $\kappa$ value, which assesses the agreement between the two annotators, was calculated as 0.56. This indicates an intermediate degree of agreement. Table 1 lists annotation examples, and Table 2 gives the annotation breakdown. Non-sentences accounted for 12% of the database. Hereafter, we call the non-sentence annotation data on which the annotators agreed “the non-sentence corpus” (containing 174,007 utterances).

Table 3: Examples of focus annotation (0: off-focus, 1: on-focus). Annotators 1 and 2 give labels by two different annotators (A1 and A2).

<table>
<thead>
<tr>
<th>Focus</th>
<th>Utterance</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>秋冬 (Fall &amp; winter)</td>
<td>車価が高いですか? (Is the unit price high?)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>秋冬 (Fall &amp; winter)</td>
<td>グーブが多いのでしょうか? (Are there a lot of boots?)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>秋冬 (Fall &amp; winter)</td>
<td>空気が混ざってるんですか? (Is air clear?)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Statistics of focus annotation (0: Off-focus, 1: On-focus)

<table>
<thead>
<tr>
<th></th>
<th># of utterances</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 annotators labeled 0</td>
<td>7,528</td>
<td>5%</td>
</tr>
<tr>
<td>2 annotators labeled 1</td>
<td>121,511</td>
<td>80%</td>
</tr>
<tr>
<td>1 annotator labeled 0, other labeled 1</td>
<td>21,916</td>
<td>15%</td>
</tr>
<tr>
<td>Total</td>
<td>150,955</td>
<td>100%</td>
</tr>
</tbody>
</table>

Focus annotation  By using the utterances annotated as valid-sentences in the non-sentence corpus (i.e., 150,955 utterances), two annotators labeled whether the utterances were appropriate to their foci. The annotators were shown pairs of a focus and utterance and labeled each pair with the following instructions:

- If you feel the combination of utterance and focus is unnatural, label it 0 (off-focus).
- If you feel the combination of utterance and focus is natural, label it 1 (on-focus). When the focus has multiple meanings, if there is at least one reasonable interpretation, label the combination 1.

A total of 24 annotators participated; pairs of annotators were randomly selected for labeling pairs of a focus and utterance. Cohen’s $\kappa$ value was 0.32, which indicates a reasonable degree of agreement when considering the subjective nature of judging naturalness. Table 3 shows an example of this annotation, and Table 4 gives the annotation breakdown. Utterances inappropriate for their associated foci accounted for 5% of the database. Hereafter, we call the focus annotation data on which the annotators agreed "the focus corpus" (containing 129,039 utterances).

4 Proposed methods  We found that there are 12% non-sentences and 5% utterances inappropriate for their associated foci in our database. Since this means the system utterances can often be erroneous, we need to reduce these utterances to improve the quality of our database. We also see it as a problem that the utterances in our database are monotonous and uninformative because they were generated from single PASs.

In this paper, we propose methods of filtering non-sentences and off-focus utterances for refining the database. We also propose a method to concatenate pairs of utterances about the same focus to reduce monotony of system utterances.
4.1 Method for creating non-sentence filter

Since the detection of non-sentences can be regarded as a task of sentence classification, we created a non-sentence filter by using machine-learning methods. We used standard machine-learning methods for sentence classification such as SVM and neural-network-based methods, which have been extensively used in recent years. We used the following machine-learning methods for training our classifiers:

SVM We train an SVM classifier with a linear kernel. The features are the averaged word vectors of words contained in an utterance. We use a pretrained word vector provided by Suzuki et al. [23], the dimensions of which are 200. We use the same pretrained word vectors for MLP, CNN, and LSTM, which we describe below.

Multi-Layer Perceptron (MLP) We train a classifier by MLP. We have five layers: the input layer, three non-linear layers (each layer having 200 units) with sigmoid activation, and the output layer. We use averaged word vectors as input. The output layer outputs a binary decision by a softmax function.

Convolutional Neural Network (CNN) We train a classifier by a CNN. We have an input layer, a convolutional layer, a pooling layer, and an output layer. The model structure is the same as that used by Kim [24]. A filter whose size is 200 × 3 is used for convolution. The stride is set to one. We used relu as an activation function. The max pooling layer uses a window size of three to output a fixed length vector. The output layer outputs a binary decision by a softmax function.

Long Short-Term Memory (LSTM) We train a classifier by LSTM. We have an input layer, an LSTM layer, three hidden layers, and an output layer. The LSTM layer has 200 units. Each word is converted into an embedding, and the sequence of word embeddings is converted into a hidden representation, corresponding to a sentence vector. Then, this vector is fed to three non-linear layers (each layer having 200 units) with sigmoid activation, the output of which is input to the output layer, making a binary decision by a softmax function.

4.2 Method for creating focus filter

To filter out off-focus utterances, we use co-occurrence statistics, namely, point-wise mutual information (PMI) between the subject of the utterance and its focus. We use PMI because it has been successfully used to filter sentences unrelated to topics [25]. We calculate the PMI with the following equation:

$$\text{PMI}(S, F) = \log_2 \frac{\text{count}(S, F) / N}{\text{count}(S) / N \times \text{count}(F) / N}$$

where $S$ is a subject; $F$ is a focus; ‘count’ is a function that returns the number of documents containing $S$, $F$, or both; and $N$ is the maximum number of documents in a text database. We use a sentence as a document unit.

If the PMI value is below a certain threshold, we can filter the utterance because the association can be considered low. The threshold can be determined experimentally, that is, we find the threshold that produces the best accuracy using training/development data. Note that the best accuracy depends on the objective. If we want the resulting database to be as clean as possible, we can set a high threshold. If we do not want to lose much data, the threshold can be set lower. In this study, we set the target recall for detecting off-focus utterances to 80% because we want most off-focus utterances removed. We determine the threshold that achieves this recall on the training/development data and use it for filtering possible off-focus utterances.

Note that an appropriate text database must be chosen for calculating the PMI. We consider using Wikipedia (containing roughly 8M sentences) and blogs (we use one year’s worth of blogs containing about 2B sentences). The former is smaller but more informational. The latter is larger but noisy and is a mixture of contents of varying quality. We will verify which one is more useful in a later experiment, although we naturally assume that blog data are more suitable because they have more variety, which is a requirement for chat-oriented dialogue systems.

4.3 Utterance concatenation

For one solution to reduce monotony, we propose a method of concatenating pairs of automatically generated utterances about the same focus so that the utterances can be longer and richer in content. More specifically, we propose concatenating two random utterances that have the same focus.

Although this approach may seem simplistic, it can be effective because, at the very least, it increases the utterance length of a system. Note that it is not trivial to create a reasonable utterance by concatenating two utterances. It has been shown that implicit discourse relations are still hard to detect [26]. This means that utterances that will be coherent in terms of discourse are difficult to accurately select. In addition, we believe our simple concatenation method may just work because the concatenated utterance will satisfy the local coherence [27] with the same underlying entity (i.e., the focus).

5 Evaluation

We first individually evaluated the performance of our non-sentence and focus filtering methods and then conducted a subjective evaluation involving human participants on the filtered and concatenated utterances.

5.1 Evaluation of our non-sentence filtering methods

We trained a non-sentence filter by using the non-sentence corpus (see Section 3.2). We split the data into training, development, and test sets corresponding to 3837, 500, and 500 foci, respectively.

We trained the classifiers using the training data and evaluated the accuracy with the test data by using the highest
Table 5: Precision, recall, and F-measure for the detection of non-sentences. Bold font represents top score for each evaluation criterion.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.93</td>
<td>0.81</td>
<td>0.71</td>
<td>0.76</td>
</tr>
<tr>
<td>MLP</td>
<td>0.90</td>
<td>0.63</td>
<td>0.84</td>
<td>0.72</td>
</tr>
<tr>
<td>CNN</td>
<td>0.94</td>
<td>0.86</td>
<td>0.75</td>
<td>0.79</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.95</td>
<td>0.88</td>
<td>0.78</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Table 6: Precision, recall, and F-measure for off-focus/on-focus utterances for training and test data when thresholds of 2.2 and 2.8 are used for Wikipedia and blog data, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>off-focus</td>
<td>0.09</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>on-focus</td>
<td>0.98</td>
<td>0.49</td>
</tr>
<tr>
<td>Blog data</td>
<td>off-focus</td>
<td>0.09</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>on-focus</td>
<td>0.97</td>
<td>0.42</td>
</tr>
</tbody>
</table>

F-measure model yielded from the development data\(^1\). The classification results are listed in Table 5. We can see that our method successfully detected the non-sentences with high accuracy. The model that uses LSTM had the highest accuracy (0.95) and F-measure (0.83). LSTM has the highest accuracy probably because the determination of non-sentences depends on the sequence of words that can best be captured with recurrent models.

### 5.2 Evaluation of our focus filtering method

We split the focus corpus (see Section 3.2) into 80% training data and 20% test data. We first calculated the PMI values between the subjects and foci for all utterances by using the training data. Then, we looked for the threshold of the PMI that achieved 80% recall for off-focus utterances through a grid search.

When we used Wikipedia as the data for PMI calculation, we obtained a threshold of 2.2, and when we used the blog data, the threshold was 2.8. See Figures 1 and 2 for the changes in precision, recall, and F-measure when we changed the threshold by an interval of 0.1. Table 6 shows the precision, recall, and F-measure for off-focus/on-focus utterances for training and test data when the thresholds of 2.2 and 2.8 are used for Wikipedia and the blog data, respectively. As expected, the use of blog data yielded much better results, resulting in higher precision/recall for on-focus utterances at the point of 80% recall for off-focus utterances. The results indicate that our off-focus filter can successfully filter utterances that are not associated with their foci (off-focus utterances).

\(^1\)Note that for SVM, we used the training data for training and the test data for evaluation; we did not use the development data.

### 5.3 Subjective evaluation

We conducted a subjective evaluation involving human participants to verify the effectiveness of our non-sentence and focus filtering methods as well as our concatenation method (see Section 4.3).

#### Evaluation procedure

Four participants took part in the evaluation. We made each of eight methods for comparison (see the following subsection for details) generate utterances for 100 randomly selected foci, resulting in 800 utterances (8×100 foci) for use in the experiment. The utterances were randomly shuffled and presented to the participants. Each participant rated the 800 utterances in terms of familiarity, understandability, and content richness (we describe these criteria later).

#### Methods for comparison

We compared the following eight methods (a)–(h). Note that for non-sentence filtering, we use the LSTM model, which showed the best performance in our experiment. For focus filtering, we use the PMI threshold of 2.8 calculated by using the blog data.

(a) Random (Single): Baseline

We randomly select a single utterance from the utterance database.

(b) Random (Pair): Proposed

We randomly select two utterances from the utterance database and concatenate them to create a system utterance.

(c) NS-filtered (Single): Proposed

We randomly select one utterance from the test data of the non-sentence corpus that was classified as a valid-sentence with non-sentence filtering.

(d) NS-filtered (Pair): Proposed

We randomly select two utterances from the test data of the non-sentence corpus that were classified as valid-sentences with non-sentence filtering. Then we concatenate these utterances to create a system utterance.

(e) NS+F-filtered (Single): Proposed

We randomly select one utterance from the test data of the non-sentence corpus that was classified as a valid-sentence with non-sentence filtering and as on-focus with focus filtering.

(f) NS+F-filtered (Pair): Proposed

We randomly select two utterances from the test data of the non-sentence corpus that were classified as valid-sentences with non-sentence filtering and as on-focus with focus filtering. Then we concatenate these utterances to create a system utterance.

(g) Gold NS (Single)

We randomly select one utterance annotated as a valid sentence in the test data of the non-sentences corpus.

(h) Gold F (Single)

We randomly select one utterance annotated as on-focus in the test data of the focus corpus.
Random (Single) is the baseline, which is our current method of just using a single utterance for a given focus from the utterance database. Table 7 lists example utterances generated by the eight methods.

**Evaluation criteria**

Sugiyama et al. [29] used the semantic differential (SD) method to derive the dimensions to evaluate utterances in chat-oriented dialogue systems. They identified three dimensions, and we used them in our evaluation. The evaluation criteria together with the statements used in the evaluation were as follows:

- **Familiarity:** You feel familiar with the system and that you want to talk more.
- **Content Richness:** You feel that the utterance is interesting and informative.
- **Understandability:** You feel that the utterance is natural and easy to understand.

Each participant rated their level of agreement to the above statements using a Likert scale between 1 and 5, where 5 indicates the highest agreement.

**Results**

Table 8 lists the evaluation results. By comparing (a) Random (Single) to (c) NS-filtered (Single), we can see that understandability and familiarity were improved by using nonsentence filtering. By comparing (c) NS-filtered (Single) to (e) NS+F-filtered (Single), although there was no significant difference, we can see that understandability further improved. Since both (c) NS-filtered (Single) and (e) NS+F-filtered (Single) significantly outperform the baseline, this verifies the effectiveness of our filters. In addition, by comparing (g) Gold NS (Single) to (h) Gold F (Single), we can confirm that utterances need to be appropriate for their associated foci. The results here indicate that our filters contribute greatly to the understandability of the utterances in the utterance database. In addition, we surprisingly also see improvements in familiarity and content richness.

By comparing (a) Random (Single) to (b) Random (Pair), although content richness improved, we can see that understandability significantly decreased. This means that just...
randomly concatenating utterances in the current utterance database for the same focus does not lead to good utterances. However, by comparing (a) Random (Single) to (d) NS-filtered (Pair), we can see that our concatenation method improved familiarity and content richness while maintaining understandability. By comparing (d) NS-filtered (Pair) to (f) NS+F-filtered (Pair), we can see further improvements in content richness and understandability. Although it does not seem to be a good idea to concatenate possibly low-quality utterances, it is a good idea to concatenate valid and on-focus utterances. Because content richness has improved without loss of understandability, we can safely say that our concatenation method can reduce the monotony and generate richer utterances.

6 Summary and future work

To refine our utterance database and generate non-monotonous utterances, we proposed methods of filtering non-sentences and utterances inappropriate for their associated foci using neural-network-based methods and co-occurrence statistics. To reduce monotony, we also proposed a simple but powerful method of concatenating two utterances related to the same focus so that the utterances can be longer and richer in content. Experimental results show that our non-sentence filter can successfully remove non-sentences with an accuracy of 95% and that we can filter utterances inappropriately for their foci with high recall. Also, we examined the effectiveness of our filtering methods and concatenation method through an experiment involving human participants. Experimental results show that our automatic methods of incorporating non-sentence and focus filtering significantly outperformed the current single-utterance baseline. The experimental results also indicate that the concatenation of two utterances leads to higher familiarity and content richness while maintaining understandability. We believe our proposed methods can especially contribute to commercial chat-oriented dialogue systems, in which the quality of utterances is critical.

For future work, we plan to update the utterance database of our current chat-oriented dialogue system with our filtering methods and concatenation method. We also plan to consider methods of concatenating two utterances more appropriately, for example, by taking discourse relations [30; 26] into account.

References


[8] Joseph Weizenbaum. ELIZA— a computer program for the study of natural language communication between


Improving Goal-Oriented Visual Dialog Agents via Advanced Recurrent Nets with Tempered Policy Gradient

Rui Zhao and Volker Tresp
Siemens AG, Corporate Technology, Munich, Germany
Ludwig-Maximilians-Universität München, Munich, Germany
{ruizhao, volker.tresp}@siemens.com

Abstract
Learning goal-oriented dialogues by means of deep reinforcement learning has recently become a popular research topic. However, training text-generating agents efficiently is still a considerable challenge. Commonly used policy-based dialogue agents often end up focusing on simple utterances and suboptimal policies. To mitigate this problem, we propose a class of novel temperature-based extensions for policy gradient methods, which are referred to as Tempered Policy Gradients (TPGs). These methods encourage exploration with different temperature control strategies. We derive three variations of the TPGs and show their superior performance on a recently published AI-testbed, i.e., the GuessWhat?! game. On the testbed, we achieve significant improvements with two innovations. The first one is an extension of the state-of-the-art solutions with Seq2Seq and Memory Network structures that leads to an improvement of 9%. The second one is the application of our newly developed TPG methods, which improves the performance additionally by around 5% and, even more importantly, helps produce more convincing utterances. TPG can easily be applied to any goal-oriented dialogue systems.

1 Introduction
In recent years, deep learning has shown convincing performance in various areas such as image recognition, speech recognition, and natural language processing (NLP). Deep neural nets are capable of learning complex dependencies from huge amounts of data and its human generated annotations in a supervised way. In contrast, reinforcement learning agents [Sutton and Barto, 1998] can learn directly from their interactions with the environment without any supervision and surpass human performance in several domains, for instance in the game of GO [Silver et al., 2016], as well as many computer games [Mnih et al., 2015]. In this paper we are concerned with the application of both approaches to goal-oriented dialogue systems [Bordes and Weston, 2017; de Vries et al., 2017; Das et al., 2017; Strub et al., 2017; Das et al., 2017; Lewis et al., 2017; Dhingra et al., 2016], a problem that has recently caught the attention of machine learning researchers. De Vries et al. [2017] have proposed as AI-testbed a visual grounded object guessing game called GuessWhat?!.. Das et al. [2017] formulated a visual dialogue system which is about two chatbots asking and answering questions to identify a specific image within a group of images. More practically, dialogue agents have been applied to negotiate a deal [Lewis et al., 2017] and access certain information from knowledge bases [Dhingra et al., 2016]. The essential idea in these systems is to train different dialogue agents to accomplish the tasks. In those papers, the agents have been trained with policy gradients, i.e. REINFORCE [Williams, 1992].

In order to improve the exploration quality of policy gradients, we present three instances of temperature-based methods. The first one is a single-temperature approach which is very easy to apply. The second one is a parallel approach with multiple temperature policies running concurrently. This second approach is more demanding on computational resources, but results in more stable solutions. The third one is a temperature policy approach that dynamically adjusts the temperature for each action at each time-step, based on action frequencies. This dynamic method is more sophisticated and proves more efficient in the experiments. In the experiments, all these methods demonstrate better exploration strategies in comparison to the plain policy gradient.

We demonstrate our approaches using a real-world dataset called GuessWhat?!.. The GuessWhat?! game [de Vries et al., 2017] is a visual object discovery game between two players, the Oracle and the Questioner. The Questioner tries to identify an object by asking the Oracle questions. The original works [de Vries et al., 2017; Strub et al., 2017] first proposed supervised learning to simulate and optimize the game. Strub et al. [2017] showed that the performance could be improved by applying plain policy gradient reinforcement learning, which maximizes the game success rate, as a second processing step. Building on these previous works, we propose two network architecture extensions. We utilize a Seq2Seq model [Sutskever et al., 2014] to process the image along with the historical dialogues for question generation. For the guessing task, we develop a Memory Network [Sukhbaatar et al., 2015] with Attention Mechanism [Bahdanau et al., 2014] to process the generated question-answer pairs. We first train these two models using the plain policy gradient and use them
as our baselines. Subsequently, we train the models with our new TPG methods and compare the performances with the baselines. We show that the TPG method is compatible with state-of-the-art architectures such as Seq2Seq and Memory Networks and contributes orthogonally to these advanced neural architectures. To the best of our knowledge, the presented work is the first to propose temperature-based policy gradient methods to leverage exploration and exploitation in the field of goal-oriented dialogue systems. We demonstrate the superior performance of our TPG methods by applying it to the GuessWhat?! game. Our contributions are:

- We introduce Tempered Policy Gradients in the context of goal-oriented dialogue systems, a novel class of approaches to temperature control to better leverage exploration and exploitation during training.

- We extend the state-of-the-art solutions for the Guess-What?! game by integrating Seq2Seq and Memory Networks. We show that TPGs are compatible with these advanced models and further improve the performance.

2 Preliminaries

In our notation, we use \( \mathbf{x} \) to denote the input to a policy network \( \pi \), and \( \mathbf{z} \) to denote the \( i \)-th element of the input vector. Similarly, \( \mathbf{w} \) denotes the weight vector of \( \pi \), and \( \mathbf{w}_i \) denotes the \( i \)-th element of the weight vector of that \( \pi \). The output \( y \) is a multinomial random variable with \( N \) states that follows a probability mass function, \( f(y = n \mid \pi(\mathbf{x} \mid \mathbf{w})) \), where \( \sum_{n=1}^{N} f(y = n \mid \pi(\mathbf{x} \mid \mathbf{w})) = 1 \) and \( f(\cdot) \geq 0 \). In a nutshell, a policy network parametrizes a probabilistic unit that produces the sampled output, mathematically, \( y \sim f(\pi(\mathbf{x} \mid \mathbf{w})) \).

At this point, we have defined the policy neural net and now discuss performance measures commonly used for optimizations. Typically, the expected value of the accumulated reward, i.e., return, conditioned on the policy network parameters \( E(r \mid \mathbf{w}) \) is used. Here, \( E \) denotes the expectation operator, \( r \) the accumulated reward signal, and \( \mathbf{w} \) the network weight vector. The objective of reinforcement learning is to update the weights in a way that maximizes the expected return at each trial. In particular, the REINFORCE updating rule is: \( \Delta \mathbf{w}_i = \alpha_i (r - b_i) e_i \), where \( \Delta \mathbf{w}_i \) denotes the weight adjustment of weight \( \mathbf{w}_i \), \( \alpha_i \) is a non-negative learning rate factor, and \( b_i \) is a reinforcement baseline. The \( e_i \) is the characteristic eligibility of \( \mathbf{w}_i \), defined as \( e_i = \langle \partial f / \partial \mathbf{w}_i \rangle / f = \partial \ln f / \partial \mathbf{w}_i \). Williams [1992] has proved that the updating quantity, \( (r - b_i) \partial \ln f / \partial \mathbf{w}_i \), represents an unbiased estimate of \( \partial E(r \mid \mathbf{w}) / \partial \mathbf{w}_i \).

3 Tempered Policy Gradient

In order to improve the exploration quality of REINFORCE in the task of optimizing policy-based dialogue agents, we attempt to find the optimal compromise between exploration and exploitation. In TPGs we introduce a parameter \( \tau \), the sampling temperature of the probabilistic output unit, which allows us to explicitly control the strengths of the exploration.

3.1 Exploration and Exploitation

The trade-off between exploration and exploitation is one of the great challenges in reinforcement learning [Sutton and Barto, 1998]. To obtain a high reward, an agent must exploit the actions that have already proved effective in getting more rewards. However, to discover such actions, the agent must try actions, which appear suboptimal, to explore the action space. In a stochastic task like text generation, each action, i.e., a word, must be tried many times to find out whether it is a reliable choice or not. The exploration-exploitation dilemma has been intensively studied over many decades [Carmel and Markovitch, 1999; Nachum et al., 2016; Liu et al., 2017]. Finding the balance between exploration and exploitation is considered crucial for the success of reinforcement learning [Thrun, 1992].

3.2 Temperature Sampling

In text generation, it is well-known that the simple trick of temperature adjustment is sufficient to shift the language model to be more conservative or more diversified [Karpathy and Fei-Fei, 2015]. In order to control the trade-off between exploration and exploitation, we borrow the strength of the temperature parameter \( \tau \geq 0 \) to control the sampling. The output probability of each word is transformed by a temperature function as:

\[
\hat{f}^\tau(y = n \mid \pi(\mathbf{x} \mid \mathbf{w})) = \frac{f(y = n \mid \pi(\mathbf{x} \mid \mathbf{w}))^{\frac{1}{\tau}}}{\sum_{m=1}^{N} f(y = m \mid \pi(\mathbf{x} \mid \mathbf{w}))^{\frac{1}{\tau}}}
\]

We use notation \( \hat{f}^\tau \) to denote a probability mass function that is transferred by a temperature function with temperature \( \tau \). When the temperature is high, \( \tau > 1 \), the distribution becomes more uniform; when the temperature is low, \( \tau < 1 \), the distribution appears more spiky. TPG is defined as an extended algorithm of the Monte Carlo Policy Gradient approach. We use a higher temperature, \( \tau > 1 \), to encourage the model to explore in the action space, and conversely, a lower temperature, \( \tau < 1 \), to encourage exploitation. In the extreme case, when \( \tau = 0 \), we obtain greedy search.

3.3 Tempered Policy Gradient Methods

Here, we introduce three instances of TPGs in the domain of goal-oriented dialogues, including single, parallel, and dynamic tempered policy gradient methods.

**Single-TPG:** The Single-TPG method simply uses a global temperature \( \tau_{\text{global}} \) during the whole training process, i.e., we use \( \tau_{\text{global}} > 1 \) to encourage exploration. The forward pass is represented mathematically as: \( y_{\text{global}} \sim \hat{f}_{\text{global}}(\pi(\mathbf{x} \mid \mathbf{w})) \), where \( \pi(\mathbf{x} \mid \mathbf{w}) \) represents a policy neural network that parametrizes a distribution \( \hat{f}_{\text{global}} \) over the vocabulary, and \( y_{\text{global}} \) means the word sampled from this tempered word distribution. After sampling, the weight of the neural net is updated using:

\[
\Delta \mathbf{w}_i = \alpha_i (r - b_i) \partial \ln f(y_{\text{global}} \mid \pi(\mathbf{x} \mid \mathbf{w})) / \partial \mathbf{w}_i.
\]

Noteworthy is that the actual gradient, \( \partial \ln f(y_{\text{global}} \mid \pi(\mathbf{x} \mid \mathbf{w})) / \partial \mathbf{w}_i \), depends on the sampled word, \( y_{\text{global}} \), however, does not depend directly on the temperature, \( \tau \). We prefer to find the words that lead to a reward, so that the model can learn quickly from these actions, otherwise, the neural network only learns to avoid current failure actions. With Single-TPG and \( \tau > 1 \), the entire vocabulary of a dialogue
LSTM

Is it a person?

Formula is given by

The weights are updated with the sum of gradients. The forward pass, multiple copies of the neural nets parameterize multiple word distributions. The words are sampled in parallel at various temperatures, mathematically, \( y_1^n, \ldots, y_n^n \sim f^{\tau_1,\ldots,\tau_n}(\pi(x \mid w)) \). After sampling, in the backward pass, the weights are updated with the sum of gradients. The formula is given by

\[
\Delta w_i = \Sigma_k \alpha_i (r - b_i) \partial \ln f(y_i^n \mid \pi(x \mid w)) / \partial w_i,
\]

where \( k \in \{1, \ldots, n\} \). The combinational use of higher and lower temperatures ensures both exploration and exploitation at the same time. The sum over weights updates of parallel policies gives a more accurate Monte Carlo estimate of \( \partial E(r \mid w) / \partial w \), due to the nature of Monte Carlo methods [Robert, 2004].

Thus, compared to Single-TPG, we would argue that Parallel-TPG is more robust and stable, although Parallel-TPG needs more computational power. However, these computations can be easily distributed in a parallel fashion using state-of-the-art graphics processing units.

Dynamic-TPG: As a third variant, we introduce the Dynamic-TPG, which is the most sophisticated approach in the current TPG family. The essential idea is that we use a heuristic function \( h \) to assign the temperature \( \tau \) to the word distribution at each time step, \( t \). The temperature is bounded in a predefined range \( [\tau_{\min}, \tau_{\max}] \). The heuristic function we used here is based upon the term frequency inverse document frequency, tf-idf [Leskovec et al., 2014]. In the context of goal-oriented dialogues, we use the counted number of each word as term frequency \( f \) and the total number of generated dialogues during training as document frequency \( df \). We use the word that has the highest probability to be sampled at current time-step, \( y_t^i \), as the input to the heuristic function \( h \). Here, \( y_t^i \) is the maximizer of the probability mass function \( f \). Mathematically, it is defined as \( y_t^i = \arg \max_i (f(x \mid w)) \). We propose that tf-idf \((y_t^i)\) approximates the concentration level of the distribution, which means that if the same word is always sampled from a distribution, then the distribution is very concentrated. Too much concentration prevents the model from exploration, so that a higher temperature is needed. In order to achieve this effect, the heuristic function is defined as

\[
\tau_t^h = h(tf-idf(y_t^i)) = \tau_{\min} + (\tau_{\max} - \tau_{\min}) \frac{tf-idf(y_t^i) - tf-idf_{\min}}{tf-idf_{\max} - tf-idf_{\min}}.
\]

With this heuristic, words that occur very often are depressed by applying a higher temperature to those words, making them less likely to be selected in the near future. In the forward pass, a word is sampled using \( y_t^o \sim f^{\tau_t^h}(\pi(x \mid w)) \). In the backward pass, the weights are updated correspondingly, using

\[
\Delta w_i = \alpha_i (r - b_i) \partial \ln f(y_i^h \mid \pi(x \mid w)) / \partial w_i,
\]

where \( \tau_t^h \) is the temperature calculated from the heuristic function. Compared to Parallel-TPG, the advantage of Dynamic-TPG is that it assigns temperature more appropriately, without increasing the computational load.

4 GuessWhat?! Game

We evaluate our concepts using a recent testbed for AL called the GuessWhat?! game [de Vries et al., 2017], available at https://guesswhat.ai. The dataset consists of 155 k dialogues, including 822 k question-answer pairs, each composed of around 5 k words, about 67 k images [Lin et al., 2014] and 134 k objects. The game is about visual object discovery trough a multi-round QA among different players.

Formally, a GuessWhat?! game is represented by a tuple \((I, D, O, o^*)\), where \( I \in \mathbb{R}^{H \times W} \) denotes an image of height \( H \) and width \( W \); \( D \) represents a dialogue composed of \( M \) rounds of question-answer pairs (QAs), \( D = (q_m, a_m)_{m=1}^M \); \( O \) stands for a list of \( K \) objects \( O = (o_k)_{k=1}^K \), and \( o^* \) is the target object. Each question is a sequence of words, \( q_m = \{y_m,1, \ldots, y_m,N_m\} \) with length \( N_m \). The words are taken from a defined vocabulary \( V \), which consists of the words and a start token and an end token. Each answer is either yes, no, or not applicable, i.e. \( a_m \in \{yes, no, n.a.\} \). For each object \( o_k \), there is a corresponding object category \( c_k \in \{1, \ldots, C\} \) and a pixel-wise segmentation mask \( S_k \in \{0, 1\}^{H \times W} \). Finally, we use colon notation (:) to select a subset of a sequence, for instance, \((q, a)_{1:m}\) refers to the first \( m \) rounds of QAs in a dialogue.

4.1 Models and Pretraining

Following [Strub et al., 2017], we first train all three models in a supervised fashion.

Oracle: The task of the Oracle is to answer questions regarding to the target object. We outline here the simple neural network architecture that achieved the best performance in the study of [de Vries et al., 2017], and which we also used in our experiments. The input information used here is of three modalities, namely the question \( q \), the spatial information \( x_{spatial} \) and the category \( c^* \) of the target object. For encoding the question, de Vries et al. first use a lookup table to learn the embedding of words, then
use a one layer long-short-term-memory (LSTM) [Hochreiter and Schmidhuber, 1997] to encode the whole question. For spatial information, de Vries et al. extract an 8-dimensional vector of the location of the bounding box $[x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}]$, where $x, y$ denote the coordinates and $w_{box}, h_{box}$ denote the width and height of the bounding box, respectively. De Vries et al. normalize the image width and height so that the coordinates range from -1 to 1. The origin is at the image center. The category embedding of the object is also learned with a lookup table during training. At the last step, de Vries et al. concatenate all three embeddings into one feature vector and fed it into a one hidden layer multilayer perceptron (MLP). The softmax output layer predicts the distribution, Oracle $: = p(a \ | \ q, c, x_{spatial}^O)$, over the three classes, including no, yes, and not applicable. The model is trained using the negative log-likelihood criterion. The Oracle structure is shown in Fig. 1.

**Question-Generator:** The goal of the Question-Generator (QGen) is to ask the Oracle meaningful questions, $q_{m+1}$, given the whole image, $I$, and the historical question-answer pairs, $(q, a)_{1:m}$. In previous work [Strub et al., 2017], the state transition function was modelled as an LSTM, which was trained using whole dialogues so that the model memorizes the historical QAs. We refer to this as dialogue level training. We develop a novel QGen architecture using a modified version of the Seq2Seq model [Sutskever et al., 2014]. The modified Seq2Seq model enables question level training, which means that the model is fed with historical QAs, and then learns to produce a new question. Following [Strub et al., 2017], we first encode the whole image into a fixed-size feature vector using the VGG-net [Simonyan and Zisserman, 2014]. The features come from the fc-8 layer of the VGG-net. For processing historical QAs, we use a lookup table to learn the word embeddings, then again use an LSTM encoder to encode the history information into a fixed-size latent representation, and concatenate it with the image representation:

$$s_{m,N,m}^{enc} = \text{encoder}(\text{LSTM}(q, a)_{1:m}, \text{VGG}(I)).$$

The encoder and decoder are coupled by initializing the decoder state with the last encoder state, mathematically, $s_{m+1,0}^{dec} = s_{m,N,m}^{enc}$. The LSTM decoder generates each word based on the concatenated representation and the previous generated word (note the first word is a start token):

$$y_{m+1,n} = \text{decoder}(\text{LSTM}((y_{m+1,n-1}, s_{m+1,n-1}^{dec}))$$

The decoder shares the same lookup table weights as the encoder. The Seq2Seq model, consisting of the encoder and the decoder, is trained end-to-end to minimize the negative log-likelihood cost. During testing, the decoder gets a start token and the representation from the encoder, and then generates each word at each time step until it encounters a question mark token, QGen $: = p(y_{m+1,n} \ | \ (q, a)_{1:m}, I)$. The output is a complete question. After several question-answer rounds, the QGen outputs an end-of-dialogue token, and stops asking questions. The overall structure of the QGen model is illustrated in Fig. 2.

**Guesser:** The goal of the Guesser model is to find out which object the Oracle model is referring to, given the complete history of the dialogue and a list of objects in the image, $p(o^* \ | \ (q, a)_{1:M}, x_{spatial}^O, c^O)$. The Guesser model has access to the spatial, $x_{spatial}^O$, and category information, $c^O$, of the objects in the list. The task of the Guesser model is challenging because it needs to understand the dialogue and to focus on the important content, and then guess the object. To accomplish this task, we decided to integrate the Memory [Sukhbaatar et al., 2015] and Attention [Bahdanau et al., 2014] modules into the Guesser architecture used in the previous work [Strub et al., 2017]. First, we use an LSTM header to process the varying lengths of question-answer pairs in parallel into multiple fixed-size vectors. Here, each QA-pair has been encoded into some facts, $\text{Fact}_m = \text{LSTM}((q, a)_m)$, and stored into a memory base. Later, we use the sum of the spatial and category embeddings of all objects as a key, $\text{Key}_1 = \text{MLP}(x_{spatial}^O,c^O)$, to query the memory and calculate an attention mask, $\text{Attention}_1(\text{Fact}_m) = \text{Fact}_m \circ \text{Key}_1$, over each fact. Next, we use the sum of attended facts and the first key to calculate the second key. Further, we use the second key to query the memory base again to have a more accurate attention. These are the so called “two-hops” of attention in the literature [Sukhbaatar et al., 2015]. Finally, we compare the attended facts with each object embedding in the list using a dot product. The most similar object to these facts is the prediction, Guesser $: = p(o^* \ | \ (q, a)_{1:M}, x_{spatial}^O, c^O)$. The intention of using the attention module here is to find out the most relevant descriptions or facts concerning the candidate objects. We train the whole Guesser network end-to-end using the negative log-likelihood criterion. A more graphical description of the Guesser model is shown in Fig. 3.

### 4.2 Reinforcement Learning

Now, we post-train the QGen and the Guesser model with reinforcement learning. We keep the Oracle model fixed. In each game episode, when the models find the correct object, $r = 1$, otherwise, $r = 0$. 
Next, we can assign credits for each action of the QGen and the Guesser models. In the case of the QGen model, we spread the reward uniformly over the sequence of actions in the episode. The baseline function, \( b \), used here is the running average of the game success rate. Consider that the Guesser model has only one action in each episode, i.e., taking the guess. If the Guesser finds the correct object, then it gets an immediate reward and the Guesser’s parameters are updated using the REINFORCE rule without baseline. The QGen is trained using the following four methods.

**REINFORCE:** The baseline method used here is REINFORCE [Williams, 1992]. During training, in the forward pass the words are sampled with \( r = 1 \), \( y_{m+1,n} \sim f(QGen(x | w)) \). In the backward pass, the weights are updated using REINFORCE, as,

\[
w = w + \alpha(r - b)\nabla_w \ln f(y_{m+1,n} | QGen(x | w)).
\]

**Single-TPG:** We use temperature \( \tau_{global} = 1.5 \) during training to encourage exploration, mathematically, \( y_{m+1,n} \sim f^{\tau_{global}}(QGen(x | w)) \). In the backward pass, the weights are updated using

\[
w = w + \alpha(r - b)\nabla_w \ln f(y_{m+1,n} | QGen(x | w)).
\]

**Parallel-TPG:** For Parallel-TPG, we use two temperatures \( \tau_1 = 1.0 \) and \( \tau_2 = 1.5 \) to encourage the exploration. The words are sampled in the forward pass using \( y_{m+1,n} \sim f^{\tau_1,\tau_2}(QGen(x | w)) \). In the backward pass, the weights are updated using

\[
w = w + \alpha(r - b)\nabla_w \ln f(y_{m+1,n} | QGen(x | w)).
\]

**Dynamic-TPG:** The last method we evaluated is Dynamic-TPG, using the REINFORCE rule without baseline. The words are sampled in the forward pass using \( y_{m+1,n} \sim f^{\tau}(QGen(x | w)) \). In the backward pass, the weights are updated using

\[
w = w + \alpha(r - b)\nabla_w \ln f(y_{m+1,n} | QGen(x | w)).
\]

**5 Experiment**

We first train all the networks in a supervised fashion, and then further optimize the QGen and the Guesser model using reinforcement learning. The source code is available at https://github.com/ruizhaogit/GuessWhat-TemperedPolicyGradient, which uses Torch7 [Collobert et al., 2011].

### Table 1: Performance comparison of our methods to other methods reported in literature after reinforcement learning

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[Strub et al., 2017]</td>
<td>52.30%</td>
</tr>
<tr>
<td>2</td>
<td>[Strub and de Vries, 2017]</td>
<td>60.30%</td>
</tr>
<tr>
<td>3</td>
<td>REINFORCE</td>
<td>69.66%</td>
</tr>
<tr>
<td>4</td>
<td>Single-TPG</td>
<td>69.76%</td>
</tr>
<tr>
<td>5</td>
<td>Parallel-TPG</td>
<td>73.86%</td>
</tr>
<tr>
<td>6</td>
<td>Dynamic-TPG</td>
<td>74.31%</td>
</tr>
</tbody>
</table>

5.1 Pretraining

We train all three models using 0.5 dropout [Srivastava et al., 2014] during training, using the ADAM optimizer [Kingma and Ba, 2014]. We use a learning rate of 0.0001 for the Oracle model and the Guesser model, and a learning rate of 0.001 for QGen. All the models are trained with at most 30 epochs and early stopped within five epochs without improvement on the validation set. We report the performance on the test set which consists of images not used in training. We report the game success rate as the performance metric for all three models, which equals to the number of succeeded games divided by the total number of all games. Compared to previous works [de Vries et al., 2017; Strub et al., 2017; Strub and de Vries, 2017], after supervised training, our models obtain a game success rate of 48.77%, that is 4% higher than state-of-the-art methods [Strub and de Vries, 2017], which has 44.6% accuracy.

5.2 Reinforcement Learning

We first initialize all models with pre-trained parameters from supervised learning and then post-train the QGen using either REINFORCE or TPG for 80 epochs. We update the parameters using stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64. In each epoch, we sample each image in the training set once and randomly pick one of the objects as a target. We track the running average of the game success rate and use it directly as the baseline, \( b \), in REINFORCE. We limit the maximum number of questions to 8 and the maximum number of words to 12. Simultaneously, we train the Guesser model using REINFORCE without baseline and using SGD with a learning rate of 0.0001. The performance comparison between our baseline (\( #3 \)) with methods from literature (\( #1 \) & \( #2 \)) is shown in Tab. 1.

**REINFORCE Baseline:** From Tab. 1, we see that our models trained with REINFORCE (\( #3 \)) are about 9% better than the state-of-the-art methods (\( #1 \) & \( #2 \)). The improvements are due to using advanced mechanisms and techniques such as the Seq2Seq structure in the QGen, the memory and attention mechanisms in the Guesser, and the training of the Guesser model with reinforcement learning. One important difference is that our QGen model is trained in question level. This means that the model first learns to query meaningfully, step by step. Eventually, it learns to conduct a meaningful dialog. Compared to directly learning to manage a strategic conversation, this bottom-up training procedure helps the model absorb knowledge, because it breaks large tasks down into smaller, more manageable pieces. This makes the learn-
Table 2: Some samples generated by our improved models using REINFORCE (left column: “Policy Gradient”) and Dynamic-TPG (right column: “Tempered Policy Gradient”). The green bounding boxes highlight the target objects; the red boxes highlight the wrong guesses.

<table>
<thead>
<tr>
<th>Image</th>
<th>Policy Gradient</th>
<th>Tempered Policy Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is it in left?</td>
<td>No</td>
<td>Is it a giraffe?</td>
</tr>
<tr>
<td>Is it in front?</td>
<td>Yes</td>
<td>Is it a person?</td>
</tr>
<tr>
<td>Is it in right?</td>
<td>No</td>
<td>Is it in front of photo?</td>
</tr>
<tr>
<td>Is it in middle?</td>
<td>Yes</td>
<td>Is it a truck?</td>
</tr>
<tr>
<td>Is it person?</td>
<td>No</td>
<td>In the left half?</td>
</tr>
<tr>
<td>Is it ball?</td>
<td>No</td>
<td>Is it in the middle of photo?</td>
</tr>
<tr>
<td>Is it bat?</td>
<td>No</td>
<td>Is it to the right photo?</td>
</tr>
<tr>
<td>Is it car?</td>
<td>Yes</td>
<td>Is it in the middle of photo?</td>
</tr>
<tr>
<td>Status:</td>
<td>Failure</td>
<td>Status:</td>
</tr>
</tbody>
</table>

**TPG Dialogue Samples:** The generated dialogue samples in Tab. 2 can give some interesting insights in explaining why TPG methods give a better result. First of all, the sentences generated from TPG-trained models are on average longer and use slightly more complex structures, such as “Is it in the middle of photo?” instead of a simple form “Is it in middle?”. Secondly, TPGs enable the models to explore better and comprehend more words. For example, in the first task (upper half of Tab. 2), both models ask about the category. The REINFORCE-trained model can only ask with the single word “car” to query about the vehicle category. In contrast, the TPG-trained model can first ask a more general category with the word “vehicle” and follows up querying with a more specific category “trucks”. These two words “vehicle” and “trucks” give much more information than the single word “car”, and help the Guesser model identify the truck among many cars. Lastly, similar to the category case, the models trained with TPG can first ask a larger spatial range of the object and follow up with a smaller range. In the second task (lower half of Tab. 2), we see that the TPG-trained model first asks “In the left half?”, which refers to all the three giraffes in the left half, and the answer is “Yes”. Then it asks “Is it to the left of photo?”, which refers to the second left giraffe, and the answer is “Yes”. Eventually the QGen asks “In the left in photo?”, which refers to the most left giraffe, and the answer is “No”. These specific questions about locations are not learned using REINFORCE. The REINFORCE-trained model can only ask a similar question with the word “left”. In this task, there are many giraffes in the left part of the image. The top-down spatial questions help the Guesser model find the correct giraffe. To summarize, the TPG-trained models use longer and more informative sentences than the REINFORCE-trained models.

### 6 Conclusion

Our paper makes two contributions. Firstly, by extending existing models with Seq2Seq and Memory Networks we could improve the performance of a goal-oriented dialogue system by 9%. Secondly, we introduced TPG, a novel class of temperature-based policy gradient approaches. TPGs boosted the performance of the goal-oriented dialogue systems by another 4.7%. Among the three TPGs, Dynamic-TPG gave the best performance, which helped the agent comprehend more words, and produce more meaningful questions. TPG is a generic strategy to encourage word exploration on top of policy gradients and can be applied to any text-generating agents.
References


