Incorporating Unstructured Textual Knowledge Sources into Neural Dialogue Systems

Ryan Lowe, Nissan Pow, Laurent Charlin, Joelle Pineau
School of Computer Science, McGill University
{ryan.lowe, nissan.pow}@mail.mcgill.ca
lcharlin@gmail.com, jpineau@cs.mcgill.ca

Iulian V. Serban
MILA, Université de Montréal
julian.serban@gmail.com

Abstract

We present initial methods for incorporating unstructured external textual information into neural dialogue systems for predicting the next utterance of a user in a two-party chat conversation. The main objective is to leverage additional information about the topic of the conversation to improve the prediction accuracy. We propose a simple method for extracting this knowledge, using a combination of hashing and TF-IDF, and a way to use it for selecting the best next utterance of a conversation, by encoding a vector representation with a recurrent neural network (RNN). This is combined with an RNN encoding of the context and response of the conversation in order to make a prediction. We perform a case study using the recently released Ubuntu Dialogue Corpus, where the additional knowledge considered consists of the Ubuntu manpages. Preliminary results suggest that leveraging external knowledge sources in such a manner could lead to performance improvements for predicting the next utterance.

1 Introduction

Several recent results have shown that neural networks provide powerful models for various modules of spoken dialogue systems, from speech recognition [4, 5], to state tracking [7] and natural language generation [19]. Deep learning approaches that are trainable end-to-end are beginning to surface in the dialogue community [16, 15], which essentially combine parsing, dialogue planning and natural language generation modules. Although initial results are promising, they still require significant additions before replacing more conventional *pipeline* dialogue systems [8].

Simultaneously, much progress has been made towards incorporating a form of external memory into various neural network architectures for sequence modeling. Models such as Memory Networks [20, 17] and Neural Turing Machines [6] store some part of their input in a memory, which is then reasoned over in order to perform a variety of sequence to sequence tasks. These primarily use soft attention mechanisms [1], which compute a weighted sum over each entry of the memory. While this allows complex memory addressing as a function of the inputs, it is prohibitively expensive for large memories, since the computational complexity is linear in the number of memory addresses.

We are interested in incorporating large unstructured sources of external knowledge into dialogue systems. Compared to the above work on learning memories, we explore using visible (that is preloaded and read-only) memories. Using external information is of great importance to dialogue systems, particularly in the task-oriented setting (for example, helping users find movies or catch their bus using corresponding schedules [14, 10]). Even chat-oriented dialogue systems whose purpose is to entertain the user could benefit from leveraging external information, such as current news articles or movie reviews, in order to better converse about real-world events. To our knowledge, incorporating external information into a dialogue system has only been done if the knowledge base

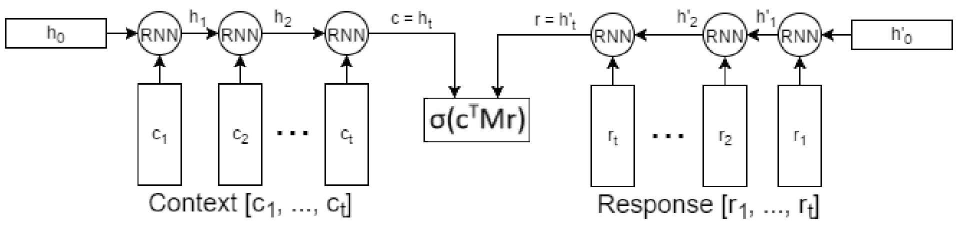

Figure 1: Diagram of the dual-encoder model.

is structured, and the dialogue system has been programmed to extract specific information from that structure (e.g. using a fixed set of database queries). We review relevant work in Section 6.

In this paper, we extend a recently proposed neural architecture for selecting dialogue responses from a list of candidate responses by incorporating unstructured sources of textual knowledge. The architecture uses a simple combination of hashing and TF-IDF to quickly identify the most relevant portions of text from the external knowledge source, based on the current context of the dialogue. Three recurrent neural networks (RNNs) are trained: one to encode the selected external information, one to encode the context of the conversation, and one to encode a response to the context. Outputs of these modules are combined to produce the probability that the response was the actual next utterance given the context. We apply this framework to the Ubuntu Dialogue Corpus [9] and look at the impact of incorporating the Ubuntu manpages as an external knowledge source, to improve performance on the task of predicting the next utterance of the conversation.

2 Technical Background

2.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a variant of neural networks that allow for time-delayed directed cycles between units. RNNs are often described by 'un-rolling' these cycles and instantiating a time-dependent internal, or hidden, state of the network h_t. This state is updated at each time step as a function of both the observed variables x_t, and the hidden state from the previous time step h_{t-1}. Input-to-hidden (W_{ih}) and hidden-to-hidden (W_{hh}) parameter matrices are used to transform respectively the input and the previous hidden state: $h_t = f(W_{hh}h_{t-1} + W_{ih}x_t)$, where $f(\cdot)$ is a (typically) non-linear function.

RNNs can learn to model probability distributions over input sequences, by being trained to predict the next symbol in the sequence. They have proven to be very useful for modeling a variety of sequential data, and have in particular become the primary building block of many recent neural language models [3, 16], which use RNNs as encoders and decoders.

2.2 The Dual-Encoder Model

Here we adopt the dual-encoder model used by Lowe et al. [9], illustrated in Figure 1. The model scores potential context-response pairs to determine the probability that a certain response was the actual next utterance, given the context of a conversation. The model consists of two RNNs with tied weights, which are used to produce the embeddings for the context and response, $c, r \in \mathbb{R}^d$, respectively. This is done by feeding in the word embeddings, one at a time, into the respective RNNs. At each step, the hidden state of the RNN is updated until the final input symbol. Intuitively, the final hidden state can be thought of as a *summary* of the input sequence.

Using the final hidden states from both RNNs, the model calculates the probability that the response r is the correct response given the context c (ie. $y = 1$):

$$p(\mathrm{y} = 1 \mid c, r) = \sigma(c^\top M r + b),$$

where the bias b and the matrix $M \in \mathbb{R}^{d\times d}$ are parameters learned by the model. The first term within the function can be thought of as a projection of the response into the context subspace via the product $c' = Mr$, and then taking the dot product between this 'generated context' c'

and the actual context c. The result is converted to a probability with the sigmoid function. The model is trained by minimizing the cross entropy of all labeled (context, response) pairs [21]: $\mathcal{L} = -\sum_n \log p(y_n|c_n, r_n, M)$. Regularization can also be applied on the parameters of the model.

3 Model Architecture

3.1 Knowledge Retrieval from Unstructured Documents

Given the context of a conversation, we aim to leverage external knowledge. Our goal is to use information that is contained in large unstructured textual knowledge sources, which are separate from the context of the conversation. Since we assume that the knowledge can be large, it is infeasible to use mechanisms that require intensive computations on each knowledge item, such as the recent soft-attention mechanisms [1]. Although there are a large number of possible methods that could be devised, we propose a simple method that can be applied in several domains. We encourage exploration of more complex memory addressing mechanisms.

We currently make use of the term frequency-inverse document frequency (TF-IDF) criteria [13] for identifying which knowledge source to use. TF-IDF is standard in information retrieval, and can be computed as:

$$\text{tfidf}(w, d, D) = f(w, d) \times \log \frac{N}{|\{d \in D : w \in d\}|},$$

where $f(w, d)$ indicates the number of times word w appeared in document d, N is the total number of documents, and the denominator represents the number of documents in which the word w appears. In our model, we calculate the TF-IDF score by comparing the term frequency in the current conversation with its frequency in a document in our knowledge source. This requires assuming that the knowledge source is divided into a set of documents (e.g. Wikipages, or paragraphs of a technical manual.) We use TF-IDF to compute a score between the dialogue context and each document, and choose the document, or a set of documents, with the highest score.

While TF-IDF gives us a way to select documents, it may be infeasible to calculate the TF-IDF score over all documents at each turn of a conversation. Thus we incorporate hashing to facilitate quick retrieval of plausible documents. We construct two hashtables: The first, which we refer to as the 'entity hashtable', contains as keys important entities that could appear in the conversation, with values being information corresponding to those entities. This could include, for example, actor or movie names with their IMDB summaries, technical commands with their documentation, or landmarks and restaurants with related Wikipedia pages or travel reviews. For any of these important entities that appear in the context, the extracted knowledge is the concatenation of the values of the entries in the entity hashtable.

If no match between the context and the entity hashtable is found, we search a second hashtable, the 'relation hashtable'. The keys here correspond to more common words that are related to the important entities, which is done by extracting a subset of salient words from the descriptions of the entities. Importantly, the *values* of the relation hashtable are *keys* (or lists of keys) from the entity hashtable, e.g. the key 'Chinese' could have as value a list of Chinese restaurants, so long as the description of these restaurants mentioned that they were Chinese. The list of extracted values from the relation hashtable are treated as if they were present in the context of the conversation, and these are used to search the entity hashtable. In this way, we can quickly extract knowledge for most conversation contexts.

3.2 The Knowledge Encoder Model

We propose an extension of the dual-encoder model (Sec. 2.2) that can incorporate the information extracted from the knowledge source as presented above to improve ability to select the next utterance in a conversation. The model must compute the similarity between the response, as encoded by an RNN, and a representation of the extracted knowledge, m, as encoded using another RNN with untied weights. The objective function becomes: $p(\text{y} = 1 \mid c, r) = \sigma(c^\top Mr + m^\top Nr + b)$, where $N \in \mathbb{R}^{d \times d}$ is another matrix of parameters learned by the model.

Similarly to the dual-encoder model, we can intuitively think of projecting the response into the corresponding context and knowledge subspaces, with $c' = Mr$ and $m' = Nr$. For a response to

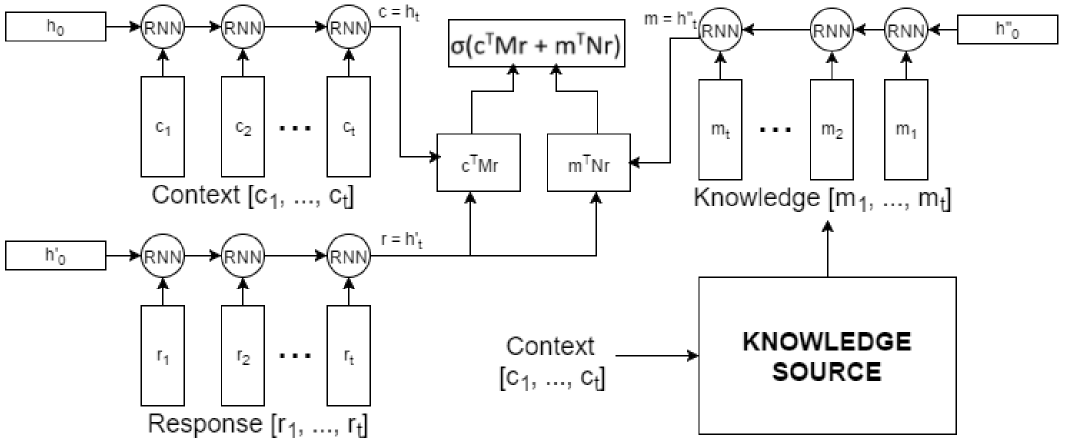

Figure 2: Diagram of the knowledge encoder model. c, r, m are the last hidden states from the RNNs. c_i, r_i, m_i are word vectors for the context, response and retrieved knowledge $i < t$. We consider contexts up to a maximum of $t = 160$.

have a high probability of being the next utterance given some context, it has to either 1) logically follow from the context (high $c^\top c'$), or 2) have information in common with the extracted knowledge (high $m^\top m'$). A diagram representing the Knowledge Encoder model is shown in Figure 2.

4 Case Study: The Ubuntu Dialogue System

4.1 The Ubuntu Dialogue Corpus

We consider the task domain associated with the Ubuntu Dialogue Corpus [9]. This dataset, extracted from the Ubuntu IRC chat logs, consists of a chat stream where users ask and answer technical questions about Ubuntu. Although the channel is multi-party, a sequence of rules are used to disentangle it into two-person dialogue (see [9] for more info). This results in a large corpus of multi-turn technical dialogues, which could be useful for training dialogue systems for technical support channels.

We focus on the task of 'next utterance classification' (NUC) [9]. Given the context of a conversation, a set of candidate answers is supplied: the 'correct' answer is the actual next utterance of the conversation, while the 'false' answers are randomly drawn from elsewhere in the corpus. This is more general than traditional question answering (QA), as the prediction is made based on the entire conversation context, and the context does not have to include a question at all. Intuitively, models that can easily select the next utterance given a context require a certain understanding of both the context and response, as simple keyword matching gives only baseline performance. In practice, these models can be used as re-ranking mechanisms for stochastic generative models [11, 18, 19], or they can be used as a generative model themselves by extending the set of responses to encompass the training set.

4.2 The Ubuntu Manpages

The Ubuntu Manpages refer to a collection of Unix and Linux manual pages, extracted from every package of every supported version of Ubuntu. Manpages are self-contained reference documents (packaged with the distribution) that provide information on various Ubuntu commands, toolkits, and extensions. These manpages contain useful information for aiding with technical Ubuntu problems. We hypothesize that manpages can be successfully used as a source of external knowledge in the NUC task.

The format of the manuals is a series of entries, each with the name of the command along with a description of its purpose. The amount of information available per entry varies significantly, as the subject of the entry can be both very specific (a niche command such as `411toppm`), or can

Input Context	Retrieved Knowledge	Candidate Responses
ok but back to the point, I guess I could download a theme for the decorator only... __EOS__ if you get a metacity theme, thats what it is. there's nothing else *to* theme, its just that since metacity doesnt have multiple decorators, theres no need to give a more specific name to the themes than metacity	this manual page documents briefly metacity . metacity is a minimal x window manager aimed at nontechnical users and is designed to integrate well with the gnome desktop . metacity lacks some features that may be expected by traditional unix or other technical users; these users may want to investigate other available window managers for use with gnome or standalone	1) *metacity theme includes info about the border+title and the rest, I mean get a theme only for the first part, then go to some metacity theme in gnome-appearance-properties, then change the title look in one of its options* 2) you will need to reinstall the nvidia driver
Newbie question : are there any good graphical tools for handling fat32 disk mounting in linux ?__EOS__ no __EOS__ ok ;) __EOS__ but you can tell ubuntu to mount fat32 at start up so you dont have to think about it	baobab is able to scan either specific folders (local and remote) or devices , in order to give the user a tree representation including each directory size or percentage in the branch . it also auto-detects any mounted/unmounted device . a graphical representation is also provided for any selected folder .	1) *yes, I know - but it sure would be nice with a gui and some smooth icons rather than poking around with fstab* 2) why would he need an ssh server on windows ?

Table 1: Example of the knowledge retrieval method for various contexts in the Ubuntu Dialogue Corpus, along with two corresponding responses. "__EOS__" indicates the end of an utterance in the context. Italic response is the actual next utterance given the context.

encompass many sub-commands (such as BusyBox). A common entry, in addition to the name and description, will contain an 'Options' section with various sub-commands that can be used, along with the author of the manpage and a 'See Also' section with other related commands. Some entries will also have diagnostic information, example usages, and links to further web documentation. In total for all of the downloaded manpages, there are approximately 5,000 entries, with an average of approximately 190 lines and 10 paragraphs.

For knowledge extraction, we treat the name of the technical commands as the keys to the entity hashtable, and their descriptions as the corresponding values. Each paragraph of the description is considered a document for the purposes of TF-IDF. The words from the short 'NAME' section (which has a one-sentence summary of the command) are used as keys to the relation hashtable. Lists are restricted to be of 5 or fewer commands, and we filter the keys of the hashtable by removing common English words. Since we limit the number of retrieved paragraphs to a maximum of 20, we also search the relation hashtable in order of length of the key word, as longer word are more likely to carry technical information.

We show one useful and one non-useful example of external knowledge use in Table 1, for examples from the Ubuntu Dialogue Corpus. In the first example, the key word 'metacity' is identified, and a description of the term is retrieved. Using this description gives a higher probability of selecting the correct response (italics), as both contain references to 'gnome' which do not appear in the context. In the second example, no command is found that matches the keys in the entity dictionary; the key word 'graphical' is used to find the command 'Baobab', which is a graphical toolkit. However, in this case the users are not talking about Baobab, so the retrieved knowledge is not used.

Implementing document retrieval with hashing was critical in order to be able to train the model in a reasonable amount of time. Speedups of approximately 200 times were observed, compared to naïvely searching through the entire document with TF-IDF (from 4s to 0.02s per query).

5 Results

We present results in Table 2 on the next utterance classification task, using both the dual-encoder (DE) model, the knowledge encoder (KE) model[1], and a baseline model that directly computes the TF-IDF between the context and each response. KE uses shared weights for all RNNs, while KE-untied uses separate weights for encoding the external knowledge. Further, we pretrain KE-untied as the DE model (ie. $N = 0$), and then fine-tune the weights. In other words, the model first learns to predict the next utterance only given the context, and then learns to use the external memory for cases it cannot otherwise solve.

We test on Recall@k metrics (denoted R@1 R@2, R@5). Here the agent is asked to select the k most likely responses, and it is correct if the true response is among these k candidates. Our prelim-

[1]DE results differ slightly from the results in [9], as a different random initialization was used (results are averaged between the two).

Model	1 of 2 R@1	1 of 10 R@1	1 of 10 R@2	1 of 10 R@5
TF-IDF Baseline [9]	65.9%	41.0%	54.5%	70.8%
DE [9]	75.3%	38.7%	52.3%	79.8%
KE [Section 3.2]	73.2%	33.8%	48.7%	77.3%
KE-untied [Section 3.2]	**77.4%**	**41.3%**	**55.4%**	**82.4%**

Table 2: Results of the model on the Ubuntu Dialogue Corpus, using various recall measures for binary (1 in 2) and 1 out of 10 (1 in 10) next utterance classification %.

inary results seem to indicate that using external knowledge can provide an increase of 2-3% over the DE model. However, this is only slightly higher than the TF-IDF method; large improvements are mostly seen with LSTM networks [9], which were not tested due to time constraints.

We use the same setup and hyperparameters as in [9]. For training, we have a 1:1 ratio between true responses (flag = 1), and negative responses (flag=0) drawn randomly from elsewhere in the training set. Word vectors are initialized using GloVe [12], and are fine-tuned during training. We train on 1,000,000 training examples, and use a validation set for early stopping. The optimal hyperparameters for the regular RNN model from [9] were used.

6 Related Work

Task-oriented dialogue systems whose goal is to provide information to the user naturally make use of external knowledge sources. For example, in the Let's Go! dialogue system [14], the user requests information about various bus arrival and departure times. Thus, a critical input to the model is the actual bus schedule, which is used in order to generate the system's utterances. Such examples are abundant both in the literature and in practice. Recent work incorporates neural networks in order to improve the performance of these models for state tracking [7, 22], speech recognition [4, 5], and natural language generation [19]. Although these models could be seen as neural dialogue systems that have some external knowledge component, these knowledge sources are highly structured and are only used to place hard constraints on the possible states of an utterance to be generated. Furthermore, these structured external knowledge components require expensive maintenance, and there is no guarantee that they contain all relevant information. With our model, any useful source of textual knowledge can be incorporated in order to aid the prediction of the next utterance.

Our work is also related to question-answering systems, which extract answers from external knowledge sources. However, these systems are generally unable to take into account dialogue context when answering questions. It is beyond the scope of this paper to review these systems, but the reader may find the recent neural network architectures for question-answering interesting [2, 21].

7 Discussion

Using external knowledge is critical for goal-oriented dialogue systems, and opens up many possibilities for chat-oriented systems. We propose a simple method for both retrieving relevant information from an unstructured textual knowledge source, and a way to incorporate this into a neural dialogue system for predicting the next utterance of a conversation. Our results, while preliminary, suggest that using external knowledge may be useful.

Significantly more complex models could be devised for both knowledge retrieval, and incorporation into the prediction of the next utterance. For example, in the case where documents of the knowledge source have overlapping information, information retrieval methods could be used (such as clustering) for knowledge extraction, as well as methods from modern question answering systems. Many reasonable variants of the objective function may be worth considering. Extracted information could be used to condition a decoder RNN, leading to a fully generative model.

For future work, we plan to determine whether the performance improvements hold when implementing the model with LSTM units. We will then investigate a more complex attention mechanism for large input spaces, using distributed representations of the external knowledge, and implement a dialogue model that uses this mechanism. This paper represents an initial attempt at incorporating unstructured textual knowledge; we encourage further exploration in this area.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[2] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.

[4] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

[5] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.

[6] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[7] Matthew Henderson, Blaise Thomson, and Steve Young. Deep neural network approach for the dialog state tracking challenge. In *SIGDIAL*, pages 467–471, 2013.

[8] Esther Levin and Roberto Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Eurospeech*, volume 97, pages 1883–1886, 1997.

[9] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGDIAL*, 2015.

[10] Elmar Nöth, Axel Horndasch, Florian Gallwitz, and Jürgen Haas. Experiences with commercial telephone-based dialogue systems. *it–Information Technology (vormals it+ ti)*, 46(6/2004):315–321, 2004.

[11] Alice H Oh and Alexander I Rudnicky. Stochastic language generation for spoken dialogue systems. In *ANLP/NAACL Workshop on Conversational Systems*, pages 27–32, 2000.

[12] J. Pennington, R. Socher, and C.D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, 2014.

[13] J. Ramos. Using tf-idf to determine word relevance in document queries. In *ICML*, 2003.

[14] Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. LetâĂŹs go public! taking a spoken dialog system to the real world. In *INTERSPEECH*. Citeseer, 2005.

[15] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*, 2015.

[16] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. *NAACL*, 2015.

[17] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015.

[18] Tsung-Hsien Wen, Milica Gašic, Dongho Kim, Nikola Mrkšic, Pei-Hao Su, David Vandyke, and Steve Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *EMNLP*, 2015.

[19] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *SIGDIAL*, 2015.

[20] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

[21] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *NIPS Deep Learning Workshop*, 2014.

[22] Lukas Zilka and Filip Jurcicek. Incremental lstm-based dialog state tracker. *arXiv preprint arXiv:1507.03471*, 2015.