# LSTM-based Mixture-of-Experts for Knowledge-Aware Dialogues

**Phong Le***
University of Amsterdam
p.le@uva.nl

**Marc Dymetman, Jean-Michel Renders**
Xerox Research Centre Europe
{firstname.lastname}@xrce.xerox.com

## Abstract

We introduce an LSTM-based method for dynamically integrating several word-prediction experts to obtain a conditional language model which can be good simultaneously at several subtasks. We illustrate this general approach with an application to dialogue where we integrate a neural chat model, good at conversational aspects, with a neural question-answering model, good at retrieving precise information from a knowledge-base, and show how the integration combines the strengths of the independent components. We hope that this focused contribution will attract attention on the benefits of using such mixtures of experts in NLP and dialogue systems specifically.

## 1 Introduction

The mainstream architecture for virtual agents in dialogue systems (McTear, 2004; Jokinen and McTear, 2009; Rieser and Lemon, 2011; Young et al., 2013) involves a combination of several components, which require a lot of expertise in the different technologies, considerable development and implementation effort to adapt each component to a new domain, and are only partially trainable (if at all). Recently, Vinyals and Le (2015), Serban et al. (2015), Shang et al. (2015) proposed to replace this complex architecture by a single network (such as a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997)) that predicts the agent's response from the dialogue history up to the point where it should be produced: this network can be seen as a form of conditional neural language model (LM), where the dialogue history provides the context for the production of the next agent's utterance.

Despite several advantages over the traditional architecture (learnability, adaptability, better approximations to human utterances), this approach is inferior in one dimension: it assumes that all the knowledge required for the next agent's utterance has to be implicitly present in the dialogues over which the network is trained, and to then be precisely memorized by the network, while the traditional approach allows this knowledge to be dynamically accessed from external knowledge-base (KB) sources, with guaranteed accuracy.

To address this issue, we propose the following approach. As in Vinyals and Le (2015), we first do train a conditional neural LM based on existing dialogues, which we call our *chat model*; this model can be seen as an "expert" about the conversational patterns in the dialogue, but not about its knowledge-intensive aspects. Besides, we train another model, which this time is an expert about these knowledge aspects, which we call our *QA model*, due to its connections to Question Answering (QA). We then combine these two expert models through an LSTM-based *integration model*, which at each time step, encodes the whole history into a vector and then uses a softmax layer to compute a probability mixture over the two models, from which the next token is then sampled.

While here we combine in this way only two models, this core contribution of our paper is immediately generalizable to several expert models, each competent on a specific task, where the (soft) choice between the models is done through the same kind of contextually-aware "attention" mechanism. Additional smaller contributions consist in the neural regime we adopt for training the QA model, the way in which we reduce the memorization requirements on this model.

It is worth noting that concurrently with our

*Work performed during Phong Le's internship at XRCE in 2015.



*Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 94–99,
Berlin, Germany, August 11th, 2016. 

work, Yin et al. (2015) have proposed a similar idea focusing only on QA in a traditional set-up. Our case is more difficult because of the chat interaction; and the integration framework we propose is generally applicable to situations where a pool of word-prediction "experts" compete for attention during the generation of text. Outside of dialogue applications, also independently and even more recently, Ling et al. (2016) have proposed a "Latent Predictor Network for Code Generation", which has some close similarities to our LSTM-based mixture of experts.

## 2 LSTM-based Mixture of Experts

The method is illustrated in Figure 1. Let $w_1^t = w_1 ... w_t$ be a history over words. We suppose that we have $K$ models each of which can compute a distribution over its own vocabulary $V_k$ : $p_k(w \in V_k | w_1^t)$, for $k \in [1, K]$. We use an LSTM to encode the history word-by-word into a vector $\mathbf{h}_t$ which is the hidden state of the LSTM at time step $t$. We then use a softmax layer to compute the probabilities

$$p(k|w_1^t) = \frac{e^{u(k,\mathbf{h}_t)}}{\sum_{k'=1}^{K} e^{u(k',\mathbf{h}_t)}}$$

where $[u(1, \mathbf{h}_t), ..., u(K, \mathbf{h}_t)]^T = \mathbf{W}\mathbf{h}_t + \mathbf{b}$, $\mathbf{W} \in \mathbb{R}^{K \times dim(\mathbf{h}_t)}$, $\mathbf{b} \in \mathbb{R}^K$. The final probability of the next word is then:

$$p(w|w_1^t) = \sum_{k=1}^{K} p(k|w_1^t)\, p_k(w|w_1^t). \qquad (1)$$

Our proposal can be seen as bringing together two previous lines of research within an LSTM framework. Similar to the *mixture-of-experts* technique of Jacobs et al. (1991), we predict a label by using a "gating" neural network to mix the predictions of different experts based on the current situation. Similar to the approach of Florian and Yarowsky (1999), we dynamically combine distributions on words to produce an integrated LM. However Florian and Yarowsky (1999) focus on the combination of topic-dependent LMs, while in our case, the components can be arbitrary distributions over words — we later use a component that produces answers to questions appearing in the text. In our case, the labels are words, the gating neural network is an LSTM that stores a representation of a long textual prefix, and the combination mechanism is trained by optimizing the parameters of this LSTM.

## 3 Data

Our corpus consists of 165k dialogues from a "tech company" in the domain of mobile telephony support. We split them into train, development, and test sets whose sizes are 145k, 10k, and 10k dialogues. We then tokenize and lowercase each dialogue, and remove unused information such as head, tail, chat time (Figure 2). For each response utterance found in a dialogue, we create a context-response pair whose context consists of all sentences appearing before the response. This process gives us 973k/74k/75k pairs for training/development/testing.

**Knowledge-base** The KB we use in this work consists of 1,745k device-attribute-value triples, e.g., (Apple iPhone 5; camera megapixels; 8.0). There are 4729 devices and 608 attributes. Because we consider only numeric values, only triples that have numeric attributes are chosen, resulting in a set of 65k triples of 34 attributes.

**Device-specification context-response pairs** Our target context-response pairs are those in which the client asks about numeric value attributes. We employ a simple heuristic to select target context-response pairs: a context-response pair is chosen if its response contains a number and one of the following keywords: cpu, processor, ghz, mhz, memory, mb(s), gb(s), byte, pixel, height, width, weigh, size, camera, mp, hour(s), mah. Using this heuristic, we collect 17.6k/1.3k/1.4k pairs for training/dev/testing. These sets are significantly smaller than those extracted above.

## 4 KB-aware Chat Model

### 4.1 Neural Chat Model

Ouur corpus is comparable to the one described in Vinyals and Le (2015)'s first experiment, and we use here a similar neural chat model.

Without going into the details of this model for lack of space, it uses a LSTM to encode into a vector the sequence of words observed in a dialogue up to a certain point, and then this vector is used by another LSTM for generating the next utterance also word-by-word. The approach is reminiscent of seq2seq models for machine translation such as (Sutskever et al., 2014), where the role of "source

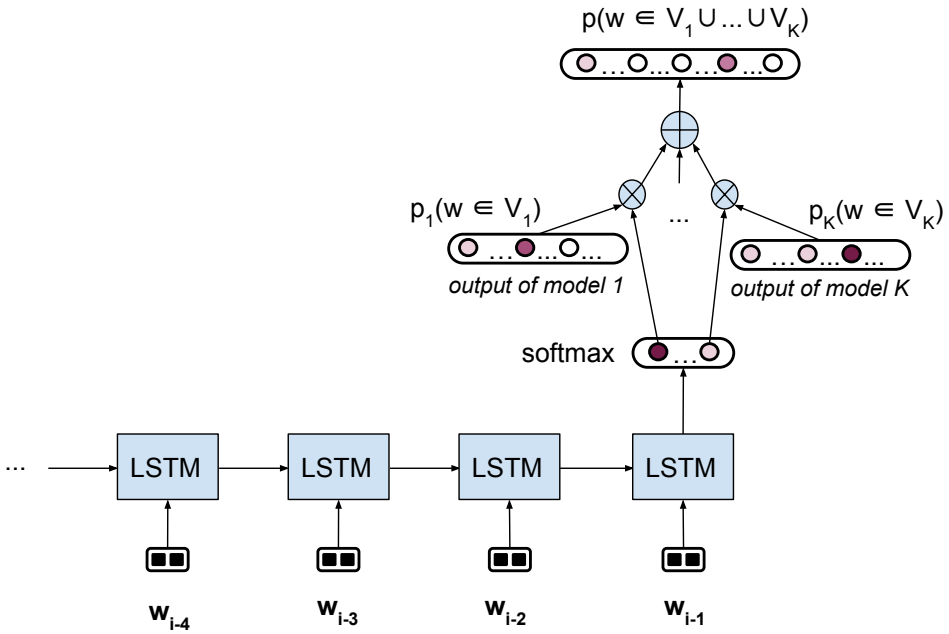


Figure 1: LSTM-based mixture-of-experts for Language modelling. $\otimes$ denotes multiplication, $\oplus$ denotes sum.

```
7760 | 121686798 | log started fri may 06 10:50:43 pdt 2011
-lsb- 10:51:33 -rsb- you have been connected to X .

-lsb- 10:51:49 -rsb- X : hello and thanks for contacting Z ! my name is X , how can i assist
                       you today ?
-lsb- 10:52:13 -rsb- Y : how do i change the text notification on my htc evo
-lsb- 10:53:06 -rsb- X : sorry you are having problems with that but you are in the right
                       place . before we begin can i start with you name please ?
-lsb- 10:53:28 -rsb- Y : Y test
-lsb- 10:53:55 -rsb- X : thank you Y . one moment while i pull up the information on that
                       device .
-lsb- 10:54:36 -rsb- Y : i am using this to showcase the shack support to an employee if you
                       guys are busy we can try this later

-lsb- -- end of transcript as seen by customer -- -rsb-
-lsb- 10:55:10 -rsb- the customer has ended the chat session .
...
```

head

time

tail

Figure 2: An example dialogue.

sentence" is played by the dialogue prefix, and that of "target sentence" by the response utterance.

### 4.2 Neural Question Answering Model

In a standard setting, a question to query a KB must be formal (e.g., SQL). However, because a human-like QA system should take natural questions as input, we build a neural model to translate natural questions to formal queries. This model employs an LSTM to encode a natural question into a vector. It then uses two softmax layers to predict the device name and the attribute. This model is adequate here, since we focus on the QA situation where the client asks about device specifications. For more complex cases, more advanced QA models should be considered (e.g., Bordes et al. (2014), Yih et al. (2015)).

Given question $w_1^l$, the two softmax layers give us a distribution over devices $p_d(\bullet|w_1^l)$ and a distribution over attributes $p_a(\bullet|w_1^l)$. We can then compute a distribution over the set $V_{qa}$ of all values found in the KB, by marginalizing over $d, a$:

$$p_{qa}(v|w_1^l) = \sum_{\langle d,a,v\rangle \in T} p_d(d|w_1^l) p_a(a|w_1^l), \quad (2)$$

where $T$ is the set of all triples in the KB.

Initial experiments showed that predicting values in this indirect way significantly improves the accuracy compared to employing a single softmax layer to predict values directly, because it does not require the hidden states to directly memorize the value for each device-attribute pair.

**Data Generation** One serious difficulty is that we do not have a corpus of natural questions on which to train the QA model, so we have to resort to a method for generating virtual question/answer pairs, on which to train our QA model. However, existing corpora and methods for generating such data (e.g., Fader et al. (2013)) hardly meet our needs here. This is because our case is very different from (and somewhat more difficult than) traditional QA set-ups in which questions are independent. In our case several scenarios are possible, resulting from the chat interaction (e.g., in a chat, questions can be related as in Figure 3). We therefore propose a simple heuristic method for generating artificial QA data that can cover several scenarios.

For each pair <device name, attribute>, we paraphrase the device name by randomly dropping some words (e.g., "apple iphone 4" becomes "iphone 4"), and paraphrase the attribute using a small handcrafted dictionary and also randomly dropping some words ("battery talk time" becomes "battery life" which can become "battery"). We then draw a sequence of $l$ words from a vocabulary w.r.t word frequency, where $l \sim Gamma(k, n)$ (e.g., "i what have"), and shuffle these words. The output of the final step is used as a training datapoint like: `have` *`iphone 4`* `what` *`battery`* `i` $\rightarrow$ `apple_iphone_4 battery_talk_time`. To make it more realistic, we also generate complex questions by concatenating two simple ones. Such questions are used to cover the dialogue scenario where the client continues asking about another device and attribute. In this case, the system should focus on the latest device and attribute. Using this method, we generate a training set of 7.6m datapoints and a development set of 10k.

### 4.3 Integration

We now show how we integrate the chat model with the QA model using the LSTM-based mixture-of-experts method. The intuition is the following: the chat model is in charge of generating smooth responses into which the QA model "inserts" values retrieved from the KB. Ideally, we should employ an independent LSTM for the purpose of computing mixture weights, as in Section 2. However, due to the lack of training data, our integration model makes use of the chat model's hidden state to compute these weights. Because this hidden state captures the uncertainty of generating the next word, it is also able to detect whether or not the next word should be generated by the chat model.

The chat model is the backbone because it generates most tokens. The QA model, on the other hand, is crucial since we want the system to generate correct values. (E.g., the chat model alone cannot provide the precise information shown in Figure 3.) More importantly, in the future when new devices are released, we do not need to collect new chat data, which are often expensive, to retrain the chat model.

Let $C$ and $w_1^t$ be a context and words generated up to this point. $p_c(\bullet|w_1^t, C)$ and $p_{qa}(\bullet|w_1^t, C)$ are given by the chat model and the QA model. We then compute the distribution $p(\bullet|w_1^t, C)$ over $V_c \cup$

$V_{qa}$ as a mixture of $p_c$ and $p_{qa}$:

$$p(w|w_1^t, C) = \alpha.p_c(w|w_1^t, C) + (1-\alpha).p_{qa}(w|w_1^t, C)$$

where $\alpha = \sigma(\mathbf{w}^T \mathbf{h}_t^c + b)$, $\mathbf{h}_t^c$ is the hidden state of the chat model, $\sigma$ is the sigmoid function; $\mathbf{w} \in \mathbb{R}^{dim(\mathbf{h}_t^c)}$ and $b \in \mathbb{R}$. Note that the sigmoid is equivalent to the softmax for two output units.

**Training** To train this integration model, we keep the chat model and the QA model frozen, and minimize the objective:

$$J(\theta) = -\sum_{(C,w_1^t)\in D} \sum_{t=0}^{l-1} \beta(w_{t+1}) \log p(w_{t+1}|w_1^t, C; \theta) + \frac{\lambda}{2}||\theta||^2$$

w.r.t. $\theta = (\mathbf{w}, b)$, where $\beta(w) = 100$ if $w \in V_{qa} \setminus V_c$, $\beta(w) = 1$ otherwise. $\lambda$ is the regularization parameter and $D$ is the training set. We set $\beta(w \in V_{qa} \setminus V_c)$ high because we want the training phase to focus on those tokens representing values in the KB but not supported by the chat model.

**Decoding** To find the most probable responses, our decoder employs the uniform-cost-search algorithm (Russell and Norvig, 2003), which is guaranteed to find optimal solutions and is feasible with our search space. We stipulate a constraint that a response is to answer not more than one question.

## 5 Experiments

We implement our models in C++ using CUDA. Since automatically evaluating a conversation system is still challenging, we, following Vinyals and Le (2015), use word perplexity only. In our experiments, every LSTM has 1024 hidden units and 1024 memory cells. The vocabulary of the chat model has 19.3k words, that of the QA model 12.7k words.

We firstly train the chat model on all chat data with the learning rate 0.01, and continue training it on the device-specification data with the learning rate 0.001. Using this smaller learning rate we expect that the model will not forget what it has learnt on all the chat corpus. Next, we train the QA model on the data generated in Section 4.2 with the learning rate 0.01. Finally, we train the integration model on the device-specification training data also with the learning rate 0.01.

Our initial results are as follows. The integration slightly increases the perplexity on all tokens (15.4, compared to 14.7 of the chat model), but it does help to significantly decrease perplexity 38% on the *numeric* tokens (46.8, compared to 75.8 of the chat model). This decrease is due to the improved ability of the integration model to detect places where a numeric value associated with a device-attribute information request should be inserted and to predict this value. Not all numeric values are associated with information requests of this type, but the reported perplexities are over all numeric values. The decrease in perplexity over the numeric values is not enough to decrease overall perplexity because the numeric tokens represent only 6.7% of the tokens, and the integration model wrongly puts some small probability mass on the QA expert also in the case of the many non-numeric tokens. However, the fact that the perplexity decreases over the numeric tokens shows that the integration model is useful for predicting correct values, which are informationally much more critical to the user than general words (admittedly, perplexity is here a weak proxy for what a human evaluation of usefulness would provide.)

Figure 3 shows a chat example with our integration model.

## 6 Conclusions

We introduce a general LSTM-based mixture-of-experts method for language modelling and illustrate the approach by integrating a neural chat model with a neural QA model. The experimental results, while limited to measures of perplexity, do show that the integration model is capable of handling chats inside of which the user may ask about device specifications; a more thorough and convincing evaluation would require human assesments of the quality of the produced responses.

We believe that the proposed integration method has potential for a wide range of applications. It allows to pool a number of different language models, each expert in a specific domain or class of problems (possibly trained independently based on the most appropriate data) and to generate the next word based on a competition between these models, under the supervision of an LSTM-based attention mechanism.

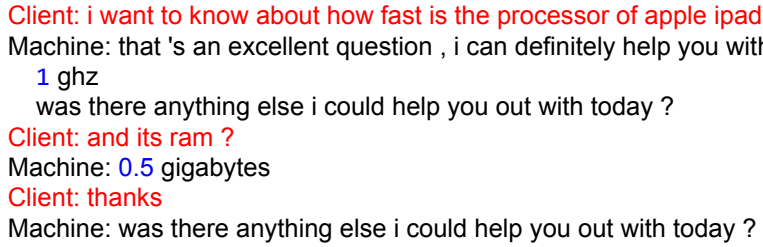


Figure 3: A dialogue with the integration model. The first value is produced by the pair <apple_ipad_2, cpu_maximum_frequency>, the second value by <apple_ipad_2, internal_ram>.

## References


Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar, October. Association for Computational Linguistics.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618. Citeseer.

Radu Florian and David Yarowsky. 1999. Dynamic nonlocal language modeling via hierarchical topic-based adaptation. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 167–174. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Robert A Jacobs, Michael I Jordan, and Andrew G Barto. 1991. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250.

Kristiina Jokinen and Michael F. McTear. 2009. *Spoken Dialogue Systems*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *CoRR*, abs/1603.06744.

Michael F. McTear. 2004. *Spoken Dialogue Technology: Towards the Conversational User Interface*. Springer.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement learning for adaptive dialogue systems : a data-driven methodology for dialogue management and natural language generation*. Theory and applications of natural language processing monographs. Springer, Heidelberg, New York.

Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

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

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2015. Neural generative question answering. *CoRR*, abs/1512.01337.

Steve Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.