CONTEXTUAL DOMAIN CLASSIFICATION IN SPOKEN LANGUAGE UNDERSTANDING SYSTEMS USING RECURRENT NEURAL NETWORK

Puyang Xu, Ruhi Sarikaya

Microsoft Corporation, Redmond WA 98052, USA

{puyangxu, ruhi.sarikaya}@microsoft.com

ABSTRACT

In a multi-domain, multi-turn spoken language understanding session, information from the history often greatly reduces the ambiguity of the current turn. In this paper, we apply the recurrent neural network (RNN) to exploit contextual information for query domain classification. The Jordan-type RNN directly sends the vector of output distribution to the next query turn as additional input features to the convolutional neural network (CNN). We evaluate our approach against SVM with and without contextual features. On our contextually labeled dataset, we observe a 1.4% absolute (8.3% relative) improvement in classification error rate over the non-contextual SVM, and 0.9% absolute (5.5% relative) improvement over the contextual SVM.

Index Terms— Recurrent neural network, contextual domain classification

1. INTRODUCTION

Spoken language understanding (SLU) applications are becoming increasingly important in our daily lives. Many portable devices such as smartphones have personal assistants that are built with SLU technologies. SLU typically involves determining the user intent and extracting relevant semantic slots from the natural language sentence. In a commonly used architecture, a sentence is first classified into one of the supported domains, after that domain dependent intent analysis and slot filling (i.e. entity extraction) are carried out.

In such pipelines, domain classification is the first step of the semantic analysis. While it is a standard classification task and seemingly less complex than other semantic analysis such as entity extraction, the errors made by a domain classifier are usually much more visible – they often lead to clearly wrong system responses such as invoking a wrong application because it routes the query to wrong intent and slot models for semantic analysis. The focus of this work is to improve the accuracy of domain classification by exploiting the session context.

We would like to thank Zhaleh Feizollahi for preparing the dataset, and Yangyang Shi for helpful discussions.

We are particularly interested in the multi-turn multi-domain classification problem in the context of a conversational session, in which the user issues a series of consecutive queries to the system. An example of a session consisting of 5 queries (T1 through T5) is shown below:

T1: How is the weather in Houston (weather domain)
T2: how about Orlando (weather domain)
T3: am I free tomorrow (calendar domain)
T4: find hours for disney world (places domain)
T5: get driving directions (places domain)

In this example, the user starts with two weather queries, then checks the calendar to make sure she has free time, and eventually asks the system to provide information about disney world, which is potentially her travel destination for tomorrow. In such a session, we often observe that the users tend to follow an intuitive and predictable thought process, resulting in exploitable patterns in the domain sequence. It can be particularly useful when the query is short and ambiguous such as T2 in the above example – ”how about Orlando” can have very different domain interpretations depending on the previous query turn. The contextual modeling of the query domains reduces the likelihood of abrupt switches between domains, leading to a more coherent interaction during an SLU session.

In [1], it was shown that contextual information, more specifically the intent label of the previous turn, can improve the intent classification of the current turn using the standard support vector machine (SVM) classifier. This work shares the same objective – exploiting information from previous turns in a query session to improve the classification of the current turn. However, we focus on domain classification, and instead of using SVM with simple handcrafted features, we use the recurrent neural network (RNN).

The RNN is a special class of neural network (NN) with feedback connections from one time stamp to the next. Its ability to succinctly keep track of the history makes it particularly suitable and powerful for modeling temporal dependencies in sequential data.

The RNN based language model (LM) is one of the bigger successes of using RNN in recent years [2, 3] – on top of the standard feed-forward NN based LM [4, 5], recurrent connections were introduced and achieved impressive improvement in LM quality. RNN was more recently proposed for the slot filling task in the SLU area [6, 7], in which the task is to predict the slot tag for each word in the sequence. In addition to the success of using RNN, recent years have also witnessed the surge of interests in general NN based techniques for speech and language applications [8, 9, 10], some of them are specifically proposed for the utterance classification tasks in SLU, but limited to using features from the current turn [11, 12, 13].

Compared with previous related work, this paper targets a novel problem that has rarely been looked at – contextual domain classification for a multi-turn multi-domain SLU task. We also extend the work in [1] by employing RNN to extract features from the session context.

2. THE BENEFIT OF CONTEXT

For natural language conversations such as the examples shown in the previous section, features extracted from the current turn are often insufficient for statistical models to determine the type of the query accurately. This becomes increasingly true as the user dives deeper into the conversation and the correct semantic interpretation relies more and more on the session context.

Before presenting systematic experimental results, we would like to present a set of *oracle* experiments measuring the classification error rate of each turn and the potential gains at each turn by adding the previous domain label as a feature. The first turn has no informative session context and usually does not benefit from contextual modeling. As demonstrated in Table 1, the error rate using only non-contextual (n-gram) features increases monotonically as the conversation evolves, while the room for improvement provided by the previous domain label also grows monotonically (25% relative at turn 5).

Turn ID	n-gram only	+ true prev dom
1	12.3	**13.1** (+6.5%)
2	16.1	**13.5** (-16.1%)
3	16.4	**13.6** (-17.1%)
4	18.8	**15.7** (-16.5%)
5	20.3	**15.3** (-24.6%)
AVG	16.8	**14.5** (-13.7%)

Table 1. *Per-turn classification error rate (%) before and after adding the TRUE previous domain label as feature (SVM).*

3. CONVOLUTIONAL NEURAL NETWORK BASED DOMAIN CLASSIFICATION

Before presenting our RNN based model, we first describe its non-recurrent version based on the convolutional neural network (CNN). While other NN architectures for semantic classification exist [11, 12, 13], we like to present this work in the context of CNN feeding off our previous experiences [14, 15]. The choice of using CNN as opposed to others is not critical to the main theme of this work – recurrent connections can also be added to other NN based models. The comparison of them is out of the scope of this paper.

CNN has been used extensively for various learning tasks. By sharing the local feature extractor, it is able to reduce the number of parameters and capture some of the translational invariance in the input data. Longer range features can be extracted by *stacking* the CNN layers.

Using CNN as a multitask learning framework for natural language processing (NLP) was first described in [9]. The proposed architecture is a general technique for structured and non-structured classification tasks. For the purpose of non-structured classification (such as domain classification), the network structure is illustrated in Figure 1.

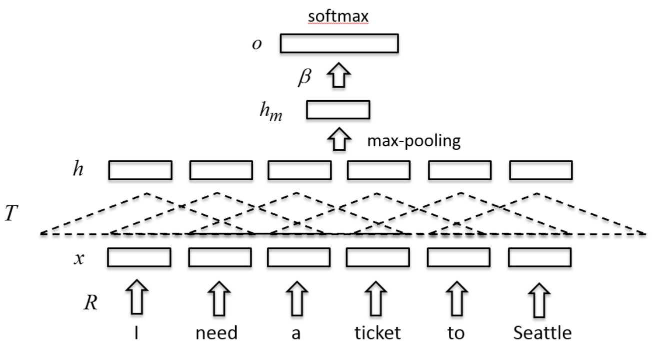

Fig. 1. CNN based classification.

As commonly done for NN based NLP techniques, each word is associated with a continuous vector. Such vectors, initialized either randomly or from some other modeling tasks, are concatenated to form the input layer x to the network. The transform T spans over each n-gram window and slides over the whole sentence. The resulting feature vectors in h usually go through some non-linear operation (such as $tanh$) before the *max-pooling* step produces a fixed-dimensional feature vector h_m. Finally, the transform β constructs a multinomial output distribution at the softmax layer o.

4. ADDING RECURRENT CONNECTIONS

Adding recurrent connections to the proposed CNN based architecture is no different from adding them to the standard NN models. Information from the history can be retained by appending to the current input feature layer either the previous hidden feature layer h_m or the previous output layer o. Depending on the source of the added features, the resulting architectures are called the Elman-type RNN [16] and the Jordan-type RNN [17] respectively.

In [7], it was shown that for slot filling, the Jordan-type RNN (propagating the output layer) gives slightly better results. While we have not compared the two types of architectures in our experiments, the Jordan-type RNN does have complexity advantages for our purpose – in the domain classification task that we are dealing with, the number of output labels is less than 10, which is significantly smaller than the typical size of the hidden feature layer (e.g. 100), resulting in much reduced complexity for computing the network. The Jordan-type recurrent CNN is depicted in Figure 2.

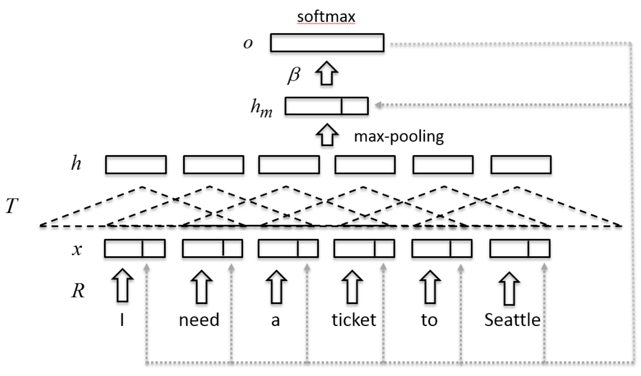

Fig. 2. CNN based classification with recurrent connections.

As illustrated by the figure, cycles are created in the network by connecting the output layer with each input vector in x, as well as the hidden layer h_m – the feature vectors in x and h_m are augmented with the network output vector from the previous time stamp, the linear transforms T and β are also expanded correspondingly. When there is no history available (e.g. sentences at the first turn), the augmented part of the vector is set to zero.

Note that the previous model prediction is fed into the network as a multinomial vector, as opposed to one single discrete feature indicating the predicted label from the previous turn (as done in [1]). Such *softened* features can help alleviate the negative impact of inaccurate model predictions for the previous turn.

The training of the proposed model can follow the same procedure for training general RNN based models – back propagation can be carried out on the *unfolded* network, which is commonly known as the back propagation through time (BPTT).

5. EXPERIMENTAL RESULTS

5.1. The Dataset

Our dataset contains 1461 natural language query sessions, each session consists of 5 turns on average. The queries in each session are sent to the SLU system in order. The examples of such sessions are demonstrated in section 1. The queries can be categorized in 9 different domains, ranging from checking weather, calling and texting, to setting reminders. The goal of our modeling task, is to assign the correct domain label to each query in the session.

Since we have limited amount of data, to reduce the variance of our evaluation results, all experiments are conducted in a cross-validation fashion. The dataset is partitioned into 10 fold. To test on each fold, the model is trained on 8 of the other 9 folds while the remaining 1 fold is used as the development set to tune various model settings and track the progress of online training.

5.2. Baseline Approaches

To compare with our RNN based approach, support vector machine (SVM) is our primary baseline. We use our internal SVM tool for all the experimentation. SVM is a powerful classification technique with both intuitive interpretation and strong theoretical guarantee. In essence, SVM is a linear model with regularized hinge loss as the optimization objective. In the linear modeling framework, features are usually handcrafted and the models are expressed as the dot product between the feature vector and the corresponding weight vector.

For text classification in general, n-gram features are commonly used as the baseline. For contextual classification, information from the history can also be encoded as feature constraints in the linear model. In [1], the predicted label from the previous turn was added as an additional feature and led to noticeable gains – this is another baseline approach we compare with.

While the kernel trick is often believed to be able to overcome the linear nature of the SVM model, we have found the basic linear SVM yields the best results for our experiments. Linear models in general are sometimes perceived as *shallow* learners [18], as opposed to NN based models in which more complex dependencies can be discovered via multiple layers of feature extraction. To compensate for the alleged "shallowness" of the linear SVM, we add a *product* feature directly modeling the joint effect of the previous domain label and the n-grams of the current turn. This can be perceived as context dependent n-gram modeling. For each domain label of the previous turn, we have a different set of n-gram features in addition to the shared n-gram features. It is worth pointing out that in our RNN model, such dependencies between previous label and n-grams are not explicitly defined – they are automatically discovered through NN layers.

For such SVM models using previous predictions as features, it is not possible to obtain the accurate contextual features during training – the model prediction will not be available before the model is fully trained. Instead of training on the true label of the previous turn and causing mismatch between training and testing, the prediction label during training is provided by a non-contextual classifier, namely an SVM that uses only n-gram features from the current turn. Such approximation is also adopted in [1].

5.3. RNN implementation

As we have described, the initialization of the word vectors can be either random or obtained from other tasks. The latter is essentially the *pretraining* step found to be crucial to the recent success of deep learning [19]. We have also found it useful in our experiments. Specifically, we initialize the word vectors using the SENNA embedding [20] obtained from an LM task on Wikipedia. The SENNA embedding provides a 50-dimensional vector for each one of the 130K words in its vocabulary. It covers 85% of the words on our dataset, the uncovered words are initialize randomly.

The hidden layer size for each convolutional unit is set it to be 100 for the experiments presented here. We use the rectifier activation function for the hidden units, and the dropout technique [21] is applied to the hidden features (with probabilty 0.5).

By design, the recurrent connections send the output distribution vector to the next turn. However, we have found that during training, sending a *one-hot* vector indicating the true previous label yields similar results. The benefit is that since the recurrent features are now constant, it generates no error signal for the previous turn. In other words, BPTT only has to be done for the current turn, thus greatly reducing the training complexity. We want to emphasize that sending the true previous label only happens for training – during test time, we still send the multinomial vector predicted by the model to the next turn.

5.4. Error Rate Comparisons

Table 2 demonstrates the domain classification error rate of the proposed RNN approach as well as the baseline techniques we described.

Model	Error rate (%)
SVM trigram only	16.8
SVM trigram + pdom	16.6
SVM trigram + pdom + CD 1g	16.3
RNN	**15.4**

Table 2. *Domain classification error rate (10-fold cross validation) of using RNN compared with using SVM with various feature sets. 'pdom' indicates the previous domain label; CD 1g indicates the context dependent unigram feature.*

As shown in the table, the RNN approach improves over the non-contextual SVM baseline by 1.4% absolute, 8.3% relative. Even against the SVM with contextual features, the RNN model yields a 0.9% absolute reduction (5.5% relative) in classification error rate. It is also worth mentioning that adding previous two domain labels as features for SVM only led to negligible additional improvement.

For the SVM models, the fact that the previous model prediction contains errors diminishes the potential gains substantially. The same problem also affects the RNN model, although the impact tends to be smaller because the predicted distribution (a multinomial vector) is used as features instead of the predicted label. According to Table 3, the prediction error costs about 2% absolute improvement for SVM, and 1.5% for RNN.

Model	True	Predicted
SVM trigram only	16.8	16.8
SVM trigram + pdom	14.5	16.6
SVM trigram + pdom + CD 1g	14.3	16.3
RNN	14.0	15.4

Table 3. *Difference in classification error rate of using true and predicted previous label.*

As we have shown in Table 1, later turns in a user session tend to have higer classification error rate. This also resulted in less and less accurate contextual features as the session evolves. On our multi-turn dataset, the negative impact of prediction errors appear to grow larger in later turns. As the Table 4 shows, while the first turn (for which there is no informative history) is not hurt by using the predicted features, from turn 2 to turn 5, we see a monotonic increase in the difference between using true and predicted features for RNN (from 1% to 2.6%). This problem is also reflected in the comparison between RNN and SVM: At turn 5, for which the model prediction is presumably the least reliable, there is no gain of using RNN over the SVM baseline. While the two approaches are also identical at turn 1, the benefit of using RNN is obvious at turn 2, 3, 4, and peaks at turn 3.

Turn ID	RNN True	RNN Pred	SVM Pred
1	13.8	12.9	12.9
2	13.1	14.1	14.6
3	12.5	14.6	16.7
4	14.7	16.9	18.4
5	16.1	18.7	18.7

Table 4. *Per-turn comparison of RNN and SVM (column 3 and 4), as well as using true and predicted label for RNN (column 2 and 3).*

6. CONCLUSIONS

We employed the recurrent neural network for contextual domain classification in a multi-turn multi-domain SLU session. The proposed RNN approach directly uses the previous model prediction as additional features for the current turn. The resulting model outperforms SVM with contextual features by significant margins.

7. REFERENCES

[1] A. Bhargava, A. Celikyilmaz, D. Hakkani-Tur, and R. Sarikaya, "Easy contextual intent prediction and slot detection," in *INTERSPEECH*, 2013.

[2] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010.

[3] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *ICML*, 2011.

[4] Y. Bengio, R. Ducharme, P. Vicent, and C. Jauvin, "A neural probablistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[5] H. Schwenk, "Continuous space language model," *Computer Speech and Language*, vol. 21, pp. 492–518, 2007.

[6] K. Yao, G. Zweig, M. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *INTERSPEECH*, 2013.

[7] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *INTERSPEECH*, 2013.

[8] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context dependent pretrained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, pp. 30–42, 2012.

[9] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *ICML*, 2008.

[10] A. Deoras and R. Sarikaya, "Deep belief network based semantic taggers for spoken language understanding," in *INTERSPEECH*, 2013.

[11] R. Sarikaya, G.E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *ICASSP*, 2011.

[12] G. Tur, L. Deng, D. Hakkani-Tur, and X. He, "Towards deeper understanding deep convex network for semantic utterance classification," in *ICASSP*, 2013.

[13] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernel deep convex networks and end-to-end learning for spoken langauge understanding," in *IEEE SLT*, 2012.

[14] P. Xu, S. Khudanpur, M. Lehr, E. Prud'hommeaux, D. Glenn, N. Karakos, B. Roark, K. Sagae, M. Saraclar, I. Shafran, D. Bikel, C. Callison-Burch, Y. Cao, K. Hall, E. Hasler, P. Koehn, A. Lopez, M. Post, and D. Riley, "Continuous space discriminative language modeling," in *ICASSP*, 2011.

[15] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *IEEE ASRU*, 2013.

[16] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, 1990.

[17] M. Jordan, "Serial order: A parallel distributed processing approach," *Tech. Rep. No. 8604, San Diego: University of California, Institute for Cognitive Science.*

[18] Y. Bengio and Y. Lecun, "Scaling learning algorithms towards ai," *Large-Scale Kernel Machines*, 2007.

[19] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, 2011.

[21] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv*, 2012.