Zero Shot Intent Classification Using Long-Short Term Memory Networks

Kyle Williams

Microsoft
One Microsoft Way, Redmond, Washington, 98052
kyle.williams@microsoft.com

Abstract

We describe a zero shot approach to intent classification that allows for the identification of intents that were not present during training. Our approach makes use of a Long-short Term Memory neural network to encode user queries and intents and uses these encodings to score previously unseen intents based on their semantic similarity to the queries. We test our model on intent classification in a personal digital assistant and show an improvement of 15% over a strong baseline. We also investigate the effect of adding a few training samples for the previously unseen intents in a few shot learning setting and show improvements of up to 16% over the baseline method.

Index Terms: conversational systems, spoken intent detection, zero shot learning

1. Introduction

Spoken Language Understanding (SLU) is an important component of Intelligent Assistant (IA) systems that interact with users, such as Siri, Cortana, and Google Assistant. One of the areas in which these assistants have shown the most promise is in task completion, where they are used to assist a user in completing a variety of predetermined tasks, such as setting an alarm or booking a taxi. In order to complete these tasks, IAs include a SLU component that is used to parse user utterances into predefined domains, intents and semantic slots [1]. Typically, each domain is implemented independently according to a schema that specifies the intents and slots associated with that domain. For instance, an IA designed to work in the hotel reservation domain may include intents for checking room availability and booking a hotel. However, it may not be able to handle new intents that are not part of the schema, such as booking a taxi to the hotel. A standard solution to this problem in SLU is to redesign the schema so that it covers the new intents as well as their associated slots. However, the downside of this is that redesigning a schema can be time consuming as it requires human annotation of new data as well as model retraining.

Zero shot learning is a promising area of research that attempts to address this problem by learning a classifier that is capable of assigning data points not only to classes that were seen during training, but also to entirely new classes [2]. Zero shot learning has successfully been applied in image recognition [2], neural activity decoding [3], and named entity recognition [4].

In this paper, we use zero shot learning with Long-short Term Memory Networks (LSTM) to identify intents that do not appear in the domain schema. The idea is to produce a classifier that is able to assign utterances to intents that occurred in the training data as well as intents that did not occur in the training data. To do this, we train an encoder to produce embeddings of intents and utterances. The intuition is that the encoder is able to capture the semantics such that the embeddings of utterances are *near to* the embeddings of their associated intents in the semantic space. Therefore, when a previously unseen intent is encountered, the encoder is able to produce an embedding for that unseen intent that is semantically *near to* the embeddings for utterances belonging to that intent. This is shown visually in Figure 1, which shows a 2D semantic space with 3 intents. An utterance projected to the same semantic space is positioned *near to* its associated *send_email* intent.

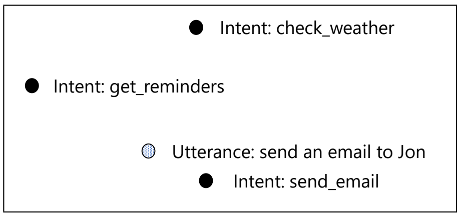

Figure 1: *A 2D semantic space with intents and an utterance.*

Previous work approached this problem using convolutional neural networks and the deep similarity model [5]. We instead propose to use a LSTM neural network as the encoder since LSTMs have produced state of the art on many natural language processing tasks [6]. Our intuition is that the sequential nature of the LSTM will be able to better capture the generic semantics of intents and utterances. We perform experiments in a zero shot setting to demonstrate the model performance. We also experiment with the model in a few shot setting where we make a small amount of training data for each class available at training time. We show how the proposed LSTM Encoder performs well in both an zero shot and few shot setting.

2. Related Work

Dauphin et al. [7] learn a semantic utterance using a DNN that is trained on search engine query logs. They train the DNN to predict the websites that a user clicks on for a given query. They then use the final layer of the DNN to produce semantic embeddings for user utterances and intents. Since their model is trained on query click logs and not semantic utterances, they also introduce an entropy-based method to make the semantic features discriminative for semantic utterance classification.

Chen et al. [5] proposed to use the Convolutional Deep Structured Semantic Model (CDSSM) for zero shot intent classification. In their model, they use a word hashing layer, followed by a convolutional layer, and then max pooling and feed forward layers to create encodings. They propose two models: a predictive model where they compute the probability of an intent given an utterance, and a generative model where they compute the probability of an utterance given an intent. In their experiments they find that there are able to achieve 9.07% accuracy on 7 previously unseen intents. As in our work, Palangi et al. [8] use an LSTM-based encoder for creating a similarity

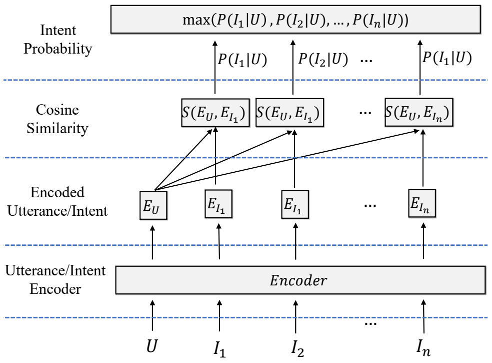

Figure 2: *An encoder encodes the utterance and each intent. The similarity is calculated and the utterance is then labeled with the intent that it is most similar to.*

model. However, they apply their model to search engine result ranking, whereas we apply our model to intent detection. In the area of natural language processing there has also been some work on zero shot learning for tasks beyond intent classification. For instance, Ma et al. [4] use hierarchical and prototypical features to learn label embeddings. Ferreira et al. [9] use generic word embeddings and ontological descriptions of the target domain to derive a zero shot semantic parser. Ferreira et al. [10] extend this work by proposing an online adaptive strategy with a small amount of supervision. Yazdani and Henderson [11] do zero shot learning of dialog acts. Their method uses word embeddings to map labels and utterances into a vector space where hyperplanes separate utterances with the same label from those with different labels.

3. Zero Shot Model for Intent Classification

In this section we describe the model that we use for zero shot intent classification. The end-to-end model is shown visually in Figure 2, and follows the same approach as previous similarity-based models [5, 8, 12, 13]. At its core, the zero shot model computes a similarity score between a user utterance and a set of intents and the utterance is labeled with the intent that it is most similar to. The similarity score is based on embeddings of the utterance and intents as produced by an encoder.

Formally, given a user utterance U and a set of intents I, the encoder produces a semantic representation of the utterance y_U and each intent $y_{I_j}, j = 1, 2, ..., |I|$. We use cosine similarity to measure the semantic similarity between an utterance U and an intent I_j, where the cosine similarity is given by: CosSim $(y_U, y_{I_j}) = \frac{\sum_{i=1}^{n} y_{U_i} y_{I_{j_i}}}{\sqrt{\sum_{i=1}^{n} y_{U_i}^2}\sqrt{\sum_{i=1}^{n} y_{I_{j_i}}^2}}$. The probability of an intent given an utterance is based on the softmax function:$P(I_j|U) = \frac{\exp CosSim(y_U, y_{I_j})}{\sum_{y_{I_k}, k!=j} CosSim(y_U, y_{I_k})}$

For training we minimize the negative likelihood of the correct intent for each associate training utterance. The focus of this work is in how the encodings y are produced. In this work, we propose to use a LSTM neural network as the encoder. We describe the encoding procedure in more detail next.

3.1. Long-short Term Memory Encoder

We propose to use a LSTM neural network as our encoder. The intuition for using an LSTM-based encoder is that it is well suited to sequential data, such as intents and user utterances, as they contain sequences of words and those words contain sequences of characters. The LSTM-based encoder is shown visually in Figure 3 and described in detail here.

We induce an encoding of utterances and intents based on the characters and words that appear in the utterances and intent labels (usually 2-3 words). We closely follow the approach of previous studies [14, 15, 16] and induce both character and word embeddings using bidirectional LSTMs. As in [14], for a given sequence of words $W = w_1, w_2, ..., w_n$ where word w_i has character $w_i(j)$ at position j. We define the following:

- Character embedding: e_c for each $c \in C$
- Character LSTM: ϕ_f^C, ϕ_b^C
- Word embedding: e_w for each $w \in W$
- Word LSTM: ϕ_f^W, ϕ_b^W,

where $\phi_f^C, \phi_b^C, \phi_f^W, \phi_b^W$ refer to the forward and backward character and word LSTMs. A character sensitive word representation v_i is computed as as:

$$f_j^C = \phi_f^C(e_{w_i(j)}, f_{j-1}^C), \forall j = 1...|w_i| \quad (1)$$

$$b_j^C = \phi_b^C(e_{w_i(j)}, b_{j+1}^C), \forall j = |w_i|...1 \quad (2)$$

$$v_i = f_{|w_i|}^C \oplus b_1^C \oplus e_{w_i}, \quad (3)$$

where $\oplus$ represents the vector concatenation operation whereby the final states of the forward and backward LSTMs are concatenated with the word embedding. Next the model computes:

$$f_i^W = \phi_f^W(v_i, f_{i-1}^W), \forall i = 1...n \quad (4)$$

$$b_i^W = \phi_b^W(v_i, b_{i+1}^W), \forall i = n...1 \quad (5)$$

In other words, the forward and backward word LSTMs are used to induce character and context sensitive word representations. The states of the forward and backward LSTMs are concatenated for the n-th word to induce the final encoding for the utterances and intents:

$$r = f_n^W \oplus b_n^W. \quad (6)$$

Finally, we build a semantic representation of the utterance or the intent:

$$y = tanh(W \cdot r + b) \quad (7)$$

These y are the encodings for utterances and intents that we use for computing utterance and intent similarity.

4. Experiments

In this section we present experiments evaluating our proposed model for zero shot and few shot learning. We describe the dataset we used, our baselines, and our methodology.

4.1. Data

We use a dataset of utterances and intent pairs from a commercial Intelligent Assistant (IA). The utterances were captured as speech and automatically transcribed using a production text-to-speech system and the intents were then labeled by trained annotators. In total, our dataset consists of almost 200,000 utterances belonging to 119 unique intents. The intents are usually made up of 1-3 words, such as *set_alarm* and *check_reminders*. The intents belong to common task completion domains in IAs, such as weather, reminders, calendar, email, restaurants, etc. We follow a similar approach to previous work on zero shot intent classification [5] and mimic the scenario where an initial set

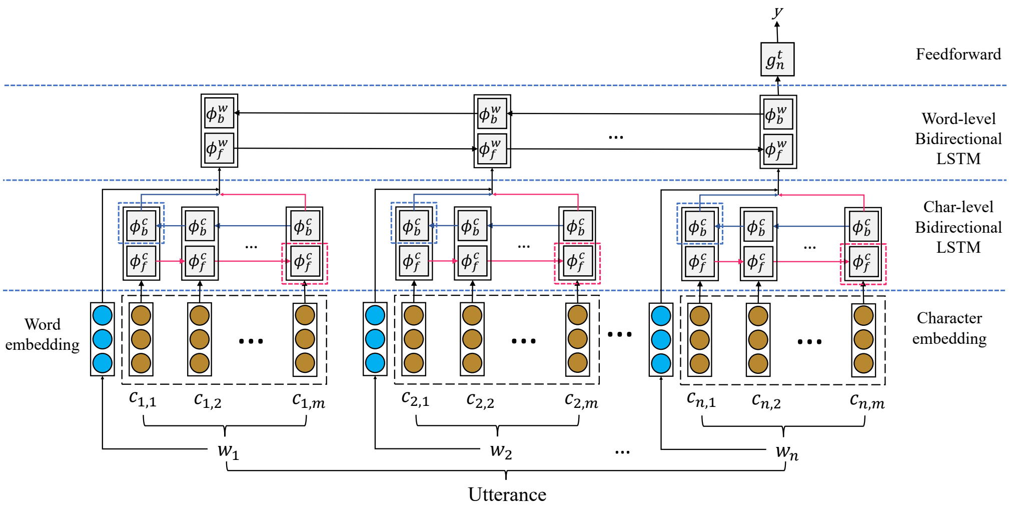
y
g_n^t
Feedforward
ϕ_b^w
ϕ_f^w
...
Word-level Bidirectional LSTM
ϕ_b^c
ϕ_f^c
Char-level Bidirectional LSTM
Word embedding
Character embedding
$c_{1,1}$ $c_{1,2}$ $c_{1,m}$
$c_{2,1}$ $c_{2,2}$ $c_{2,m}$
$c_{n,1}$ $c_{n,2}$ $c_{n,m}$
w_1
w_2
w_n
Utterance

Figure 3: *LSTM Encoder for producing semantic representations of utterance and intents.*

of intents is used to train the IA and then new intents are added to the schema later. Thus we randomly partition the utterances into train, validation, and test sets based on the intents. Partitioning the data by intent type ensures that there are no overlapping intents in the train, validation, and test sets. For the train set, there are 154,072 utterances from 89 intents. For the validation set there are 20,915 utterances belonging to 10 intents. The test set has 20,000 utterances belonging to 19 intents.

4.2. Baselines

LSTM with Softmax: This is a baseline bidirectional LSTM model defined by Equations 1-7. We take the softmax of the the output of Equation 10 to produce the probabilities of each intent and we train the model by minimizing the cross entropy loss. This model follows a standard supervised classification approach and therefore cannot classify previously unseen intents. However, we use it as a baseline to compare model performance for previously seen intents and for few shot learning.

Deep Semantic Similarity Model: We implemented the Deep Semantic Similarity Model [13] as an encoder as in Figure 2. The key difference between DSSM and our method is that DSSM uses word hashing and a series of feed forward layers to produce an encoding for an utterance or an intent. By contrast, our method uses an LSTM to produce this encoding. Excluding the encoder, everything else remains the same as described in Section 3. We also evaluated the Convolutional DSSM (CDSSM) [12], but found that it did not perform as well as the standard DSSM for this task.

4.3. Methodology

For all experiments we randomly initialize all model parameters and shuffle the training data. To account for randomization, we repeat each experiment 10 times and report the mean accuracy. For each experiment, we allow for up to 100 epochs of training and employ early stopping when there is no improvement in validation loss for 5 epochs. We make use of stochastic gradient descent and the Adam optimization algorithm [17] to train the network end-to-end to predict the correct intent for each utterance, thus allowing the network to learn the encoder for utterances and intents automatically.

For the baseline LSTM model and for our encoder we set the size of the character and word embeddings to 25 and 100, respectively. The character and word LSTMs have 25 and 100 units, respectively. For the DSSM model, we use the same network size as in [13] and make use of word hashing, with 2 nonlinear projection layers with dimension 300, and one semantic layer with dimension 128. To conduct experiments we train the DSSM and our LSTM Encoder and then evaluate them on previously seen intents, unseen intents, and in a few shot setting. We use the training and validation sets that do not have overlapping intents as described above to train the model and for early stopping. When training the LSTM Softmax model, we use validation sets that contain the same intents as the training set since the validation cannot be performed on the previously unseen intents.

4.4. Performance on Previously Seen Intents

Out first experiment tests the ability of the models to classify previously seen intents. To do this, we collect another 19,260 utterances with the same 89 intents as the training set to serve as a testing set. For the LSTM Softmax classifier we also collect 19,260 utterances to serve as a validation set, while the similarity-based models use the original validation set of 10 intents that do not occur in the training data as described in Section 4.1. The results of this experiment are shown in Table 2

As the table shows, the LSTM Softmax model, which represents a standard classifier, greatly outperforms the similarity-based approaches with an accuracy of 81.87%. The DSSM and LSTM Encoder models were able to learn to classify utterances into the 89 intents that occurred in the test set, even though they performed early stopping on a validation set with a completely different set of intents. These results show that a standard classifier is still the most appropriate in a setting where large amounts of training data are available and when the the intents used for testing match the intents used for training. However, they also suggest that the similarity-based methods are still able to learn to create meaningful semantic representations, even when a validation set with different intents is used.

4.5. Zero Shot Intent Classification

The previous experiment showed how a standard classifier is most appropriate if the training and test labels are the same.

Table 1: *Model accuracy for few shot learning where the model is initialized with the model trained for zero shot learning.*

Accuracy	0	1	2	3	4	5	6	7	8	9	10
LSTM Softmax	0.00	21.58	31.02	39.94	50.41	50.91	57.50	58.20	60.96	59.23	56.79
DSSM	28.60	33.61	40.38	45.34	49.22	47.35	53.69	56.86	57.57	60.12	54.58
LSTM Encoder (Ours)	**43.83**	**45.64**	**47.43**	**50.54**	**58.84**	**57.32**	**62.04**	**62.29**	**66.49**	**62.89**	**66.20**

Table 2: *Accuracy of models on previously seen intents*

Model	Accuracy
LSTM Softmax	81.87%
DSSM	57.20%
LSTM Encoder (Ours)	49.78%

Table 3: *Accuracy of models for zero shot intent classification*

Model	Accuracy
LSTM Softmax	0%
DSSM	28.60%
LSTM Encoder (Ours)	**43.83**%

However, we are concerned with the case where the training and test labels differ. We evaluate the models in zero shot setting where no samples of the testing intents were available during training. The results of this evaluation are shown in Table 3. For this experiment, we train the models with the same training set as before, which contains 89 intents. We then evaluate it on 19 previously unseen intents.

As would be expected, the baseline LSTM Softmax model is not capable of identifying any of the unseen intents since they were not in the training set. The proposed LSTM Encoder model outperforms the baseline DSSM model significantly with an accuracy of 43.83% for the proposed model compared to 28.60% for the DSSM model. This result indicates that the model is capable of detecting the correct intent of utterances 43% of the time, even though those intents were not in the training data. This experiment shows that the LSTM-based encoder is able to meaningfully encode utterance and intents and confirms our intuition that the sequential nature of the LSTM model is more appropriate than the deep feed forward architecture of the DSSM model for zero shot intent classification. This is similar to the finding in [8], where an LSTM-based encoder was shown to outperform DSSM for search engine result ranking.

4.6. Few Shot Learning

Another scenario that commonly occurs is when a few training samples are available for new intents. This scenario could occur when a designer for a new domain has some idea of the type of utterances that may occur for the intents. We evaluate the models under this scenario by labeling a few random utterances belonging to each of the 19 previously unseen intents for use in training. We experiment with $n = 1, 2, .., 10$ labeled utterances per intent. For each n, we use the models that were trained on the original 89 intents as initializers, and then continue training with the few new training samples. Our intuition is that the models trained on the 89 intents have already learned to produce meaningful features that can be used for computing semantic similarity and we are simply adapting that model to the new intents. We test the model on the same test set containing 19 intents as used in Section 4.5. We use a validation set containing 2,805 utterances with the same intents as the test set for all models. For the LSTM Softmax model we do not adapt a pre-existing model, but rather train the model from scratch to show how it performs under few shot learning conditions. The results of this experiment are shown in Table 1.

There are a few trends to notice from Table 1. The first is that the accuracy of the models increases as the number of few shot training examples increases. For instance, our proposed LSTM Encoder model goes from an accuracy of 43.83% in a zero shot setting to 45.64% when one training example is available and 47.43% when two examples are available. The accuracy of the LSTM Encoder model continues to increase until it reaches an accuracy of 66.20% when 10 training examples are available for each class. We observe a similar trend with the DSSM model where the zero shot accuracy is 28.60%, which increases to 33.61% and 40.38% for one and two training examples, respectively. When there are 9 training examples the accuracy reaches 60.12%. A similar trend can be observed for the LSTM Softmax model, though with lower accuracy. Overall, our proposed LSTM Encoder model achieves the highest accuracy for every number of samples in the few shot setting. It should be noted that the reason that accuracy results for this experiment are higher than the experiment with seen intents in Section 4.4 is because in this experiment the task is to differentiate among the 19 previously unseen intents, whereas in Section 4.4 the model needed to differentiate among 89 intents.

5. Discussion & Conclusions

In this paper we have explored zero and few shot learning for intent classification. The paper focuses on the common scenario that occurs where an SLU system needs to be expanded to support additional intents. Our experiments showed that the proposed LSTM-based encoder is a good candidate for both zero shot learning and few shot learning. In the zero shot learning setting it achieved an accuracy of of 48.83% compared to 28.60% for the DSSM model. For the few shot setting we showed how adding a few training examples of each testing intent leads to improvements in performance accuracy for all models with the proposed LSTM Encoder model performing best with each number of training samples. Furthermore, even though the LSTM Softmax model outperformed the similarity-based models in the experiment on previously seen intents when large amounts of training data were available, it performed worse than the similarity-based models in the few shot setting. Thus, these results suggest that the similarity models, and especially the LSTM Encoder, are a good choice in settings where there are only small amounts of training data available for new domains. The results are important as they provide a step in the direction of training zero shot models for SLU systems.

6. References

[1] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, "Investigation of Language Understanding Impact for Reinforcement Learning Based Dialogue Systems," *arXiv preprint arXiv:1703.07055*, 2017.

[2] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot learning through cross-modal transfer," in *Advances in neural information processing systems*, 2013, pp. 935–943.

[3] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in neural information processing systems*, 2009, pp. 1410–1418.

[4] Y. Ma, E. Cambria, and S. Gao, "Label embedding for zero-shot fine-grained named entity typing," in *Proceedings of the 26th International Conference on Computational Linguistics*, 2016, pp. 171–180.

[5] Y. N. Chen, D. Hakkani-Tur, and X. He, "Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 6045–6049.

[6] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[7] Y. N. Dauphin, G. Tur, D. Hakkani-Tur, and L. Heck, "Zero-shot learning for semantic utterance classification," *arXiv preprint arXiv:1401.0509*, 2013.

[8] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, p. 694, January 2016.

[9] E. Ferreira, B. Jabaian, and F. Lefevre, "Zero-shot semantic parser for spoken language understanding," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[10] ——, "Online adaptative zero-shot learning spoken language understanding using word-embedding," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5321–5325.

[11] M. Yazdani and J. Henderson, "A model of zero-shot learning of spoken language understanding," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 244–249.

[12] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for web search," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 373–374.

[13] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, 2013, pp. 2333–2338.

[14] K. Williams, "Neural lexicons for slot tagging in spoken language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, 2019, pp. 83–89.

[15] Y.-B. Kim, K. Stratos, and D. Kim, "Domain Attention with an Ensemble of Experts," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 643–653.

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural Architectures for Named Entity Recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[17] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.