Interspeech 2018
2-6 September 2018, Hyderabad

Capsule Networks for Low Resource Spoken Language Understanding

Vincent Renkens, Hugo Van hamme

Department Electrical Engineering-ESAT, KULeuven
Kasteelpark Arenberg 10, Bus 2441, B-3001 Leuven Belgium
vincent.renkens@esat.kuleuven.be, hugo.vanhamme@esat.kuleuven.be

Abstract

Designing a spoken language understanding system for command-and-control applications can be challenging because of a wide variety of domains and users or because of a lack of training data. In this paper we discuss a system that learns from scratch from user demonstrations. This method has the advantage that the same system can be used for many domains and users without modifications and that no training data is required prior to deployment. The user is required to train the system, so for a user friendly experience it is crucial to minimize the required amount of data. In this paper we investigate whether a capsule network can make efficient use of the limited amount of available training data. We compare the proposed model to an approach based on Non-negative Matrix Factorisation which is the state-of-the-art in this setting and another deep learning approach that was recently introduced for end-to-end spoken language understanding. We show that the proposed model outperforms the baseline models for three command-and-control applications: controlling a small robot, a vocally guided card game and a home automation task.
Index Terms: Spoken Language Understanding, Capsule Networks, Deep Learning, Low Resource

1. Introduction

In this paper we will discuss a spoken language understanding (SLU) system for command-and-control applications. The system can learn to map a spoken command to a task description. This description can then be given to some agent that can execute the task. An example for a command in a home automation application would be *"Turn on the light in the kitchen"*. This could then be mapped to the task `Switch(kitchen light, on)`. The task is represented by the type of action, `Switch` in this example, and a collection of arguments, `kitchen light` and `on` in this example. The SLU system learns to map the spoken command to a semantic representation, which is a collection of labels, one for the action type and one for each of the arguments.

Many approaches to this problem consist of an Automatic Speech Recognition (ASR) component that transforms the spoken command into a textual transcription and a Spoken Language Understanding (SLU) component that maps the textual transcription to the semantic representation [1, 2]. Such a system makes some assumptions about the user and how they are going to use the system. The ASR is typically trained for a single language, so it is assumed that the user will use this language. If the user has a pronounced accent or if the user has a speech impairment the ASR will often introduce a lot of errors [3], which makes it difficult for the SLU component to correctly determine the task to be performed. This is especially difficult for speech impaired users whose speech impairment is caused by another cognitive or motor disability. Users with such a disability often have difficulties using devices, so speech has a large potential to improve their way of living.

Simple SLU components, like one based on key phrases assume that the user is going to use some predefined commands. From a design perspective choosing these key phrases can be difficult to impossible for some applications. More advanced methods based on Recurrent Neural Networks (RNN) [4] or Conditional Random Fields [5] need lots of data to train, which may not be available in the domain of the application.

As an alternative we propose a system that learns to understand spoken commands directly from the user through demonstrations. The user can train the system by giving a spoken command and subsequently demonstrating the corresponding task through an alternative interface to the agent. The command *"Turn on the light in the kitchen"* can be demonstrated by pressing the button to turn on the light. This demonstration can then be converted to a semantic representation. The system directly maps speech to the semantic representation, without going through an intermediate textual representation. The system is trained using only the data from the user, which means that the assumptions and restrictions mentioned above do not apply. The disadvantage of such a system is that the user needs to give some examples, which requires some effort on their part. In order to minimize this effort it is crucial that the required amount of training data is as small as possible.

In the past we have proposed a method based on Non-Negative Matrix Factorisation (NMF) for this task [6, 7]. NMF performs significantly better than other, more conventional approaches like Hidden Markov Model (HMM) based approaches [8]. There are however limitations to the NMF approach. For example, the NMF approach uses a bag-of-words representation. It does not consider the order in which the words occur, which can be important to correctly interpret the command [9]. Deep learning based approaches have shown great performance on many speech-related tasks [10, 11, 12]. These models are based on Deep Neural Networks or RNNs and require a lot of data to train, which is not available in this setting. In this paper we propose to use a capsule network with a bidirectional RNN encoder. Capsule networks were proposed in [13] and it is suggested that they make more efficient use of the training data, making them better suited for this task.

We will discuss our proposed model in section 2. In section 3 we will describe the performed experiments and we will evaluate the results in section 4. Finally we will end with some conclusions in section 5.

10.21437/Interspeech.2018-1013

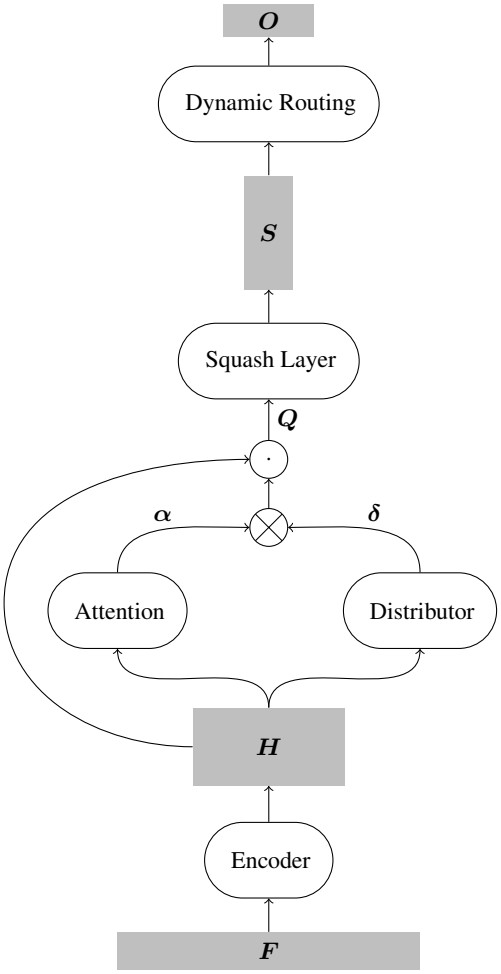


Figure 1: *A schematic of the proposed model*

2. Model

Our proposed model is presented in figure 1. The inputs to the model are a sequence of filter bank features $\boldsymbol{F}$. The inputs are first encoded into high level features $\boldsymbol{H}$. For this work a multi-layered bidirectional GRU [14] is used for this purpose. The sequence is Sub-sampled with a stride of 2 between the layers of the encoder as proposed in [15]. The sequence of high level features is therefore shorter than the sequence of input features.

The sequence of high level features is converted into capsules $\boldsymbol{S}$ using an attention mechanism [16] and a distributor. The concept of capsules was proposed in [13]. A capsule is represented by a vector. The direction of the vector represents the latent properties of the capsules. The norm of the vector lies between 0 and 1 and represents a probability that the capsule is present or not.

The attention module is used to determine a weight representing the importance of each timestep. Not the entire sequence is important to determine the meaning of the utterance (e.g. words like "please"). The attention module gives the model the capability to filter out the unimportant parts. The attention weights are determined using a sigmoid layer with a single output on the high level features:

$$\alpha_t = \text{sigmoid}(\boldsymbol{w}_\text{a} \cdot \boldsymbol{h}_t + b_\text{a}) \tag{1}$$

α_t is the attention weight for time t, $\boldsymbol{w}_\text{a}$ and b_a are the weights and bias of the sigmoid layer and $\boldsymbol{h}_t$ contains the high level features for time t.

The distributor is used to distribute each timestep to the hidden capsules $\boldsymbol{S}$. A distribution weight is determined from each timestep to each hidden capsule. Similar to the attention weights, the distribution weights are determined using a softmax layer on the high level features:

$$\boldsymbol{\delta}_t = \text{softmax}(\boldsymbol{W}^\text{d} \cdot \boldsymbol{h}_t + \boldsymbol{b}^\text{d}) \tag{2}$$

$\boldsymbol{\delta}_t$ contains the distribution weights for timestep t, one for each hidden capsule. $\boldsymbol{W}^\text{d}$ and $\boldsymbol{b}^\text{d}$ are the weights and biases of the softmax layer. Using the attention and distribution weights a context vector is created for each hidden capsule i:

$$\boldsymbol{q}_i = \sum_t \alpha_t \delta_{ti} \boldsymbol{h}_t \tag{3}$$

Where $\boldsymbol{q}_i$ is the context vector for capsule i. The context vectors are then converted to the capsule representation using a squash layer. The squash layer is a linear transformation followed by a squashing function:

$$\boldsymbol{s}_i = \sigma(\boldsymbol{W}^\text{s} \cdot \boldsymbol{q}_i) \tag{4}$$

$\boldsymbol{s}_i$ is the vector representation for capsule i, $\boldsymbol{W}^\text{s}$ are the weights of the squashing layer. Notice that no bias is included in the linear transformation to ensure that context vectors with a small norm result in capsules with a small norm. $\sigma(\cdot)$ is the squashing function as defined in [13]:

$$\sigma(\boldsymbol{x}) = \frac{||\boldsymbol{x}||^2}{1 + ||\boldsymbol{x}||^2} \frac{\boldsymbol{x}}{||\boldsymbol{x}||} \tag{5}$$

The squashing function ensures that the norm of the vector representations lies between 0 and 1. The output capsules $\boldsymbol{O}$ are computed using the iterative dynamic routing algorithm proposed in [13]. Every hidden capsule will predict the output of every output capsule using a linear transformation:

$$\boldsymbol{p}_{ij} = \boldsymbol{W}^\text{p}_{ij} \cdot \boldsymbol{s}_i \tag{6}$$

$\boldsymbol{p}_{ij}$ is the predicted vector representation of output capsule j from hidden capsule i and $\boldsymbol{W}^\text{p}_{ij}$ contains the weights for this prediction. The output capsules are computed using the coupling coefficients $\boldsymbol{C}$. The coupling coefficients represent how strongly linked the hidden capsules are to the output capsules. The coupling coefficients are computed using a softmax function on the coupling logits $\boldsymbol{B}$. The coupling logits are initialised with learnable values and then iteratively fine tuned with the dynamic routing algorithm:

Define variable $\boldsymbol{B}^{(1)}$;
for $n = 1{:}N$ **do**
  For all hidden capsules i: $\boldsymbol{c}_i = \text{softmax}(\boldsymbol{b}_i^{(n)})$;
  For all output capsules j: $\boldsymbol{o}_j = \sigma(\sum_i c_{ij} \boldsymbol{p}_{ij})$;
  For all logits $b_{ij}^{(n)}$ in $\boldsymbol{B}^{(n)}$: $b_{ij}^{(n+1)} = b_{ij}^{(n)} + \boldsymbol{p}_{ij} \cdot \boldsymbol{o}_j$;
end

Algorithm 1: Dynamic routing algorithm

At each iteration the output capsules are computed with the current connection logits. The connection logits are updated based on the agreement between the output capsule and the prediction. The agreement is measured using the scalar product. If the agreement between a prediction from a hidden capsule with an output capsule is large the connection logit will increase. The dynamic routing algorithm will look for groups of similar predictions for each output capsule. If there is a group of predictions that agree for a certain output capsule the capsule will become active and its norm will be close to one. If there is no such group the capsule will be inactive and its norm close to zero.

The probabilities of the output labels $\boldsymbol{l}$ are finally computed using the norm of the output capsules:

$$l_j = ||\boldsymbol{o}_j|| \tag{7}$$

The network is trained by minimizing the margin loss:

$$L = \sum_j t_j \max(0, 0.9 - l_j) + (1 - t_j) \max(0, l_j - 0.1) \tag{8}$$

where t_j is the target for label j, which is either 0 or 1.

3. Experiments

3.1. Datasets

The proposed model is tested and compared to the baselines for three datasets, in the domains of robotics, a card game and home automation.

The GRABO [17] dataset contains English and Dutch commands given to a robot. The robot can move in its environment, pick up objects and use a laser pointer. Typical commands given to the robot are "*move to position x*" or "*grab object y*". Output labels include positions in the world, the actions the robot can take etc. There are a total of 30 output labels. Data was recorded from 11 speakers issuing 36 different commands with 15 repetitions.

The PATCOR dataset [18] contains Dutch utterances from a vocally guided card game called Patience. The players can move cards or get new cards from the deck. Typical commands are "*Put card x on card y*" or "*New cards*". The output labels are the value and suit of the card being moved, the target position etc. There are a total of 38 output labels. Data was recorded from 8 speakers.

The DOMOTICA-3 dataset [7] is a follow up of the DOMOTICA-2 dataset [18] and contains utterances from Dutch dysarthric speakers using voice commands in a home automation task. Typical commands are "*open door x*" or "*turn on light y*". The output labels include all the lights, doors and all the actions the system can take. There is a total of 25 output labels. Data was recorded from 17 speakers with varying levels of dysarthria. Because speaking costs more effort for some speakers the amount of data per speaker varies greatly.

3.2. Methodology

We use cross-validation to get reliable experimental result. First, we split the data in multiple blocks. The blocks are chosen such that they are maximally semantically similar. This is done by minimizing the Jensen-Shannon Divergence between the blocks. We then create the training set by taking the data from a random set of blocks and the rest of the data is put in the testing set. We create learning curves by putting an increasing number of blocks in the training set. To get more reliable results we do 5 experiments for each number of blocks in the training set, each time with a different set of random blocks.

40 Mel filter banks + energy including first and second order derivatives with a window size of 25 ms and a window step of 10 ms are used as input features. A voice activity detector is used to remove long silences from the commands. The encoder consists of 2 bi-directional GRU layers with 256 units. There are 32 hidden capsules in $\boldsymbol{S}$ with 64 dimensions. There is one output capsule with a dimension of 8 for every output label. In total the network has around 2.2 million parameters. The model is trained with batches of 16 utterances for 30 epochs. Adam [19] is used as the optimization method with a learning rate of 0.001.

3.3. Baseline

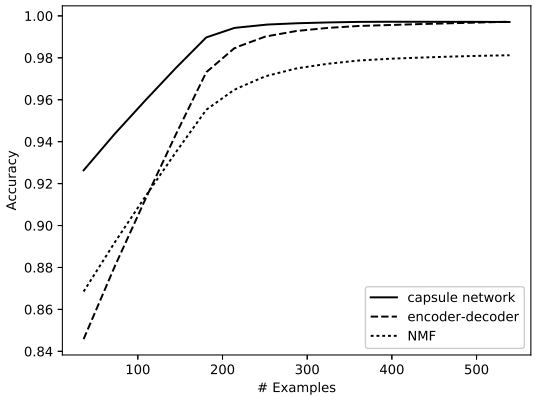
As a first baseline we use the method proposed in [6, 7]. This method is based on Non-negative Matrix Factorisation (NMF) that is used to decompose the input utterance into recurring patterns, which can be thought of as words. These words are linked to the output labels and in such a way a dictionary of words corresponding to the labels is created. This method achieves state-of-the-art performance for this task [8].

Alternatively we use a different deep learning approach proposed in [12] as a second baseline. This model was proposed in the context of end-to-end NLU to predict domain and intent labels for spoken utterances. The model consists of the same encoder, with the same number of layers and units, used in the current paper and a decoder. The decoder aggregates the high level features with max-pooling then applies a hidden ReLU layer with 1024 units followed by a sigmoid output layer to get the probabilities of the output labels. This network has around 2.3 million parameters. Adding more layers to the encoder did not improve the results. We will refer to this model as "encoder-decoder" in the results section.

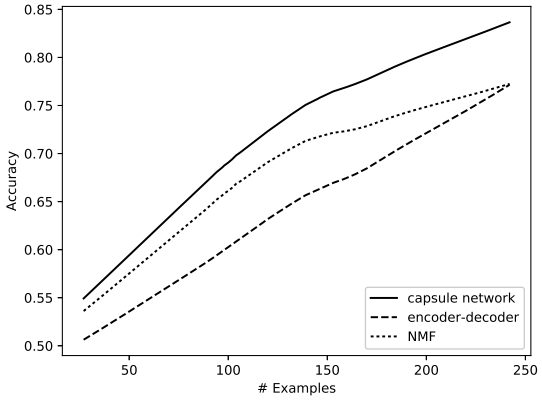
4. Results

In figure 2 the accuracy of the models is plotted as a function of the number of examples in the training set for all three data sets. In most cases the proposed model outperforms the baseline models. Only for the DOMOTICA-3 dataset, for a very small training set, the NMF model outperforms the capsule network. This may be caused by the fact that DOMOTICA-3 contains dysarthric speech, which is less consistent in terms of timing and pronunciation. NMF does not suffer a lot from this variability, but the GRU encoder might have more trouble modelling it. However, with a little bit more data the capsule network catches up with NMF and performs slightly better. For the GRABO dataset all models achieve a high accuracy, but the capsule network performs best. The encoder-decoder model does not perform well for small amounts of data, but catches up if more data is available.

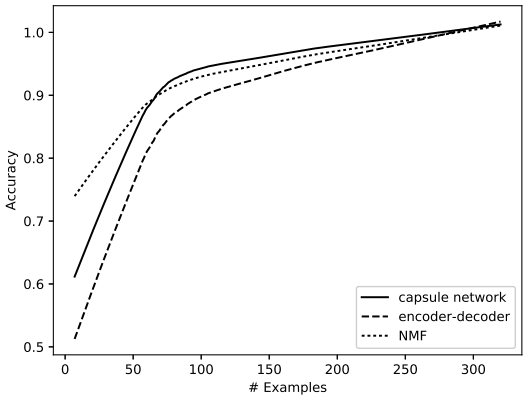
For the PATCOR dataset the accuracy for all models is significantly lower. This is because PATCOR is a more



(a)



(b)



(c)

Figure 2: *The accuracy plotted in function of the number of examples obtained by the proposed system and the baseline systems for the* `GRABO` *dataset (a), the* `PATCOR` *dataset (b) and the* `DOMOTICA-3` *dataset (c). LOWESS smoothing is used to obtain a smooth curve.*

challenging dataset. The user can look at the state of the game and they might leave out information in the command because it is obvious from the state of the game. For example if there is only one 3 that can be moved they might not mention the suit of the card to be moved. The state of the game is however not available to the NLU system in this setup, which introduces errors. In Dutch there are several names for each card. Some users alternate between these names, which also makes it more challenging for the NLU. Even on this more challenging task the capsule network performs better than the NMF model, especially with more training data. The encoder-decoder seems to have trouble with this more challenging task, which supports the findings in [20]

It is remarkable that the capsule network performs so well for only a couple dozen examples, which amounts to a few minutes of speech. These experiments seem to support the hypothesis that capsule networks make more efficient use of the training data, especially when you compare the capsule network with the encoder-decoder for small amounts of data.

5. Conclusions

In this paper we proposed a capsule network for low resource spoken language understanding for command-and-control applications. Only the data from the user is used to train the system, making it able to adapt to the domain of the application and the speaker without needing training data prior to deployment. The proposed model has been shown to significantly outperform the previous state-of-the-art. Even for small amounts data, a few dozen utterances, the capsule network performs well. In future work we will look more closely at the reason why the capsule network works well, especially for so little training data. It might also be interesting to investigate using a distributer together with an attention mechanism for attention based speech recognition.

6. Acknowledgements

The Research in this work was funded by PhD grant 151014 of the Research Foundation Flanders (FWO)

7. References

[1] Y.-Y. Wang, L. Deng, and A. Acero, "Spoken language understanding," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16–31, 2005.

[2] R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur, "Spoken language understanding," *IEEE Signal Processing Magazine*, vol. 25, no. 3, 2008.

[3] H. Christensen, S. Cunningham, C. Fox, P. Green, and T. Hain, "A comparative study of adaptive, automatic recognition of disordered speech," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[4] M. Dinarelli and I. Tellier, "Improving recurrent neural networks for sequence labelling," *arXiv preprint arXiv:1606.02555*, 2016.

[5] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[6] J. Gemmeke, B. Ons, N. M. Tessema, H. Van hamme, J. Van de Loo, G. De Pauw, W. Daelemans, J. Huyghe, J. Derboven, L. Vuegen, B. Van Den Broeck *et al.*, "Self-taught assistive vocal interfaces: An overview of the aladin project," in *Proceedings Interspeech 2013*. ISCA, 2013, pp. 2038–2043.

[7] B. Ons, J. F. Gemmeke, and H. Van hamme, "The self-taught vocal interface," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2014, no. 1, p. 43, 2014.

[8] J. F. Gemmeke, S. Sehgal, S. Cunningham, and H. Van hamme, "Dysarthric vocal interfaces with minimal training data," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 248–253.

[9] V. Renkens and H. Van hamme, "Weakly supervised learning of hidden markov models for spoken language acquisition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 2, pp. 285–295, 2017.

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[11] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[12] D. Serdyuk, Y. Wang, C. Fuegen, A. Kumar, B. Liu, and Y. Bengio, "Towards end-to-end spoken language understanding," *arXiv preprint arXiv:1802.08395*, 2018.

[13] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3859–3869.

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[15] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[17] V. Renkens, S. Janssens, B. Ons, J. F. Gemmeke, and H. Van hamme, "Acquisition of ordinal words using weakly supervised nmf," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 30–35.

[18] N. M. Tessema, B. Ons, J. van de Loo, J. Gemmeke, G. De Pauw, W. Daelemans, and H. Van hamme, "Metadata for corpora patcor and domotica-2," 2013.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] V. Vukotic, C. Raymond, and G. Gravier, "Is it time to switch to word embedding and recurrent neural networks for spoken language understanding?" in *InterSpeech*, 2015.