The Perfect Recipe: Add SUGAR, Add Data

Simone Magnolini[1,2], **Vevake Balaraman**[1,3], **Marco Guerini**[1], **Bernardo Magnini**[1]
[1] Fondazione Bruno Kessler, Via Sommarive 18, Povo, Trento — Italy
[2] AdeptMind Scholar [3] University of Trento, Italy.
{magnolini, balaraman, guerini, magnini}@fbk.eu

Abstract

English. We present the FBK participation at the EVALITA 2018 Shared Task "SUGAR – Spoken Utterances Guiding Chef's Assistant Robots". There are two peculiar, and challenging, characteristics of the task: first, the amount of available training data is very limited; second, training consists of pairs `[audio-utterance, system-action]`, without any intermediate representation. Given the characteristics of the task, we experimented two different approaches: (i) design and implement a neural architecture that can use as less training data as possible, and (ii) use a state of art tagging system, and then augment the initial training set with synthetically generated data. In the paper we present the two approaches, and show the results obtained by their respective runs.

Italiano. *Presentiamo la partecipazione di FBK allo shared task "SUGAR – Spoken Utterances Guiding Chef's Assistant Robots" a EVALITA 2018. Ci sono due caratteristiche peculiari del task: primo, la quantitá di dati di training é molto limitata; secondo, il training consiste di coppie* `[enunciato-audio, azione-sistema]`*, senza alcuna rappresentazione intermedia. Date le caratteristiche del task, abbiamo sperimentato due approcci diversi: (i) la progettazione e implementazione di una architettura neurale che riesca ad usare la minor quantitá di traning possibile; (ii) l'uso di un sistama di tagging allo stato dell'arte, aumentato con dati generati in modo sintetico. Nel contributo presentiamo i due approcci, e mostriamo i risultati ottenuti nei loro rispettivi run.*

1 Introduction

In the last few years, voice controlled systems have been arising a great interest, both in research and industrial projects, resulting in many applications such as Virtual Assistants and Conversational Agents. The use of voice controlled systems allows to develop solutions for contexts where the user is busy and can not operate with traditional graphical interfaces, such as, for instance, while driving a car or while cooking, as suggested by the SUGAR task.

The traditional approach to Spoken Language Understanding (SLU) is based on a pipeline that combines several components:

- An automatic speech recognizer (ASR), which is in charge of converting the spoken user utterance into a text.
- A Natural Language Understanding (NLU) component, which takes as input the ASR output and produces a set of instructions to be used to operate on the system backend (e.g. a knowledge base).
- A Dialogue Manager (DM), which selects the appropriate state of the dialogue, based on the context of previous interactions.
- A domain Knowledge Base (KB), which is accessed in order to retrieve relevant information for the user request.
- An utterance generation component, which produces a text in natural language by taking the dialogue state and the KB response.
- Finally, a text-to-speech (TTS) component is responsible for generating a spoken response

to the user, on the base of the text produced by the utterance generation component.

While the pipeline approach has proven to be very effective in a large range of task-oriented applications, in the last years several deep learning architectures have been experimented, resulting in a strong push toward so called end-to-end approaches (Graves and Jaitly, 2014; Zeghidour et al., 2018). One of the main advantages of end-to-end approaches is avoiding the independent training of the various components of the SLU pipeline, this way reducing the need of human annotations and the risk of error propagation among components. However, despite the encouraging results of end-to-end approaches, they still need significant amount of training data, which are often not available for the task at hand. This situation is also true in the SUGAR task, where, as training data are rather limited, end-to-end approaches are not directly applicable.

Our contribution at the SUGAR task mainly focuses on the NLU component, since we make use of an 'off the shelf' ASR component. In particular, we experimented two approaches: (i) the implementation a neural NLU architecture that can use as less training data as possible (described in Section 4), and (ii) the use of a state of art neural tagging system, where the initial training data have been augmented with synthetically generated data (described in Section 5 and 6).

2 Task and Data description

In the SUGAR task (Maro et al., 2018) the system's goal is to understand a set of `command` in the context of a voice-controlled robotic agent that acts as a cooking assistant. In this scenario the user can not interact using a "classical" interface because he/she is supposed to be cooking. The training data set is a corpus of annotated utterances; spoken sentences are annotated only with the appropriate `command` for the robot. Transcription from speech to text are not available.

The corpus is collected in a 3D virtual environment, designed as a real kitchen, where users give commands to the robot assistant to accomplish some recipes. During data collection users are inspired by silent cooking videos, which should ensures a more natural spoken production. Videos are segmented into short portions (frames), that contain a single `action`, and sequentially showed to users, who have to utter a single sentence after each frame. The user's goal is to guide the robot to accomplish the same `action` seen in the frame. The resulting dataset is a list of utterances describing the actions needed to prepare three different recipes. While utterances are totally free, the `command`s are selected from a finite set of possible actions, which may refer either to to ingredients or tools. Audio files are recorded in a real acoustic environment, with a microphone posed at about 1 mt of distance from the different speakers. The final corpus contains audio files for the three recipes, grouped for each speaker, and segmented into sentences representing isolated `command`s (although few audio files may contain multiple actions (e.g. "add while mixing")).

3 Data Pre-processing

The SUGAR dataset is constituted by a collection of audio files, that needs to be pre-processed in several ways. The first step is ASR, i.e., transcription from audio to text. For this step we made use of an external ASR, selected among the ones easily available with a Python implementation. We used the Google API, based on a comparative study of the different ASR (Këpuska and Bohouta, 2017); we conducted some sample tests to be sure that the ASR ranking is reasonable also for Italian, and we confirmed our choice.

After this step, we split the dataset into training set, development set and test set; in fact the SUGAR corpus is a unique collection and there is no train-dev-test split. Although the train-dev-test split is quite standard, with two round of 80-20 split of the dataset (80% of the dataset is the training and development set, which we split 80-20 again, and 20% is the test set), in the SUGAR task we split the dataset in a more complex way. In fact, the dataset is composed by only three different recipes (i.e. a small amount of ingredients and similar sequence of operations), and with a classical 80-20 split the training, the development and the test sets would have been too different from the final set (the one used to evaluate the system). This is due to the fact this new set is composed by new recipes, with new ingredients and new a sequence of operations. To deal with this peculiar characteristic, we decided to use the first recipe as test set and the other two as train-dev sets. The final split of the data resulted in 1142 utterance and `command` pairs for training, a set of 291 pairs for

development and a set of 286 pairs for test.

Finally we substituted all the prepositions in the corpus with an apostrophe (e.g. "d'" "l'", "un'") with their corresponding form without apostrophe (e.g. "di", "lo", "una"). This substitution helps the classifiers to correctly tokenize the utterances.

In order to take advantage of the structure of the dialogue in the dataset, in every line of the corpus we added up to three previous interactions. Such previous interactions are supposed to be useful to correctly label a sample, because it is possible that either an ingredient or a verb can appear in a previous utterance, while being implied in the current utterance. The implication is formalized in the dataset, in fact the implied entity (`action` or `argument`) are surrounded by $*$. The decision of having a "conversation history" of a maximum of three utterances is due to a first formalization of the task, in which the maximum history for every utterance was set to three previous interactions. Even if this constraint has been relaxed in the final version of the task, we kept it in our system. In addition, a sample test on the data confirms the intuition that usually a history of three utterances is enough to understand a new utterance. For sake of clarity, we report below a line of the pre-processed dataset:

un filo di olio nella padella # e poi verso lo uovo nella padella # gira la frittata # togli la frittata dal fuoco

where the first three utterances are the history in reverse order, and the final is the current utterance.

4 System 1: Memory + Pointer Networks

The first system presented by FBK is based on a neural model similar to the architecture proposed by (Madotto et al., 2018), which implements a encoder-decoder approach. The encoder consists of a Gated Recurrent Unit (GRU) (Cho et al., 2014) that encodes the user sentence into a latent representation. The decoder consists of a combination of i) a MemNN that generate tokens from the output vocabulary, and ii) a Pointer network (Vinyals et al., 2015) that chooses which token from the input is to be copied to the output.

4.1 Encoder

Each word in the input sentence x from the user is represented in high-dimension by using an embedding matrix A. These representations are encoded by a Gated Recurrent Unit. The GRU takes in the current word at time t and the previous hidden state of the encoder to yield the representation at time t. Formally,

$$h_t = GRU(h_{t-1}, x_t)$$

where x_t is the current word at time t and h_{t-1} is the previous hidden state of the network. The final hidden state of the network is then passed on to the decoder.

4.2 Decoder

The input sentences, denoted by $x_1, x_2, ..x_n$, are represented as memories $r_1, r_2, ..r_n$ by using an embedding matrix R. A query h_t at time t is generated using a Gated Recurrent Unit (GRU) (Cho et al., 2014), that takes as input the previously generated output word $\hat{y}_{t-1}$ and the previous query h_{t-1}. Formally:

$$h_t = GRU(\hat{y}_{t-1}, h_{t-1})$$

The initial query h_0 is the final output vector o output by the encoder. The query h is then used as the reading head over the memories. At each time-step t, the model generates two probabilities, namely P_{vocab} and P_{ptr}. P_{vocab} denotes the probability over all the words in the vocabulary and it is defined as follows:

$$P_{vocab}(\hat{y}_t) = Softmax(Wh_t)$$

where W is the parameter learned during training. The probability over the input words is denoted by P_{ptr} and is calculated using the attention weights of the MemNN network. Formally:

$$P_{ptr}(\hat{y}_t) = a_t$$
$$a_{t,i} = Softmax(h_t^T r_i)$$

By generating two probabilities, P_{vocab} and P_{ptr}, the model learns both how to generate words from the output vocabulary and also how to copy words from the input sequence. Though it is possible to learn a gating function to combine the distributions, as used in (Merity et al., 2016), this model uses a hard gate to combine the distributions. A sentinel token \$ is added to the input sequence while training and the pointer network is trained to maximize the P_{ptr} probability for tokens that should be generated from output vocabulary. If the sentinel token is chosen by P_{ptr}, then the model

switches to P_{vocab} to generate a token, else the input token specified by P_{ptr} is chosen as output token. Though the MemNN can be modelled with n $hops$, the nature of the SUGAR task and several experiments that we carried on, showed that adding more hops is not useful. As a consequence the model is implemented as a single hop as explained above.

We use the pre-trained embeddings from (Bojanowski et al., 2016) to train the model.

5 System 2: Fairseq

The second system experimented by FBK is based on the work in (Gehring et al., 2017). In particular, we make use of the Python implementation of the toolkit known as Fairseq(-py)[1]. The toolkit is implemented using PyTorch, and provides reference implementations of various sequence-to-sequence models. There are configurations for several tasks, including translation, language model and stories generation. In our experiment we use the toolkit as a black-box since our goal is to obtain a dataset that could be used with this system; hence, we use the generic model (not designed for any specific task) without fine tuning. Moreover, we do not add any specific feature or tuning for the implicit arguments (the ones surrounded by $*$), but we let the system learn the rule by itself.

A common approach in sequence learning is to encode the input sequence with a series of bi-directional recurrent neural networks (RNN); this can be done with Long Short-Term Memory (LSTM) networks, Gated Recurrent Unit (GRU) networks or other types of network, and generate a variable length output with another set of decoder RNNs, not necessarily of the same type, both of which interface via an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015).

On the other hand convolutional networks create representations for fixed size contexts, that can be seen as a disadvantage compared to the RNNs. However, the context size of the convolutional network can be expanded by adding new layers on top of each other. This allows to control the maximum length of dependencies to be modeled. Furthermore, convolutional networks allow parallelization over elements in the sequence, because they do not need the computations of the previous time step. This contrasts with RNNs, which maintain a hidden state of the entire past that prevents parallel computation within a sequence. This can increase dramatically the training time of the system without reducing the performance, as shown in (Gehring et al., 2017).

The weak point of the system is that it needs a consistent amount of training data to create reasonable models. In fact, Fairseq(-py) trained with only the SUGAR dataset can not converge and gets stuck after some epochs, producing pseudo-random sequences. Due to the small size of the SUGAR training set, combined with its low variability (training data are composed by possible variations of only two recipes), for the system is impossible to learn the correct structure of the commands (e.g. balancing the parenthesis) or to learn how to generalize `arguments`. In order to use effectively this system we have expanded the SUGAR dataset with data augmentation techniques, presented in Section 6.

6 Data augmentation

Overfitting is still an open issue in neural models, especially in situations of data sparsity. In the realm of NLP, regularization methods are typically applied to the network (Srivastava et al., 2014; Le et al., 2015), rather than to the training data.

However, in some application fields, data augmentation has proven to be fundamental in improving the performance of neural models when facing insufficient data. The first fields exploring data augmentation techniques were computer vision and speech recognition. In these fields there now exist well-established techniques for synthesizing data. In the former we can cite techniques such as rescaling or affine distortions (LeCun et al., 1998; Krizhevsky et al., 2012). In the latter, adding background noise or applying small time shifts (Deng et al., 2000; Hannun et al., 2014).

In the realm of NLP tasks, data augmentation has received little attention so far, some notable exceptions being feature noising (Wang et al., 2013) or Kneser-Ney smoothing (Xie et al., 2017). Additionally, negative examples generation has been used in (Guerini et al., 2018).

In this paper we build upon the idea of the aforementioned papers by moving a step forward and taking advantage of the structured nature of the SUGAR task and of some domain/linguistic knowledge. In particular, we used the following methods to expand the vocabulary and the size of the training data, but applying some substitution

[1] `https://github.com/pytorch/fairseq`.

strategies to the original data:

- **most-similar token substitution**: based on a similarity mechanisms (i.e. embeddings).
- **synonym token substitution**: synonymy relations taken from an online dictionary and applied to specific tokens.
- **entity substitution**: replace entities in the examples with random entities of the same type taken from available gazetteers.

The first approach implies substituting a token from a training example with one of the five most similar tokens (chosen at random) found through cosine similarity in the embedding space described in (Pennington et al., 2014). We use the top five candidates in order to add variability, since many tokens appeared multiple times in the training data. If the token appeared also as an `argument` in the `command`, it was substituted as well, while if it appeared as `action` it was left unchanged. This approach was applied with a probability of 30% on each token of the utterances in the training data.

The second approach has been used over verbs recognized in training utterances using the TextPro PoS tagger (Pianta et al., 2008). Such verbs have been substituted with one possible synonym taken from an electronic dictionary[2]. Also in this case, the `action` in the `command` was kept the same (in fact the verbs present in the utterance are usually paired with the `action` in the `command`). The third approach has been used to substitute ingredients in the text with other random ingredients from a list of foods (Magnini et al., 2018). In this case the ingredient has been modified accordingly also in the annotation of the sentence.

These methodologies allow to generate several variants starting from a single sentence. While the first approach has been used in isolation, the second and the third one have been used together to generate additional artificial training data. Doing so, we obtained two different data sets: the first is composed by 45680 pairs of utterances and `commands` (most-similar token applied forty times per example, $1142 * 40$); the second dataset contains 500916 pairs (each original sentence got at least each verb replaced 3 times, and for each of these variants, ingredients were randomly substituted twice), the high number of variants is due to the inclusion of the history of three previous utterances in the process.

[2] `http://www.sinonimi-contrari.it/`.

7 Results

	Actions	Arguments
Memory + Pointer Networks		
- Data Augmentation	65.091	30.856
+ Data Augmentation	65.396	35.786
Fine Tuning	66.158	36.102
Fairseq		
+ Data Augmentation	66,361	46,221

Table 1: Accuracy of the two experimented approaches in recognizing actions and their arguments.

Results of the two approaches are reported in Table 1. Both approaches obtain a higher accuracy in recognizing `actions`, than in recognizing `arguments`. Fairseq trained with augmented data is the top performer of the task, outperforming more than 10% of accuracy on `arguments` compared to the others approach. The ablation test on Memory + Pointer Networks also show the importance of data augmentation for tasks with low resources, in particular fine tuning the classifier with the new data.

8 Conclusion and Future Work

We presented the FBK participation at the EVALITA 2018 Shared Task "SUGAR – Spoken Utterances Guiding Chef's Assistant Robots". Given the characteristics of the task, we experimented two different approaches: (i) a neural architecture based on memory and pointer network, that can use as less training data as possible, and (ii) a state of the art tagging system, Fairseq, trained with several augmentation techniques to expand the initial training set with synthetically generated data. This second approach seems promising and in the future we want to deeper investigate the effect of the different techniques of data augmentation on the performances.

Acknowledgments

This work has been partially supported by the AdeptMind scholarship.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Li Deng, Alex Acero, Mike Plumpe, and Xuedong Huang. 2000. Large-vocabulary speech recognition under adverse acoustic environments. In *Sixth International Conference on Spoken Language Processing*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772.

Marco Guerini, Simone Magnolini, Vevake Balaraman, and Bernardo Magnini. 2018. Toward zero-shot entity recognition in task-oriented conversational agents. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 317–326, Melbourne, Australia, July.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Veton Këpuska and Gamal Bohouta. 2017. Comparing speech recognition systems (microsoft api, google api and cmu sphinx). *Journal of Engineering Research and Application*, 7(3):20–24.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *arXiv preprint arXiv:1804.08217*.

Bernardo Magnini, Vevake Balaraman, Mauro Dragoni, Marco Guerini, Simone Magnolini, and Valerio Piccioni. 2018. Ch1: A conversational system to calculate carbohydrates in a meal. In *Proceedings of the 17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018)*.

Maria Di Maro, Antonio Origlia, and Francesco Cutugno. 2018. Overview of the EVALITA 2018 Spoken Utterances Guiding Chef's Assistant Robots (SUGAR) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Emanuele Pianta, Christian Girardi, and Roberto Zanoli. 2008. The textpro tool suite. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Sida Wang, Mengqiu Wang, Stefan Wager, Percy Liang, and Christopher D Manning. 2013. Feature noising for log-linear structured prediction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1170–1179.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

Neil Zeghidour, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, and Emmanuel Dupoux. 2018. End-to-end speech recognition from the raw waveform. *Interspeech 2018*, Sep.