A STUDY OF DIFFERENT WEIGHTING SCHEMES FOR SPOKEN LANGUAGE UNDERSTANDING BASED ON CONVOLUTIONAL NEURAL NETWORKS

Jan Švec[1]*, Adam Chýlek*[1]*, Luboš Šmídl*[2]*, Pavel Ircing*[2]

[1] NTIS, [2] Department of Cybernetics, Faculty of Applied Sciences
University of West Bohemia, Pilsen, Czech Republic

[honzas,chylek]@ntis.zcu.cz [smidl,ircing]@kky.zcu.cz

ABSTRACT

This paper describes the development of a stateless spoken spoken language understanding (SLU) module based on artificial neural networks that is able to deal with the uncertainty of the automatic speech recognition (ASR) output. The work builds upon the concept of weighted neurons introduced by the authors previously and presents a generalized weighting term for such a neuron. The effect of different forms and parameter estimation methods of the weighting term is experimentally evaluated on the multi-task training corpus, created by merging two different semantically annotated corpora. The robustness of the best performing weighting schemes is then demonstrated by experiments involving hybrid word-semantic (WSE) lattices and also limited data scenario.

Index Terms— spoken language understanding, convolutional neural networks, word-semantic lattices

1. INTRODUCTION

The processing of an ASR output described in this paper is based on the use of convolutional neural networks (CNNs) for text classification [1] which use techniques known from image processing for problems of text processing. The input text can be treated as an 1D variable-size image with $|V|$ parallel channels (V being the vocabulary of the task) that is split into regions by a convolutional layer and the information contained in its output is reduced by a pooling layer to represent more abstract concepts. The output of the pooling layer is then fed to fully connected layers. Modifications to this technique using weighted neuron in convolutional layer presented in [2] and bag-of-words feature vectors allow us to exploit the additional information present in the ASR lattices.

The nature of SLU requires dealing with the erroneous and uncertain output of an ASR. The simplest approach is to use the 1-best recognition hypothesis and treat it as a text document processed in the CNN described above. However, exploiting n-best hypotheses (lattices) from ASR improves the SLU performance [3, 4].

The work presented in this paper first explored the possibility of training a single SLU module using multiple training corpora that were originally developed for different domains. The multi-task training approach is usually used to increase the number of training examples and to improve the generalization capabilities of the model (e.g. in a reasoning task [5] or in a speech recognition [6]).

The second part of this paper discusses different weighting schemes for incorporation of posterior probabilities into the CNN-framework. This approach is novel, because it uses parts of the input ASR lattice longer than one word and associated posterior probability. Other works on SLU typically use word confusion networks [3, 7, 4]. The proposed CNN scheme is not limited to a stateless SLU, but it could be used as a feature extractor in a recurrent neural network (RNN) SLU model [8].

The last set of experiments applies the multi-task training and weighting scheme to a new type of input data consisting of word-semantic (WSE) lattices. The WSE lattices were introduced in [9]. It allows to incorporate the expert knowledge into the SLU process by replacing well-known entities in the input lattice with a semantic tags described by a human-made grammar. The approach is similar to a LUNA framework [10] or to the use of finite-state transducers to detect local meanings [11, 12]. The only difference is that it operates on the ASR lattice and produces the WSE lattice, which allows to use the same SLU model for ASR and for WSE lattices.

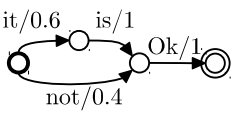
The rest of the paper is organized as follows: Sec. 2 gives a brief overview of CNNs for spoken language understanding, deals with the multi-task training and discuss the choice of weighting term, Sec. 3 shows the experimental results and Sec. 4 concludes the paper.

2. CNN FOR SPOKEN LANGUAGE UNDERSTANDING

In this section we shortly describe our method for incorporation of the information from ASR word lattice into an SLU pipeline based on convolutional neural networks. The CNN framework has the ability to encode variable-length input into a fixed-length feature-vector using the pooling operator. The fixed-length feature-vector is then processed in the fully-connected layers. The convolutional layer employs a convolutional region – the sliding window which is used to compute the region feature-vector. The number of region feature-vectors is variable and dependent on the length of the input. The CNN framework allows to estimate the parameters of the convolutional layer using standard backpropagation algorithms.

Because the word lattice encodes multiple ASR hypotheses and the corresponding posterior probabilities, it basically contains two types of information: (1) the *lexical information* contained in the words of the hypothesis and (2) the *posterior probabilities* of the given hypothesis. While the lexical information could be processed in the text-based CNN framework [1], the posterior probabilities would have to be properly modelled in an input layer of the network to appropriately weight the alternative hypotheses. The idea of the CNN framework processing the ASR lattices extends the concept of the sliding window, it moves in two dimensions: along the time axis catching the word-order information, and along the depth of the n-best ASR hypotheses. All obtained region feature-vectors are processed into a single pool using some standard pooling operator (max-pooling, sum-pooling, etc.)

In this work we use the ASR lattices preprocessed in a similar way as in [13]. Using operations defined over weighted finite state



(a) ↑

(b) →

l	p_l	[is	it	not	Ok	b_1	b_2]
0	0.6	[0	1	0	0	1	1]
1	0.6	[1	0	0	0	1	1]
2	0.4	[0	0	1	0	1	1]
3	1.0	[0	0	0	1	1	1]
4	0.6	[1	1	0	0	0	1]
5	0.6	[1	0	0	1	0	1]
6	0.4	[0	0	1	1	0	1]
7	0.6	[1	1	0	1	0	0]

Fig. 1: (a) The input lattice. (b) Region vectors ($\mathbf{r}_l$) and soft-counts (p_l) for all subpaths (indexed by l) of the lattice. The bag-of-words representation of region vectors is used. The auxiliary elements of the region vector p_1 and p_2 are paddings ensuring that the shorter subpaths have the same number of ones as the longer ones.

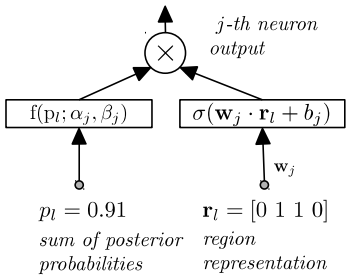


Fig. 2: Schema of a weighted neuron consisting of two non-linearities and multiplication. Note that the vector transposition symbols are omitted for clarity.

transducers the posterior probabilities of any subpath of the lattice can be easily computed [14]. In addition, we do not use the posterior probabilities directly. We sum the probabilities of the same regions forming so-called *soft-counts*.

Based on the results in [1] and with respect to the number of training examples and the average length of utterances in training data, we used the bag-of-words representation of region vectors (bow-CNN) and just one max-pooling unit. This configuration is suitable for small training sets and shorter utterances.

The input ASR lattices have a form of weighted finite state acceptors satisfying the assumption that the sum of probabilities of all paths is 1. According to this assumption the posterior probability of any subpath of the lattice could be computed. In our CNN architecture, we chose the subpaths of a maximum length of p words as the regions in the convolutional layer. Each region vector $\mathbf{r}_l$ is weighted by a corresponding sum of posterior probability (soft-count) p_l. The use of the pooling implies the order of $\mathbf{r}_l$-vectors is irrelevant.

We use padding elements b_i ensuring that the number of ones in the $\mathbf{r}$-vector is the same for all regions (see Fig. 1). The padding elements allow the network to distinguish between regions of different lengths. The padding elements could be understood as additional bias term in the convolutional layer of the network.

In [2], the authors introduced the *weighted neuron*. The weighted neuron is a convolutional layer neuron used to independently process the lexical information in the region vector $\mathbf{r}_l$ and the soft-count of the region vector p_l. The output of the j-th neuron for l-th region is defined as the multiplication of the standard neuron output and the weighting term:

$$y_{jl} = \sigma(\mathbf{w}_j \cdot \mathbf{r}_l + b_j) \times \mathrm{f}(p_l; \alpha_j, \beta_j) \tag{1}$$

where $\mathbf{w}_j$, b_j, α_j and β_j are weighting parameters shared between all convolutional units, $\sigma(\cdot)$ is a non-linearity of the convolutional unit and $\mathrm{f}(\cdot)$ is a weighting function with parameters α_j and β_j.

The max-pool output z_j is then simply the element-wise maximum of the convolutional neurons outputs:

$$z_j = \max_l y_{jl} \tag{2}$$

The fixed-length vector of max-pool output $\mathbf{z}$ is then processed in the fully connected layers of the neural network. The parameters $\mathbf{w}_j$, b_j, α_j and β_j are trainable using standard NN training algorithms.

The CNN framework could be used in many classification tasks. In [2] we used it for spoken language understanding. We used the Hierarchical Discriminative Model (HDM) [15], which was initially designed to be used with support vector machines, and we replaced the classification part with the CNN-based model. The HDM generates a semantic tree consisting of semantic concepts. Each semantic concept in the tree is expanded into a given set of successor concepts. The set of possible expansions is extracted from the training data and the expansion probabilities are predicted by the multiple-output classifier. The resulting semantic tree is the most probable semantic tree given the semantic grammar and predicted probabilities. In the CNN-based HDM, we used multiple soft-max output neurons with categorical cross-entropy as a training criterion.

The first experiment of this paper uses a multi-task training of CNN-based HDM. The experiment employing multiple semantic domains shows how well the HDM scales to larger tasks. In this work, we merged two semantically annotated corpora, virtually doubling the size of the task (see Tab. 1) in the number of different semantic concepts and the number of possible semantic trees.

To merge two different semantic corpora, we extended the first step of the HDM training algorithm – the collection of semantic concepts and their possible expansions. In our case it involves the unification of some concepts between the two semantically annotated corpora (Sec. 3). The HDM training process remains the same, the only difference is a larger number of output soft-max neurons.

2.1. Weighting term $\mathrm{f}(p; \alpha, \beta)$

As we have shown in [2], the choice of the weighting scheme for the posterior probabilities is crucial for a good SLU performance. The non-linear multiplication term used in the weighted neuron achieves better results than the simple extension of the region feature vector with the posterior probability. In this subsection, we will further study the influence of the weighting term on the SLU performance.

Since the weighting term is an integral part of the weighting neuron, its parameters could be estimated during the training process of the CNN. In this subsection, we will examine the form of the weighting term, the number of its parameters and the initial values of such parameters.

The weighting term maps the soft-counts of the region feature vectors to a multiplicative weight used to calibrate the output of convolutional neuron. The weighting term maps an interval $[0, +\infty)$ to a weight used to prioritize ASR hypotheses below the first best hypothesis. In this paper, we experimented with two forms of the weighting term: (a) *linear* and (b) *logistic sigmoid*. To unify the parametrization of the weighting term, we defined the weighting function $\mathrm{f}(p)$ using its value $\alpha = \mathrm{f}(0)$ and its derivation $\beta = \mathrm{f}'(0)$ for $p = 0$. The linear weighting term is a simple linear form:

$$\mathrm{f}(p; \alpha, \beta) = \beta \cdot p + \alpha \tag{3}$$

The logistic sigmoid is then parametrized as:

$$\mathrm{f}(p; \alpha, \beta) = \tanh(\beta \cdot p) + \alpha = \frac{2}{1 + e^{-2 \cdot \beta \cdot p}} - 1 + \alpha \tag{4}$$

The values of parameters α, β could be fixed, shared for all convolutional neurons or specific for each convolutional neuron. In the experiments below, we will denote it as *fixed*, *shared* and *specific*. The scenario with linear weighting term and fixed values $\alpha = 0$, $\beta = 1$ is equal to directly weighting the region feature vector r_j with the posterior probability p_j assuming the ReLU activation function and omitting the bias term in the j-th convolutional neuron output. In the shared scenario, all convolutional neurons share the same parameters, i.e. $\alpha_j = \alpha$ and $\beta_j = \beta$.

In the case of shared and specific scenarios, the gradient of the CNN cost function was derived and optimized using the Theano library [16]. The updates of the α and β parameters and CNN parameters were performed jointly.

3. EXPERIMENTAL EVALUATION

In the experiments we use the similar experimental setup as in [2]. The convolutional neural networks used in the experiments have the following architecture: Convolutional layer (1000 neurons); Pooling layer (1 max-pooling unit, ReLU, 1000 neurons); Dropout layer (probability 0.2); Hidden layer (ReLU, 1000 neurons); Dropout layer; Hidden layer (ReLU, 1000 neurons); Dropout layer; Hidden layer (sigmoid function, 1000 neurons); Dropout layer; Softmax layer (1 softmax unit per each semantic concept). We also use the bag-of-words region to reduce the number of trainable parameters and the training data augmentation with the human transcriptions without the ASR errors as described in [2]. To train the CNN network we used the ADAM [17] algorithm and early stopping approach implemented in Lasagne [18].

In the experiments we use two Czech semantically annotated corpora: a Human-Human Train Timetable (HHTT) corpus [19] which contains inquiries and answers about train connections; and an Intelligent Telephone Assistant (TIA) corpus containing utterances about meeting planning, corporate resources sharing and conference call management. These corpora contain unaligned semantic trees together with word-level transcriptions. We have split the corpora into train, development and test data sets (72:8:20) at the dialog level, so that the speakers do not overlap. To perform the multi-task training experiment, we had to unify some semantic concepts. In the HHTT corpus all time and date information was annotated as TIME. On the opposite site the TIA corpus contained more granular annotation of dates and times include concepts like TIME, INTERVAL, RELATIVE-DATE etc. To avoid re-annotation of the first corpus, we have merged all time- and date-related concepts into a TIME concept. We have also unified the concepts for agreement and disagreement, so that the resulting corpus contains ACCEPT and REJECT concepts.

To evaluate the SLU performance we use the *concept accuracy* measure [15] defined as $cAcc = \frac{N-S-D-I}{N} = \frac{H-I}{N}$ where H is the number of correctly recognized concepts, N is the number of concepts in reference and S, D, I are the numbers of substituted, deleted and inserted concepts. Our in-house LVCSR decoder with trigram language model trained from the in-domain data has been used to obtain the word lattices [20]. The recognition accuracy and other characteristics of both corpora are summarized in Tab. 1.

First of all, we train the CNN for the HHTT and TIA data separately. The results are shown in Tab. 2, rows HHTT and TIA. These models use the simple identity weighting term $\mathrm{f}(p) = p$. The union of these results (row HHTT+TIA) is a weighted average of particular results of two specialized models, the weights are the numbers of concepts in the corresponding set (see N in the $cAcc$ formula above). Then we train the multi-task model on the union of HHTT and TIA (row HHTT+TIA multi-task). The resulting model achieves virtually the same performance as the union of two models employing the prior information about the target domain. The same result can be achieved for other forms of the weighting term. In this work, we use this observation to train a single model on a larger corpus. This also allows us to evaluate the different weighting schemes on a single corpus.

	HHTT	TIA	multi-task
# different concepts	28	19	46
# different trees (train)	380	253	630
# train sentences	5240	6425	11665
# train concepts	8967	13499	22466
# dev. sentences	570	519	1089
# dev. concepts	989	1106	2095
# test sentences	1439	1256	2695
# test concepts	2546	2833	5379
ASR Acc	75.0%	77.9%	-
Vocabulary size	7143	4843	10469

Table 1: Corpora characteristics.

data	dev. cAcc	test cAcc
HHTT	71.89	71.76
TIA	77.94	84.22
HHTT+TIA	75.08	78.32
HHTT+TIA multi-task	74.08	78.40

Table 2: Single-task and multi-task training.

In the second set of experiments, we examine the influence of the weighting term. We use the multi-task data to create a larger dataset to evaluate the differences and we have tested two forms of weighting term – *linear* and *sigmoid* – in three setups – *fixed*, *shared* and *specific*. For each setup, we have performed experiments with $\alpha = \mathrm{f}(0) \in \{0, 1\}$ and $\beta = \mathrm{f}'(0) \in \{0.5, 1, 2\}$. The results on the development set of the multi-task corpus are shown in Tab. 3.

For each setup, we select the best performing weighting function and parameters (bold values in Tab. 3). These values were subsequently used to evaluate the model on the multi-task test set. Although the differences in the results are small, we can observe some regularities.

The results for weighting with non-zero offset ($\alpha = 1$) are better than the results for weighting term crossing the origin (in 12 cases of 18 total). The reason is probably that in order to effectively train the convolutional layer for regions with small soft-counts, it is necessary to assign them the non-zero weight to prevent the gradient from vanishing.

The average values of concept accuracy for a given weighting term and a given setup (last column of Tab. 3) suggests that the sigmoidal weighting performs better than the linear weighting. The average values also show that the ordering of setups from worse to the best is *fixed* $\rightarrow$ *specific* $\rightarrow$ *shared*.

The preference of the sigmoidal weighting is caused by the fact that we use the soft-counts of region feature vectors and such values could be greater than 1. The histogram shows a second maximum at soft-count 2. The preference of sigmoidal weighting suggests that the region with soft-count 2 is not twice important as a region with soft-count 1.

Almost all experiments outperform the baseline fixed identity weighting $\mathrm{f}(p) = p$, i.e. the linear weighting term in the fixed setup and $\alpha = 0$ and $\beta = 1$. It shows the importance of the choice of weighting function and its parameters. The values of weighting pa-

$f(p)$	β	dev. cAcc [%] $\alpha = 0$	dev. cAcc [%] $\alpha = 1$	avg cAcc [%]
linear (fixed)	.5	74.56	**76.04**	74.81
	1	74.37	75.32	
	2	74.32	74.22	
sigmoid (fixed)	.5	75.13	75.80	75.53
	1	75.18	75.47	
	2	75.99	75.61	
linear (specific)	.5	74.61	75.37	74.74
	1	74.08	74.80	
	2	74.80	74.75	
sigmoid (specific)	.5	74.89	75.66	75.61
	1	75.80	**76.04**	
	2	75.94	75.32	
linear (shared)	.5	75.61	75.23	75.33
	1	74.99	75.24	
	2	75.23	75.70	
sigmoid (shared)	.5	75.32	**76.23**	75.73
	1	75.94	75.89	
	2	75.32	75.70	

Table 3: Results for different weighting scenarios. The parameter α denotes the offset of the weighting function ($\mathrm{f}(0) = \alpha$), and the parameter β the derivation ($\mathrm{f}'(0) = \beta$). For scenarios with trainable parameters (specific and shared), these parameters are initialization values.

rameters could be determined by cross-validation (*fixed* setup) or by estimation during the CNN training process (*shared* and *specific* setups).

The selected weighting parameters for each weighting setup have been evaluated on the multi-task test data. We have also used the word-semantic (WSE) lattices from [9] as an additional type of input data. The WSE lattices have been created from ASR lattices by replacing the semantic entities (such as lexical realizations of times and dates) with semantic tags. We have modelled the semantic entities using the human-defined context-free grammars of the following types: *station*, *time*, *date*, *train_type*, *person_name* and *resource_name*. The occurrences of semantic entities in the ASR lattice have been replaced using the *split* derivation. The resulting WSE lattice contains the mix of words and semantic tags corresponding to the semantic entity, for more details see [9].

The evaluation on WSE lattices should demonstrate the robustness of a given weighting scheme – the parameters (or initial values for training) should be usable for another tasks. We have also experimented with a limited data scenario – in this case we randomly took 10% of the multi-task training data and we have trained the CNN SLU model. The model was evaluated on the full development and test data of multi-task corpus. The baseline for each case is the fixed identity weighting term $\mathrm{f}(p) = p$.

The experiments are summarized in Tab. 4 and Tab. 5. The results based on word lattices show that all three selected weighting schemes outperform the baseline fixed identity weighting (the p-value smaller than $2.2 \cdot 10^{-4}$). On the WSE level, the fixed identity weighting is outperformed also by all three weighting schemes, but the difference is statistically significant just for the *sigmoid (specific)* case (p-value 0.038). For the limited training data scenario, we observe that the differences between baseline weighting and the three selected weighting schemes are larger and all differences are statistically significant (p-value smaller than $3.2 \cdot 10^{-7}$).

The incorporation of WSE lattices into the training process of

data	weighting schema	dev. cAcc	test cAcc
words	linear $\mathrm{f}(p) = p$	74.08	78.40
words	linear (fixed)	76.04	**80.24**
words	sigmoid (specific)	76.04	80.05
words	sigmoid (shared)	76.23	79.53
WSE	linear $\mathrm{f}(p) = p$	75.12	80.50
WSE	linear (fixed)	75.66	81.06
WSE	sigmoid (specific)	75.23	**81.32**
WSE	sigmoid (shared)	75.70	80.94

Table 4: Full data scenario

data	weighting schema	dev. cAcc	test cAcc
words	linear $\mathrm{f}(p) = p$	61.24	67.24
words	linear (fixed)	64.34	70.66
words	sigmoid (specific)	64.53	**70.91**
words	sigmoid (shared)	64.53	70.70
WSE	linear $\mathrm{f}(p) = p$	63.39	70.63
WSE	linear (fixed)	66.35	73.14
WSE	sigmoid (specific)	65.97	**73.17**
WSE	sigmoid (shared)	66.54	73.14

Table 5: Limited data scenario

CNN SLU model significantly improves the SLU performance. The improvement from word lattices ($cAcc = 80.24\%$) to WSE lattices ($cAcc = 81.32\%$) is statistically significant with p-value 0.013. The improvement in the parsing accuracy is larger in the limited training data scenario – from 70.91% (word lattices) to 73.17% (WSE lattices) with p-value $3.0 \cdot 10^{-5}$.

4. CONCLUSION

We trained the multi-task CNN SLU model for two different semantic corpora. The multi-task model yielded the same accuracy as two partial models with the prior information about the target domain. Then, we studied different weighting schemes for weighting region soft-counts obtained from the lattice. The parameters of the weighting scheme could be fixed and determined using cross-validation. Another option is to estimate the weighting parameters as a part of the training process. In both cases, the results outperformed the baseline method, which multiplies the convolutional neuron output with the soft-count of the given region. We did not find any statistically significant difference between the three suggested weighting schemes.

We tested the suggested weighting schemes using word lattices in the multi-task scenario. In addition, we performed experiments with limited training data scenario. In both scenarios, we used also the word-semantic lattices. The results show that the suggested weighting schemes are robust and could be used for word and word-semantic lattices.

The word-semantic lattices proved to be useful also in the CNN-based SLU model and significantly improve the performance. The improvement is larger in the limited training data scenario, because the expert knowledge incorporated in the WSE lattices allows to better model rich and variable semantic entities such as times and dates.

5. ACKNOWLEDGMENTS

This research was supported by the Grant Agency of the Czech Republic, project No. GAČR GBP103/12/G084 and by the grant of the University of West Bohemia, project No. SGS-2016 "Intelligent Methods for Machine Perception and Understanding 3".

6. REFERENCES

[1] Rie Johnson and Tong Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks," , no. 2011, Dec. 2014.

[2] Jan Švec, Adam Chýlek, and Luboš Šmídl, "Hierarchical Discriminative Model for Spoken Language Understanding Based on Convolutional Neural Network," in *Proceedings of Interspeech 2015*, Dresden, DE, 2015, pp. 1864–1868, ISCA.

[3] Natthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young, "Discriminative Spoken Language Understanding Using Word Confusion Networks," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, 2012, pp. 176–181.

[4] Dilek Hakkani-Tür, Frédéric Béchet, Giuseppe Riccardi, and Gokhan Tur, "Beyond ASR 1-best: Using word confusion networks in spoken language understanding," *Computer Speech & Language*, vol. 20, no. 4, pp. 495–514, Oct. 2006.

[5] Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong, "Towards Neural Network-based Reasoning," pp. 1–12, 2015.

[6] Li Deng, Jinyu Li, Jui Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, Yifan Gong, and Alex Acero, "Recent advances in deep learning for speech research at Microsoft," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 8604–8608, 2013.

[7] Xiaohao Yang and Jia Liu, "Using Word Confusion Networks for Slot Filling in Spoken Language Understanding," in *Proceedings of Interspeech 2015*. 2015, pp. 1353–1357, ISCA.

[8] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig, "Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding," *IEEE/ACM Transactions on Audio*, vol. 23, no. 3, pp. 530–539, 2015.

[9] Jan Švec, Luboš Šmídl, Tomáš Valenta, Adam Chýlek, and Pavel Ircing, "Word-semantic lattices for spoken language understanding," in *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing 2015*, South Brisbane, QLD, 2015, pp. 5266–5270, IEEE.

[10] Géraldine Damnati, Frédéric Béchet, and Renato De Mori, "First implementation of the LUNA Spoken Language Understanding strategy on a telephone service application," *Intelligent Information Systems 2008*, pp. 499–505, 2008.

[11] Christian Raymond, Frédéric Béchet, Renato De Mori, and Géraldine Damnati, "On the use of finite state transducers for semantic interpretation," *Speech Communication*, vol. 48, no. 3-4, pp. 288–304, Mar. 2006.

[12] Christophe Servan, Christian Raymond, Frédéric Béchet, and Pascal Nocéra, "Conceptual decoding from word lattices: application to the spoken dialogue corpus media," in *Proceedings of International Conference on Spoken Language Processing*, Pittsburgh, 2006, pp. 1–4, ISCA.

[13] Dogan Can and Murat Saraclar, "Lattice Indexing for Spoken Term Detection," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[14] Jan Švec and Pavel Ircing, "Efficient Algorithm for Rational Kernel Evaluation in Large Lattice Sets," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2013*, Vancouver, Canada, 2013, pp. 3133–3137, IEEE.

[15] Jan Švec, Luboš Šmídl, and Pavel Ircing, "Hierarchical Discriminative Model for Spoken Language Understanding," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2013*, Vancouver, Canada, 2013, pp. 8322–8326, IEEE.

[16] F Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio, "Theano: new features and speed improvements," in *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*, 2012.

[17] Diederik P Kingma, "ADAM: A Method for Stochastic Optimization," pp. 1–15, 2015.

[18] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Soren Kaae Sonderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Brian McFee, Hendrik Weideman, Dr. Kashif Rasul, and Jonas Degrave, "Lasagne: First release; https://github.com/lasagne/lasagne," Aug. 2015.

[19] Filip Jurčíček, Jiří Zahradil, and Libor Jelínek, "A human-human train timetable dialogue corpus," *Proceedings of EUROSPEECH, Lisboa*, pp. 1525–1528, 2005.

[20] Aleš Pražák, Josef V. Psutka, Jan Hoidekr, Jakub Kanis, Luděk Müller, and Josef Psutka, "Automatic online subtitling of the Czech parliament meetings," *Text, Speech and Dialogue*, vol. 4188, pp. 501–508, 2006.