Fuzzy sets complement-based Gated Recurrent Unit

Mikel Ferrero-Jaurrieta[1], Graçaliz Pereira Dimuro[1,2], Zdenko Takáč[3], Regivan H. N. Santiago[4], Javier Fernández[1] and Humberto Bustince[1]

[1]*Department of Statistics, Computer Science and Mathematics, Public University of Navarre, Campus Arrosadía, s/n, 31006 Pamplona, Spain*

[2]*Centro de Ciências Computacionais, Universidade Federal do Rio Grande, Rio Grande, 96044540, Brazil*

[3]*Institute of Information Engineering, Automation and Mathematics, Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, Radlinského, 9, Bratislava, 812 37, Slovakia*

[4]*Department of Computer Science and Applied Mathematics, Universidade Federal do Rio Grande do Norte, Natal, 1524, Brazil*

Abstract

Gated Recurrent Units (GRU) are neural network gated architectures that simplify other ones (such as, LSTM) by joining gates mainly. For this, instead of using two gates, if x is the first gate, standard operation $1 - x$ is used to generate the second one, optimizing the number of parameters. In this work, we interpret this information as a fuzzy set, and we generalize the standard operation using fuzzy negations, and improving the accuracy obtained with the standard one.

Keywords

Fuzzy set complement, Fuzzy negations, Recurrent neural networks, Gated recurrent unit,

1. Introduction

Recurrent Neural Networks (RNN) [1, 2] were introduced in the 1980s to deal with sequential data modeling such as time series or text processing. Nevertheless, RNN had a big problem in training process, since the gradient value gradually tends to 0. This problem was known as vanishing gradient problem [3]. With the objective of solving this problem, Long Short-Term Memories (LSTM) were introduced in 1997 by S. Hochreiter and J. Schmidhuber [4]. They are based in a gated mechanism. LSTM networks have had various modifications [5]. In 2014 Cho et. al improved this system [6] simplifying the gated-based unit architecture using only a memory instead of long and short-memories.
To simplify the gate system, Gated Recurrent Unit (GRU) couples gates, using only one gate for modulating the information update of the cell. This means that, instead of using the $\mathbf{g_1}$ and $\mathbf{g_2}$ gates independently, they are coupled, using the following relation $\mathbf{g_2} = \mathbf{1} - \mathbf{g_1}$ [5] and consequently reducing the number of parameters to learn.
In this work, we consider the update process of the GRU unit as a fuzzy set [7] . In this sense, for a concrete element, a membership value near 0 means that the element is not going to update

WILF 2021: 13th International Workshop on Fuzzy Logic and Applications, Dec. 20–22, 2021, Vietri sul Mare, Italy
mikel.ferrero@unavarra.es (M. Ferrero-Jaurrieta); gracaliz.pereira@unavarra.es (G. P. Dimuro); zdenko.takac@stuba.sk (Z. Takáč); regivan@dimap.ufrn.br (R. H. N. Santiago); fcojavier.fernandez@unavarra.es (J. Fernández); bustince@unavarra.es (H. Bustince)

and a value near 1 means that is going to update almost fully. Therefore, the $1 - x$ operation can be understood as a negation or the complement of the fuzzy set in question. In this way, we generalize the expression $1 - x$ of the GRU equations by using fuzzy negations [8, 9, 10], and generating the complementary fuzzy set from these negations. Experimentally different fuzzy negations are considered, where both fixed expressions and values that are learned by the Gated Recurrent Unit itself are used. We test our results with a text classification dataset, and we show that our approach using different expressions [8] improves the performance of the GRU. The structure of this work is as follows. In Section 2 the fuzzy and GRU preliminaries are reminded. In Section 3 the GRU architecture modification is explained. In Section 4 the experimental framework and results are presented. Finally, some conclusions and future research are described in Section 5.

2. Preliminaries

In the present section we present the definitions and constructions of fuzzy negations and we also explain the main concepts about the GRU.

2.1. Fuzzy sets complementarity and fuzzy negations

From now on, we denote by X a non-empty and finite universe.

Definition 2.1. *[7] A fuzzy set A on X is given by $A = \{(x_i, A(x_i)) \,|\, x_i \in X\}$ where, by abuse of notation A denotes a map $A : X \to [0, 1]$. The value $A(x_i)$ is referred to as membership degree of the element $x_i \in X$ to the fuzzy set A.*

Definition 2.2. *[11] A function $N : [0, 1] \to [0, 1]$ is called a fuzzy negation if (N1) $N(0) = 1$ and $N(1) = 0$ and (N2) is decreasing: if $x \leq y$ then $N(x) \geq N(y)$ for all $x, y \in [0, 1]$.*

Definition 2.3. *[11] A fuzzy negation N is called strict if (N3) is continuous and (N4) is strictly decreasing, i.e. $N(x) < N(y)$ when $y < x$ for all $x, y \in [0, 1]$.*

Definition 2.4. *[11] A fuzzy negation N is called strong if it is an involution, i.e., (N5) $N(N(x)) = x$ for all $x \in [0, 1]$.*

Strong fuzzy negations are also strict fuzzy negations.

Example 2.5. (i) *The standard strong fuzzy negation is defined as $N_Z(x) = 1 - x$ known as the standard or Zadeh's Negation.*

(ii) *[11] Another examples of fuzzy negations are shown on Table 1 and represented on Figure 2.2.*

Definition 2.6. *[8] A function $\varphi : [0, 1] \to [0, 1]$ is an automorphism on the interval $[0, 1]$ if it is continuous, strictrly increasing and satisfies the boundary conditions $\varphi(0) = 0$ and $\varphi(1) = 1$.*

Theorem 2.7. *[10] A function $N : [0, 1] \to [0, 1]$ is a strong negation if and only if there exists an automorphism $\varphi : [0, 1] \to [0, 1]$ such that: $N(x) = \varphi^{-1}(1 - \varphi(x))$.*

Table 1
Fuzzy negation examples and their properties

Name	Negation function	Properties
	$N_K(x) = 1 - x^2$	(N1) - (N4). Strict.
	$N_R(x) = 1 - \sqrt{x}$	(N1) - (N4). Strict.
Sugeno class	$N^\lambda(x) = \frac{1-x}{1+\lambda x}, \lambda > -1$	(N1) - (N5). Strong.
Yager class	$N^{(\omega)}(x) = (1 - x^\omega)^{\frac{1}{\omega}}, \omega > 0$	(N1) - (N5). Strong.

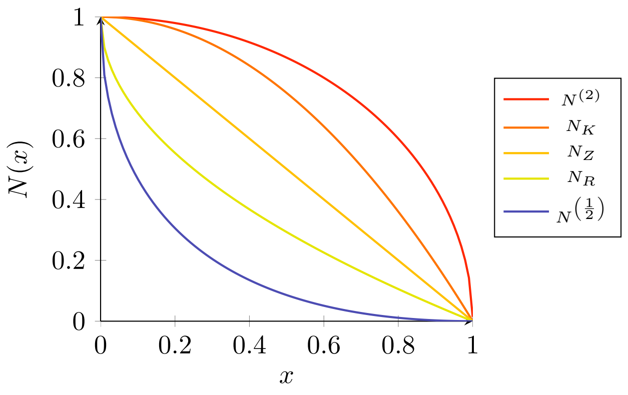


Figure 1: Graphical representation of different negations examples

The fuzzy negations constructed this way (Theorem 2.7) are called φ- transforms of standard negations.

Example 2.8. (i) *If we use $\varphi(x) = x^\omega$ as automorphism (Theorem 2.7), we obtain Yager class of negations (Table 1).*

(ii) *If we use $\varphi(x) = x^2$ ($\varphi(x) = \sqrt{x}$) as automorphism (Theorem 2.7), we obtain concrete examples of Yager class of negations $N^{(2)}(x) = \sqrt{1-x^2}$ ($N^{(\frac{1}{2})}(x) = (1-\sqrt{x})^2$), which is the same by evaluating Yager expression for $\omega = 2$ and $\omega = \frac{1}{2}$, respectively.*

Definition 2.9. *The complement of a fuzzy set A on X with respect to a fuzzy negation N is the fuzzy set $A_N : X \to [0,1]$ defined as $A_N = \{(x_i, N(A(x_i))) \mid x_i \in X\}$*

2.2. Gated Recurrent Unit (GRU)

In this subsection we explain the operation of the GRU [6]. Let k be the number of input sequence ($\mathbf{x}$), n the hidden size of the unit ($\mathbf{h}$) and T the number of timesteps. The input weight matrices are $\mathbf{W}_{zx}, \mathbf{W}_{rx}, \mathbf{W}_{hx}, \in \mathbb{R}^{n\times k}$, the recurrent weight matrices are $\mathbf{W}_{zh}, \mathbf{W}_{rh}, \mathbf{W}_{hh} \in \mathbb{R}^{n\times n}$ and the bias weight vectors are $\mathbf{b}_z, \mathbf{b}_r, \mathbf{b}_h \in \mathbb{R}^n$. The operations description for each timestep $t \in \{1, \dots, T\}$ is the following.
The input values $\mathbf{x}^{(t)} \in \mathbb{R}^k$ and $\mathbf{h}^{(t-1)} \in \mathbb{R}^n$ enter to the update (Eq. 1) and reset (Eq. 2) gates. In each of them, the value of $\mathbf{x}^{(t)}$ is multiplied by each of the input weight matrices ($\mathbf{W}_{zx}$,

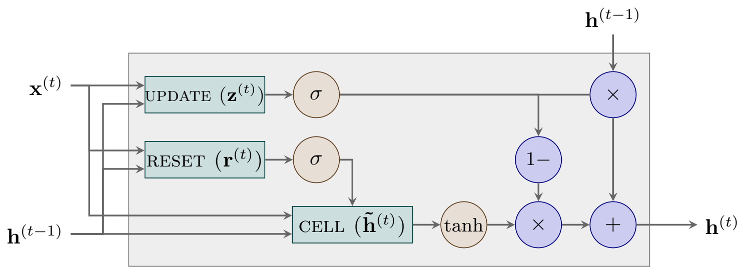


Figure 2: Graphical representation of GRU

$\mathbf{W}_{rx}$). The same occurs with the values of $\mathbf{h}^{(t-1)}$ and the recurrent weight matrices ($\mathbf{W}_{zh}$, $\mathbf{W}_{rh}$). The n-dimensional vectors obtained from these multiplications are fused summing with the corresponding bias $\mathbf{b}_z, \mathbf{b}_r$. As activation function non-linear sigmoid logistic function is used coordenate-wise ($\sigma : \mathbb{R} \to [0,1]$ where $\sigma(x) = \frac{1}{1+e^{-x}}$). Update vector ($\mathbf{z}^{(t)}$) represents the selection about which part of the current state should be removed and which part should be retained. Reset vector ($\mathbf{r}^{(t)}$) represents a weighting about which part of the previous step state is going to use in the calculation of the candidate activation.

$$\mathbf{z}^{(t)} = \sigma(\mathbf{W}_{zx}\mathbf{x}^{(t)} + \mathbf{W}_{zh}\mathbf{h}^{(t-1)} + \mathbf{b}_z) \qquad \text{(update gate)} \tag{1}$$

$$\mathbf{r}^{(t)} = \sigma(\mathbf{W}_{rx}\mathbf{x}^{(t)} + \mathbf{W}_{rh}\mathbf{h}^{(t-1)} + \mathbf{b}_r) \qquad \text{(reset gate)} \tag{2}$$

For the calculation of the candidate activation (Eq. 3), input value $\mathbf{x}^{(t)}$ is multiplied by $\mathbf{W}_{hx}$ matrix. Input value $\mathbf{h}^{(t-1)}$ is weighted with $\mathbf{r}^{(t)}$ by multiplying element-wise and the resultant vector is multiplied by $\mathbf{W}_{hh}$ matrix. As well as in the previous step, both n-dimensional structures are summed with $\mathbf{b}_h$. As activation function of the candidate activation the hyperbolic tangent $\tanh : \mathbb{R} \to [-1,1]$ is used coordenate-wise.

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{W}_{hx}\mathbf{x}^{(t)} + \mathbf{W}_{ch}(\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{b}_h) \qquad \text{(candidate activation)} \tag{3}$$

The previous timestep unit vector ($\mathbf{h}^{(t-1)}$) and the candidate activation ($\tilde{\mathbf{h}}^{(t)}$) are combined in this step. The Hadamard or element-wise product ($\circ$) is calculated between the values of the update gate ($\mathbf{z}^{(t)}$) and the complement of the update gate respect 1 ($\mathbf{1} - \mathbf{z}^{(t)}$) respectively (Eq. 4). Both values are added obtaining the current timestep value of the unit output vector, $\mathbf{h}^{(t)}$. The equations that describe the explained process are the following

$$\mathbf{h}^{(t)} = (\mathbf{1} - \mathbf{z}^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \circ \tilde{\mathbf{h}}^{(t)} \qquad \text{(output)} \tag{4}$$

3. GRU modification using fuzzy negations

Let n be the hidden size of the GRU (Section 2). In the GRU learning process, $\mathbf{z}^{(t)}$ vector represents the part of the current state is going to be retained and $\mathbf{1} - \mathbf{z}^{(t)}$ represents the part of the previous time step memory is forgotten. In this work, we generalize the second one, because the operation does not have the necessity to be a n-dimensional convex combination. Being $\mathbf{z}^{(t)} = (z_1^{(t)}, \ldots, z_n^{(t)})$ the update vector of the GRU, and having the non-empty finite universe $X = (x_1, \ldots, x_n)$ we can interpret $\mathbf{z}^{(t)}$ as a fuzzy set Z on X, where each vector element $z_i^{(t)}$ is the membership of an x_i element, hence, $Z(x_i) = z_i^{(t)}$ for all $i \in \{1, \ldots, n\}$ having the following fuzzy set:

$$Z = \{(x_i, Z(x_i)) \mid x_i \in X\}$$

If the membership of a element $x_i \in X$ to the fuzzy set is 0, this element is not updating ($\mathbf{h}^{(t)} = \mathbf{h}^{(t-1)}$), whereas if the membership is 1, is going to have a full update ($\mathbf{h}^{(t)} = \tilde{\mathbf{h}^{(t)}}$). Between 0 and 1 the membership and consequently the update measure is modelled by the fuzzy set Z, that is, the element updates a part, weighted by his membership to the fuzzy set. We can obtain the complementary set of Z with respect to a fuzzy negation $N : [0, 1] \to [0, 1]$ the following way:

$$Z_N = \{(x_i, N(Z(x_i))) \mid x_i \in X\} = \{(x_i, Z_N(x_i)) \mid x_i \in X\}$$

Here, we usually consider the standard negation N_Z to calculate the complement, although we also use different expressions (Table 1 and Figure 2.2). As in the case of the construction of Z, we can obtain a vector from Z_N as follows:

$$z_i^{(t)N} = Z_N(x_i) \text{ for all } i \in \{1, \ldots, n\}$$

obtaining $\mathbf{z}^{(t)N} = (z_1^{(t)N}, \ldots, z_n^{(t)N})$ This way, we modify the GRU Equation 4 replacing $1 - \mathbf{z}^{(t)}$ by the vector generated from the fuzzy complementarity with respect to fuzzy negations (Equation 5):

$$\mathbf{h}^{(t)} = \mathbf{z}^{(t)N} \circ \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \circ \tilde{\mathbf{h}}^{(t)} \tag{5}$$

4. Experimental study

In the present section, on one hand we explain the used framework (the dataset, the used neural network architecture,) and on the other hand we present the obtained results.

4.1. Experimental framework

4.1.1. Dataset

As the Gated Recurrent Unit improves other recurrent models in reduced datasets we have selected a small one. The dataset we use is *Text REtrieval Conference (TREC)* [12], which is a dataset for question classification. It contains 5500 questions in the training test and another 500 in the test one. The dataset is distributed in 6 classes.

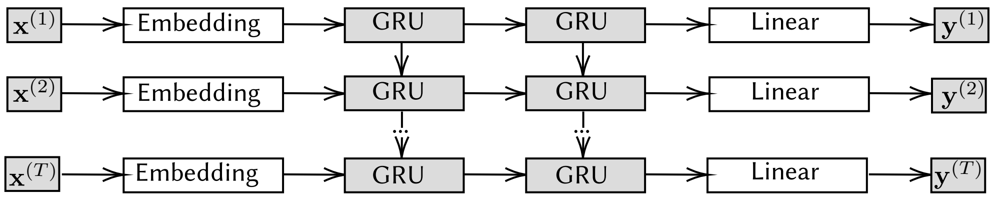


Figure 3: Graphical representation of the used double stacked GRU architecture

4.1.2. Architecture

The used architecture (Figure 4.1.2) is separated in four layers:

- *Embedding layer.* It consists in an algorithm designed to reduce the input dimensionality into a fixed one (in this case, 50) encoding the input words by means of vectors. Words with close representations have a greater relation.
- *Double stacked GRU layers.* Two Gated Recurrent Units with a hidden size of 64 each one.
- *Linear fully connected layer.* The second GRU is fully connected like a multilayer perceptron with a 6-node layer, which is a 6-dimensional probability vector. We classify as member of the class the vector position value that corresponds with the maximum.

4.1.3. Training hyperparameters

In this experiment, for each negation function, 10 independent runs of 30 epoch each are performed. The used optimization algorithm is Adam [13] and its fixed learning rate, $\gamma = 1 \times 10^{-3}$. The selected loss function is the Cross Entropy Loss.

4.1.4. Metrics

Once the architecture has been presented, we will go on to explain the metrics we will use to evaluate the experimental results. For each experiment i, the metric to be used is the accuracy on test set (acc), calculated as follows:

$$acc_i = \frac{\text{Number of correct predictions}}{\text{Total number of test dataset}} \tag{6}$$

for $1 \leq i \leq 10$ (number of experiments). For the evaluation of the experiments for each negation function, mean (Eq. 7) and standard deviation (Eq. 8) of accuracies of 10 experiments are calculated as follows:

$$\mu_{acc} = \frac{1}{10} \sum_{i=1}^{10} acc_i \tag{7}$$

$$s_{acc} = \sqrt{\frac{1}{9} \sum_{i=1}^{10} (acc_i - \mu_{acc})^2} \tag{8}$$

4.2. Experimental results

The results are presented in Table 2. The table is divided in two parts, regarding to the used negations: in the first part negations with fixed values are used and in the second part values are learned by the GRU. For each fuzzy negation expression, 10 independent runs have been executed and after the mean accuracy (Eq. 7) and its standard deviation (Eq. 8) are measured (Table 2).

According to the first part of the table, we can see that the best accuracy value is obtained when the expression of the circular negation $N^{(2)}(x) = \sqrt{1-x^2}$ is used, gaining 1.68 points of average accuracy with respect to the standard negation. Also better results than N_Z are obtained with $N_K(x) = 1 - x^2$. Taking account the first part of the table, we can resume that the best results are obtained when we use a fuzzy negation N fulfilling $N(x) > 1 - x$ for all $x \in (0, 1)$ (Table 2.2).

Regarding to the second part of the table, we have used the Sugeno class and the Yager class fuzzy negation expressions, each one depending by the parameters $\lambda \in (-1, \infty)$ and $\omega \in (0, \infty)$ respectively. These parameters are learnt by the recurrent neural network. As we can see in Table 2, both learnt expressions improve the ones selected by a fixed number. Concretely, the difference between the means of accuracy of the standard negation and the best learnt expressions is of 2.93 percentage points. This difference reflects the improvement in the use of other expressions and specifically those learned by the neural network itself. Regarding the average values learned by the network, for the Yager expression we obtain $\omega = 1.417$ for the first GRU and $\omega = 1.508$ for the second one. The standard deviation for each one is 0.016 and 0.028, respectively. For the Sugeno expression, we obtain $\lambda = -0.265$ and $\lambda = -0.384$, with the standard deviations 0.014 and 0.023, respectively. In both cases the standard deviations show that for the 10 independent runs, the obtained values have had very small differences. These learned numbers also show that better results are obtained when we use a fuzzy negation N such that $N(x) > 1 - x$ for all $x \in (0, 1)$. Regarding to a overall conclusion about the properties, we can also remark that the best 3 results are obtained using strong negations (Definition 2.4).

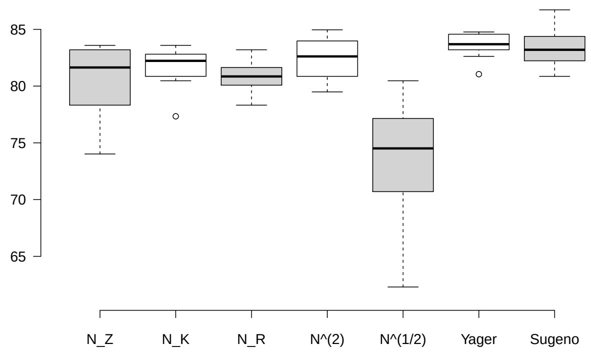
5. Conclusion

In this work we have interpreted as a fuzzy set a part of a GRU architecture and we have proposed the use of different negations to perform it. We have observed that better results are obtained using fuzzy negations N for which $N(x) > N_Z(x)$ for all $x \in (0, 1)$.

Regarding future lines of research, in the theoretical aspect our intention is to continue investigating about new ways to generalize and interpret recurrent neural networks operators, such as using n-dimensional fuzzy sets or extending the concept of fuzzy negation. On the applied side, future lines go on modifying other architectures, as well as using these architectures to other specific problems, such as language modelling.

Table 2
Mean Accuracy and standard deviation using different fuzzy negations

Name of Fuzzy Negation	Accuracy ($\mu_{acc} \pm s_{acc}$)
N_Z	80.64 ± 3.12
N_K	81.70 ± 1.73
N_R	80.82 ± 1.26
$N^{(2)}$	$\mathbf{82.32} \pm 1.73$
$N^{(\frac{1}{2})}$	73.74 ± 5.11
Best Sugeno class	83.28 ± 1.57
Best Yager class	$\mathbf{83.57} \pm 1.05$



Acknowledgments

Grant PID2019-108392GB-I00 funded by MCIN/AEI/10.13039/501100011033 and by Tracasa Instrumental and the Immigration Policy and Justice Department of the Government of Navarre.

References

[1] D. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[2] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, Studies in computational intelligence, Springer, Berlin, 2012. URL: https://cds.cern.ch/record/1503877. doi:10.1007/978-3-642-24797-2.

[3] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, in: S. C. Kremer, J. F. Kolen (Eds.), A Field Guide to Dynamical Recurrent Neural Networks, IEEE Press, 2001.

[4] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[5] J. van der Westhuizen, J. Lasenby, The unreasonable effectiveness of the forget gate, CoRR abs/1804.04849 (2018). URL: http://arxiv.org/abs/1804.04849. arXiv:1804.04849.

[6] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Ben-

gio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. arXiv:1406.1078.

[7] L. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353. doi:https://doi.org/10.1016/S0019-9958(65)90241-X.

[8] H. Bustince, P. Burillo, F. Soria, Automorphisms, negations and implication operators, Fuzzy Sets and Systems 134 (2003) 209–229. doi:https://doi.org/10.1016/S0165-0114(02)00214-2.

[9] H. Zapata, H. Bustince, L. D. Miguel, C. Guerra, Some properties of implications via aggregation functions and overlap functions, International Journal of Computational Intelligence Systems 7 (2014) 993–1001. doi:https://doi.org/10.1080/18756891.2014.967005.

[10] E. Trillas, Sobre funciones de negación en la teoría de conjuntos difusos., Stochastica 3 (1979) 47–60. URL: http://eudml.org/doc/38807.

[11] M. Baczyński, B. Jayaram, Fuzzy implications, in: Studies in Fuzziness and Soft Computing, 2008.

[12] X. Li, D. Roth, Learning question classifiers, in: Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, COLING '02, Association for Computational Linguistics, USA, 2002, p. 1–7. URL: https://doi.org/10.3115/1072228.1072378. doi:10.3115/1072228.1072378.

[13] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2015).