Vanilla Recurrent Neural Networks for Interpretable Semantic Textual Similarity

Piotr Andruszkiewicz
Warsaw University of Technology
piotr.andruszkiewicz@pw.edu.pl

Barbara Rychalska
Warsaw University of Technology
b.rychalska@mini.pw.edu.pl

Abstract

Semantic similarity systems assess to what extent two words, phrases, sentences are similar in their meaning. In this task, rule-based and neural network systems achieve the best results. The former requires intensive human workload and the latter needs heavy computing. Can we achieve high accuracy without hand-crafted rules and without intensive computing? In this paper we present three types of Vanilla Recurrent Neural Networks that fulfill the aforementioned requirements for Interpretable Semantic Textual Similarity task and we compare them to the systems from SemEval competition.

1 Introduction

Semantic similarity, also called paraphrase detection, focuses on similarity of words, phrases, sentences in terms of semantic equivalence. Having a different grammar construction or different words, yet synonyms, used, two sentences may convey the same meaning. Semantic similarity assesses the level of semantic equivalence, e.g., by saying that a pair of sentences is not semantically equivalent or is equivalent to some extent.

There are many different specifications of semantic similarity task. We focus on two definitions of this task used in SemEval competition (Cer et al., 2015). The first one is a basic approach, called herein basic semantic similarity task, that expresses semantic similarity as one number from 0 to 5. 0 means that there is no semantic similarity, e.g., *A woman is slicing big pepper.* vs. *A dog is moving its mouth.* 5 is assigned when there is perfect semantic equivalence, e.g., *The man cut down a tree with an axe.* vs. *A man chops down a tree with an axe.* In the middle there are pairs that are similar to some extent, e.g., a pair *People are playing cricket.* vs. *Men are playing cricket.* is scored as 3.

A more complicated specification of a task in question, called Interpretable Semantic Textual Similarity (iSTS), defines 8 alignment types between chunks of sentences, e.g., EQUI - semantic equivalence – *in Olympics*, *at Olympics*, OPPO - semantic opposition – *lower* vs. *higher*. Additionally, similarity for each alignment is scored on a scale 0-5. In this paper, we focus on Interpretable Semantic Textual Similarity (iSTS) task as it gives more insight in the justification of the similarity.

We can distinguish two main types of systems that assess semantic similarity, namely; hand-crafted rule-based systems and statistical systems. The first group uses many rules prepared by linguists and is highly customized for a specific task. Such systems need a lot of manual work. Statistical systems use trained models to assess semantic similarity. Some of them also involve lots of manual work in features engineering. We focus on statistical systems that do not require manual work and here comes deep learning approach.

Moreover, we prefer network architectures powered by basic network cell, because amount of annotated data is limited for Interpretable Semantic Textual Similarity (iSTS).

Hence, we focus on simple network models; that is, Vanilla Recurrent Neural Network (Vanilla RNN). Vanilla networks have been successfully applied in various tasks, for instance, char, word sequences processing, including text generation (Karpathy and Fei-Fei, 2017) or handwriting generation (Graves, 2013). The network cell we use is much simpler and has less parameters to train than more complicated gates, for instance, GRU/LSTM. Thus, we present three new network architectures with basic network cell and compare them to a baseline architecture and other available solutions for Interpretable Semantic Textual Similarity with one-to-one relations. In our work, we focus on achieving high accuracy using vanilla Recurrent Neural Networks with basic network cell in Interpretable Semantic Textual Similarity (iSTS) with one-to-one relations. That has been achieved by refining network architectures and by using basic network cell.

2 Related Work

Semantic similarity systems could be divided into two main groups: rule-based systems and statistical systems. An example of a rule-based system is UMBC EBIQUITY (Han et al., 2013), which won SemEval 2013 (Diab et al., 2013). Despite the already mentioned drawbacks of rule-based approaches, such systems have recently dominated the area of semantic similarity. They use external resources prepared by linguists, e.g., databases of synonyms, and heavily depend on manual work in rules creation. The main idea behind this kind of systems is to find words that are semantically similar in both sentences and calculate the level of semantic similarity for each pair. Then, the aggregated similarity is calculated. These systems could be supported by statistical models, however, the influence of a statistical model on the whole system is usually rather low.

The second group - statistical systems - are often based on neural network models. The first neural network that achieved high accuracy in semantic similarity task was presented in (Socher et al., 2011). It was based on autoencoder that encodes a sentence and then decodes it into another one being as close to the original sentence as possible. In this approach two encoded sentences are compared and assigned a score. The encoder transforms a sentence according to a dependency tree. Recently, recurrent neural networks (RNN) have been used for semantic similarity task. In (Tai et al., 2015), LSTM gate organized in a tree structure architecture has been used. The tree was built according to a dependency parse tree. In (Mueller and Thyagarajan, 2016) siamese RNNs with tied weights between networks were used. We do not apply this restriction in our solution. RNN approach is the current trend in semantic similarity.

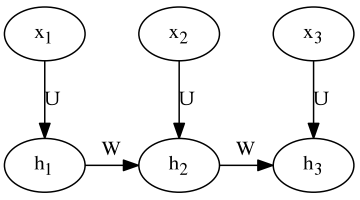

Figure 1: Linear Vanilla RNN.

There are also hybrid systems that combine rule-based approaches and statistical models. As they utilize advantages of these two kinds of systems, hybrid systems achieve high accuracy. A system of this type (Rychalska et al., 2016) won, for instance, semantic similarity task for English at SemEval 2016. Unfortunately, hybrid systems inherit also disadvantages of both of their combined approaches.

Our models are different from the above systems as they use statistical approach without time consuming manual work and apply basic network cell due to small available annotated data for iSTS.

3 Network Architectures

In this section, we present the baseline network architecture for Interpretable Semantic Textual Similarity task. We also propose three new architectures of recurrent neural networks built on top of basic network cell.

3.1 Baseline Linear Architecture

We use a linear vanilla architecture, denoted as VRNNH1-lin and shown in Figure 1, as a baseline for our three more complicated, yet still simple, vanilla architectures for semantic similarity assessment. x_i nodes represent terms from a sentence. Nodes may be a single value or a vector, e.g., an embedding vector (Turian et al., 2010), (Pennington et al., 2014) that represents semantics of a term. The hidden state of a network is calculated as $h_i = tanh(U \cdot x_i + W \cdot h_{i-1})$, where h_{i-1} is previ-

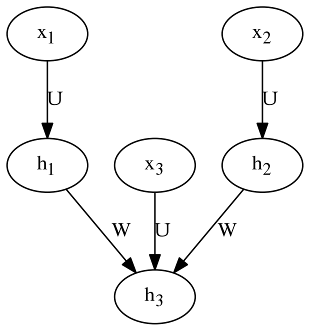

Figure 2: Tree Vanilla RNN.

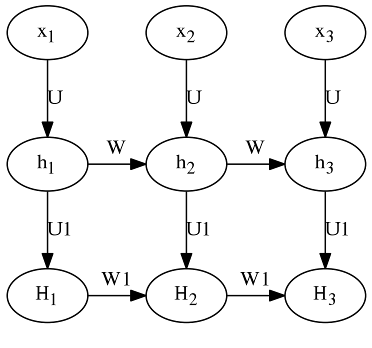

Figure 3: Linear Vanilla H2 RNN.

ous hidden state of the network and U, W are weight matrices.

3.2 Proposed architectures

The first of the proposed vanilla architectures (herein called VRNNH1-tree) is presented in Figure 2. A structure of a network is constructed according to a dependency tree of a sentence. The idea is similar to (Socher et al., 2011) (Tai et al., 2015), however, we use simpler vanilla unit. Compared to (Socher et al., 2011) we do not assume a decoding part of a network, as it is done in autoencoders. Moreover, we do not restrict a tree to be binary. Contrary to a linear network, a tree structured network does not process terms in the natural order as human beings do. However, we assume that a grammatical structure, represented by a parse tree, will be more beneficial for semantic representation of a sentence. The hidden state of a network is calculated as $h_i = tanh(U \cdot x_i + \sum_{j=1}^{m} W \cdot h_{i-1,j})$.

To extend the linear vanilla architecture, we add one more hidden layer. The extended architecture is depicted in Figure 3. A vector, H_i, for a node

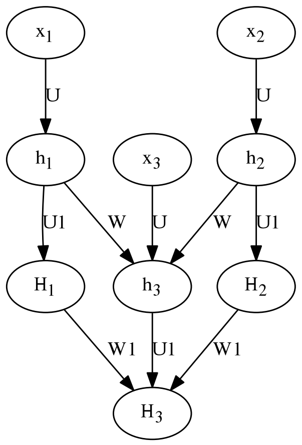

Figure 4: Tree Vanilla H2 RNN.

in the second hidden layer is computed as a non-linear function of a current hidden state h_i and a previous hidden state h_{i-1}. The additional non-linear function allows a network to model more complicated dependencies than those with only one hidden layer. Terms are processed according to a natural order of a sentence. h state is computed in the same way as in linear vanilla architecture: $h_i = tanh(U \cdot x_i + W \cdot h_{i-1})$. Additional hidden state is calculated as $H_i = tanh(U2 \cdot h_i + W2 \cdot H_{i-1})$. We denote this architecture as VRNNH2-lin.

The same extension can be applied to VRNNH1-tree network, thus Figure 4 presents a tree based network with additional hidden layer called VRNNH2-tree. Compared to VRNNH1-lin we try to leverage grammatical structure of a sentence and allow for modeling of more complicated dependencies by incorporating the additional hidden layer. The hidden state in the first hidden layer of a network is calculated like in VRNNH1-tree, as $h_i = tanh(U \cdot x_i + \sum_{j=1}^{m} W \cdot h_{i-1,j})$. The state from the second layer is computed as $H_i = tanh(U2 \cdot h_i + \sum_{j=1}^{m} W2 \cdot H_{i-1,j})$.

In order to evaluate two chunks/sentences in the context of semantic similarity, we need one more layer which gives a final result. We chose a softmax layer (Su and Xu, 2015) for this purpose. The 6 output neurons represent one of the 6 possible results on a 0-5 scale for *basic semantic similarity* task and 8 neurons represent alignment type in *Interpretable Semantic Textual Similarity* task. Each neuron of a softmax layer gives a probability of a

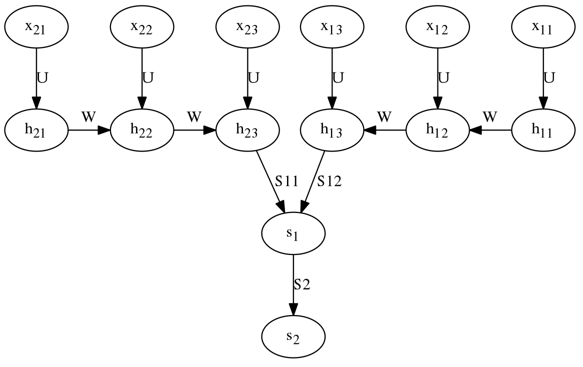

Figure 5: The full architecture of linear Vanilla RNN.

particular possible result. We can choose the result with the highest probability. A full network architecture combines final vectors representing two sentences and gives the final result. The full architecture for VRNN1-lin representation of a sentence is shown in Figure 5. h_{23} and h_{13} are nodes that output the vector representation for each of the two sentences. Then node s_1 compresses vectors representing sentences (e.g. 50 or 100 elements) to the length equal to the number of possible final outputs (e.g., six) for scoring estimation.

In the full network architecture presented in Figure 5, states for s_1 and s_2 are calculated as follows: $s_1 = tanh(S11 \cdot h_{23} + S12 \cdot h_{13})$, $s_2 = softmax(S2 \cdot s_1)$. For two hidden layer networks the final layers connect nodes from the second hidden layer, e.g., H_{23} and H_{13}. Please refer to Figure 6 that presents full network structure for VRNN2-lin sentence representation architecture. For tree structured architectures the idea of combining vectors representing each sentence is the same.

4 Non-Integer Scores

Scores provided in SemEval data are not always integers. Let us assume that the score is 3.75. Then we can guess that it comes from score 4 assigned by three annotators and score 3 assigned by one annotator. We need to choose the approach to process such scores in the softmax layer.

4.1 Scores Weighting

First approach is to reconstruct the distribution of scores. If we know that for 3.75 three persons assigned score 4 and one person chose score 3, we

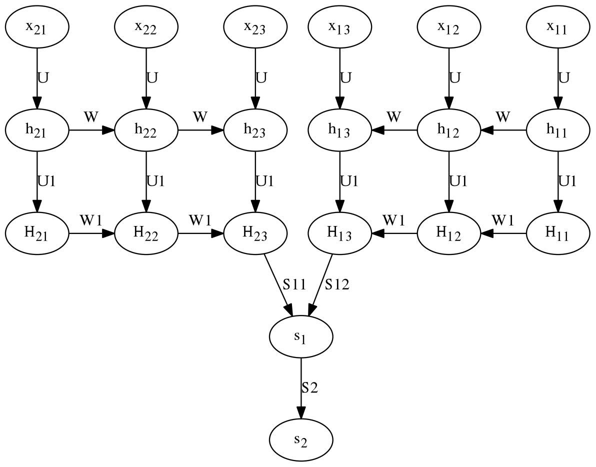

Figure 6: The full architecture of linear Vanilla H2 RNN.

Set	Training	Testing	Total
Images	375	375	750
Headlines	378	378	756

Table 1: Number of sentence pairs in the data sets.

can represent a softmax layer as (0 0 0 0.25 0.75 0) where score 3 has a probability of 0.25 and score 4 has a probability of 0.75. Thus, we take into account minority votes.

4.2 Scores Rounding

We may also think, in case of score 3.75, that score 3 was a mistake or deviation from the proper score. Hence, we round the score to the most probable one and omit the minority votes by taking into account only majority votes. In such a case we would represent the softmax layer as (0 0 0 0 1 0) that means we assume the score of 4 should be assigned.

We examine both approaches in Section 5.

5 Experiments

We test the proposed architectures with Images and Headlines (Agirre et al., 2015) data sets from SemEval 2015 Interpretable Semantic Textual Similarity (iSTS) contest to compare our models to systems submitted by SemEval participants.The data sets contain around 750 sentence pairs each (please refer to Table 1. Golden chunks, scores and types are also provided.

We also use F measures applied in SemEval iSTS

Model	F1 score	F1 s+type	Avg
H1-lin	0.8620*	**0.6894**	**0.7757**
H1-tree	0.8516	0.6819*	0.7668*
H2-lin	0.8541	0.6679	0.7610
H2-tree	**0.8651**	0.6825*	0.7738*

Table 2: Experiments summary for 50 elements embedding vectors for Images set.

Model	F1 score	F1 s+type	Avg
H1-lin	0.8559*	0.6624*	0.7592*
H1-tree	0.8578*	0.6643*	**0.7611**
H2-lin	**0.8617**	0.6557*	0.7587*
H2-tree	0.8502	**0.6704**	0.7603*

Table 3: Experiments summary for 100 elements embedding vectors for Images set.

contest. *F1 score* (Agirre et al., 2015) is F1 measure that takes into account the score assigned to the alignment. The score should match. The alignment type is ignored.

F1 score+type (also *called F1 s+type*) (Agirre et al., 2015) is F1 measure that takes into account both alignment type and score for alignments.

In our tests we use golden chunks, thus the algorithm does not need to discover chunks within the sentences because it uses chunks provided in the data set.

In the experiments, we use word embeddings described in (Turian et al., 2010) and log-loss function as a loss function. To calculate a gradient, Limited-memory BFGS algorithm is used (Liu and Nocedal, 1989). Regularization is performed with L2 method. The source code is available at https://www.dropbox.com/s/ot25z73qnhue92f/vrnn1.0.zip?dl=0 (anonymized link).

Model	F1 score	F1 s+type	Avg
H1-lin	0.8560	0.6942*	0.7751*
H1-tree	0.8536	0.6953*	0.7745*
H2-lin	**0.8652**	0.6958*	**0.7805**
H2-tree	0.8509	**0.6981**	0.7745*

Table 4: Experiments summary for 50 elements embedding vectors and rounding for Images set.

5.1 Experiments with Short Word Embeddings

First we conduct experiments using word embeddings consisting of 50 elements. The results for Images set are shown in Table 2. The first column contains the names of network architectures. “H1” indicates a network with one hidden layer. “H2” means that two hidden layers are used. “lin” denotes a linear architecture. “tree” indicates that a network constructed according to the dependency tree is used. Thus, “H2-lin” denotes a network with linear structure and two hidden layers. The second and third column contains results of F1 score and F1 score+type, respectively. The last column presents the average of F1 score and F1 score+type measures. The best results for each measure or average is marked in bold. * denotes values for which the difference between them and the best value, marked in bold, is not statistically significant at 0.05 p-value level. We use weighting for non-integer scores (please refer to Section 4).

In terms of F1 score measure, the best result (0.8651) is obtained with the most complicated network, i.e., the tree structured network with two hidden layers. However, the linear network with one hidden layer obtains the result that is not statistically different from the one marked in bold. For F1 score+type the best result (0.6894) is achieved by the simplest architecture, i.e., the linear network with one hidden layer. The same network gets the best average result (0.7757).

Comparing the statistically differences, only the linear network with two hidden layers yields the lowest results obtaining statistically lower results three times.

Table 6 presents the results for Headlines set. Considering statistic significance, the presented output of four networks does not differ for all measures, thus, all four networks perform well.

5.2 Experiments with Longer Word Embeddings

In the next set of experiments, we use longer word embedding vectors with 100 elements. Table 3 summarizes the results for Images set. In this setup, the best result for F1 score is obtained by the linear network with two hidden layers (0.8617). However, only H2-tree network yields statistically different re-

Model	F1 score	F1 s+type	Avg
H2-lin	**0.8652**	0.6958	**0.7805**
H2-tree	0.8509	**0.6981**	0.7745
NeRoSimR1 (Banjade et al., 2015)	0.7877	0.5841	0.6859
UMDuluthBlueTeam2 (Karumuri et al., 2015)	0.7968	0.5964	0.6966
FULL (Lopez-Gazpio et al., 2017)	0.8085	0.6159	0.7122

Table 5: Experiments summary for 50 elements embedding vectors and rounding for Images set.

Model	F1 score	F1 s+type	Avg
H1-lin	0.8632*	0.6615*	0.7623*
H1-tree	0.8648*	**0.6708**	**0.7678**
H2-lin	0.8662*	0.6651*	0.7657*
H2-tree	**0.8675**	0.6598*	0.7637*

Table 6: Experiments summary for 50 elements embedding vectors and Headlines set.

Model	F1 score	F1 s+type	Avg
H1-lin	0.8680*	0.6557*	0.7619*
H1-tree	**0.8700**	**0.6641**	**0.7670**
H2-lin	0.8679*	0.6543*	0.7611*
H2-tree	0.8679*	0.6514*	0.7597*

Table 7: Experiments summary for 100 elements embedding vectors and Headlines set.

sults from the one marked in bold.

For F1 score+type the network with two hidden layers and tree structure performs the best and achieves 0.6704. The average points also tree structured network but with one layer that yields 0.7611. The differences between results of different models for F1 score+type and average are in the range of around 1–1.5 percentage points (p.p.) and are not statistically different.

For Headlines (Table 7 once again, we obtain the results which are not statistically different.

Compared to 50 elements word embedding vectors and Images set, the networks trained with longer vectors perform better only for H1-tree and H2-lin in terms of F1 score measure. All averages are lower than results obtained for 50 elements word embedding vectors and Images set. For Headlines, the differences between averages for 50 and 100 elements vectors are so small that they are not statistically significant.

Model	F1 score	F1 s+type	Avg
H1-lin	**0.8734**	0.6783*	0.7758*
H1-tree	0.8680*	0.6702	0.7691
H2-lin	0.8694*	0.6747*	0.7720*
H2-tree	0.8707*	**0.6830**	**0.7769**

Table 8: Experiments summary for 50 elements embedding vectors and rounding for Headlines.

Thus, not only do longer vectors not improve the models but even lower the results, especially for F1 score+type measure. Hence, for further experiments we choose 50 elements word embedding vectors.

5.3 Non-Integer Scores

We also test the alternative approach to non-integer scores; that is, rounding (for details please refer to Section 4). The results are presented in Table 4. “H2-lin” achieves the best results for F1 score obtaining 0.8652. For F1 score+type and average the networks obtain the results that are not statistically different. These are the highest values for all set of experiments and measures for Images data set. Rounding increases especially F1 score+type measure, which is above 0.69 for all network architectures.

For Headlines data set (please refer to Table 8) only H1-tree is significantly worse than other networks for F1 score+type and average. Compared to previous results, this set of experiments obtains the best results for Headlines also.

The rounding approach resembles majority voting and reduces non-agreement between annotators. For scores close to integers, e.g., 4.75, one score of 4 may be a mistake or deviation from major score and it is reduced by rounding the value to 5. This kind of approach suggests the network the strict answer and reduces uncertainty about the golden score.

Model	F1 score	F1 s+type	Avg
H1-lin	**0.8734**	0.6783	0.7758
H2-tree	0.8707	**0.6830**	**0.7769**
NeRoSimR3 (Banjade et al., 2015)	0.8157	0.6426	0.7326
NeRoSimR2 (Banjade et al., 2015)	0.8263	0.6401	0.7332
FULL (Lopez-Gazpio et al., 2017)	0.8211	0.6185	0.7198

Table 9: Experiments summary for 50 elements embedding vectors and rounding for Headlines set.

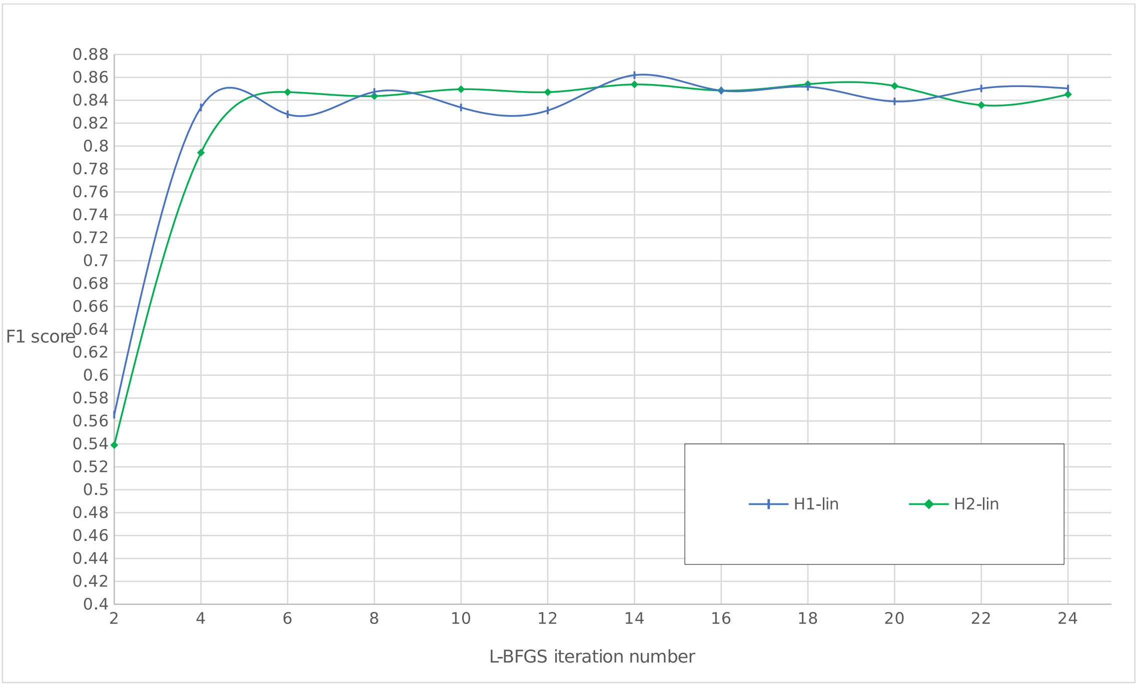

Figure 7: F1 score measure of linear Vanilla RNN with one and two hidden layers on Images set with respect to the number of iterations.

To sum up the experiments, our models assess semantic similarity quite well in iSTS. In this task, chunks are scored. Chunks usually are shorter, contain less tokens, than whole sentence, thus vanilla network with basic network cell may process it quite well as the relations within chunks are simpler than in whole sentences. This also could be the reason for networks based on parse tree not obtaining significantly better results than linear architectures. As the relations in chunks are simpler than in whole sentences additional information provided by parse tree does not help the system. Moreover, Vanilla Recurrent Neural Networks perform well in iSTS on small amount of data.

5.4 Comparison with Other iSTS Systems

As the networks trained on 50 elements word embedding vectors and rounding achieve the best results, we choose our best models to represent our approach.

We compare our models to systems from SemEval 2015 (Agirre et al., 2015) as there were changes introduced in SemEval 2016 and our models need to be adjusted. In SemEval 2015 assumed one-to-one alignments and SemEval 2016 introduced many-to-many alignments. We plan to do it in future work. We cannot compare results of systems prepared for basic semantic similarity task either as iSTS task is more complicated and systems designed for basic semantic similarity task cannot be used for iSTS task without further substantial modifications. The results of the comparison are shown in Table 5 and 9.

The system that obtained the best result for Images in SemEval 2015 iSTS for F1 score (0.7968) and F1 score+type (0.5964) measures was UMDuluth BlueTeam (Karumuri et al., 2015). NeRoSim (Banjade et al., 2015) ranked II for Images set, as it achieved 0.7877 for F1 score and 0.5964 for F1 score+type. System FULL (Lopez-Gazpio et al., 2017) outperforms two aforementioned systems. It is the solution prepared by a team that relates to organizers of SemEval 2015, thus the system could not be ranked. It scored 0.7122 in average measure.

Both our models outperform other systems in all measures for Images data set. Even all our remaining models achieve better results in terms of F1 score, F1 score+type, and average. The difference between our best model and the other best system is around 6 p.p. in F1 score, 8 p.p. in F1 score+type, and 7 p.p. in average, which is high improvement.

For Headlines our best models shown in Table 9 outperform significantly other systems also. The difference between our best model and the other best system is around 5 p.p. in F1 score, 3 p.p. in F1 score+type, and 4 p.p. in average. Moreover, our remaining models also yields better results the other systems.

The mentioned systems from SemEval 2015 used extensive feature engineering. In Interpretable Semantic Textual Similarity, NeRoSim applied hand-crafted rules. The system UMDuluth BlueTeam was a hybrid of human tuned word aligner, supervised machine learning and even translation systems. Our system achieves better results even though it does not use feature engineering nor customization performed by humans.

5.5 Iterations

Figure 7 presents the example F1 score measure with respect to number of L-BFGS algorithm iterations. We show linear networks with one and two hidden layers. 50 elements word embedding vectors are used and weighting for non-integer scores. Our models usually achieve high accuracy between 6th-8th iteration and remain stable for further iterations. The models with two hidden layers usually still yield lower F1 than models with one hidden layer at 4th iteration.

The basic network cell makes the process of training a model effective despite small number of available samples. It proves its usability in Interpretable Semantic Textual Similarity task.

6 Conclusions and Future Work

We presented three new architectures of recurrent neural networks built on top of basic network cell. Our aim was to maintain or increase the accuracy of established models by using Vanilla Recurrent Neural Networks and basic network cell. This approach was motivated by small amount of data available for Interpretable Semantic Textual Similarity (iSTS).

The proposed models are promising in iSTS as they achieved better results compared to the heavily hand-crafted systems. And there is still a lot of room for improvements in our models. We could also combine them with other hand-crated systems to obtain better results. However, we would like to stay away from feature engineering as much as we can, since this process is not automatic and requires manual work. In future work, we plan to check the accuracy of more complicated gates, such as GRU or LSTM, and test their influence on both accuracy and training time. Moreover, we would like to apply siamese networks to iSTS. We would also like to propose different network architectures tuned for Interpretable Semantic Textual Similarity task. We plan to adjust our system to modified iSTS SemEval 2016 task by introducing many-to-many alignments and test its accuracy.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In Cer et al. (Cer et al., 2015), pages 252–263.

Rajendra Banjade, Nobal Bikram Niraula, Nabin Maharjan, Vasile Rus, Dan Stefanescu, Mihai C. Lintean, and Dipesh Gautam. 2015. Nerosim: A system for measuring and interpreting semantic textual similarity. In Cer et al. (Cer et al., 2015), pages 164–171.

Daniel M. Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors. 2015. *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. The Association for Computer Linguistics.

Mona T. Diab, Timothy Baldwin, and Marco Baroni, editors. 2013. *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*. Association for Computational Linguistics.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc_ebiquity-core: Semantic textual similarity systems. In Diab et al. (Diab et al., 2013), pages 44–52.

Andrej Karpathy and Li Fei-Fei. 2017. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676.

Sakethram Karumuri, Viswanadh Kumar Reddy Vuggumudi, and Sai Charan Raj Chitirala. 2015. Umduluth-blueteam: SVCSTS - A multilingual and chunk level semantic similarity system. In Cer et al. (Cer et al., 2015), pages 107–110.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(1-3):503–528.

Iñigo Lopez-Gazpio, Montse Maritxalar, Aitor Gonzalez-Agirre, German Rigau, Larraitz Uria, and Eneko Agirre. 2017. Interpretable semantic textual similarity: Finding and explaining differences between sentences. *Knowl.-Based Syst.*, 119:186–199.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2786–2792. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak, and Piotr Andruszkiewicz. 2016. Samsung Poland NLP team at semeval-2016 task 1: Necessity for diversity; combining recursive autoencoders, wordnet and ensemble methods to measure semantic similarity. In Steven Bethard, Daniel M. Cer, Marine Carpuat, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 602–608. The Association for Computer Linguistics.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 801–809.

Hang Su and Haihua Xu. 2015. Multi-softmax deep neural network for semi-supervised training. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dres-*

den, Germany, September 6-10, 2015, pages 3239–3243. ISCA.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566. The Association for Computer Linguistics.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In Jan Hajic, Sandra Carberry, and Stephen Clark, editors, *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394. The Association for Computer Linguistics.