Lexical Simplification with the Deep Structured Similarity Model

Lis Pereira[1,3] [*] **Xiaodong Liu**[2] **John Lee**[3]
[1]Weathernews Inc., Nakase 1-3 Mihama-ku, Chiba-shi, 261-0023 Chiba, Japan
[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
[3] Department of Linguistics and Translation, City University of Hong Kong
kanash@wni.com, xiaodl@microsoft.com, jsylee@cityu.edu.hk

Abstract

We explore the application of a Deep Structured Similarity Model (DSSM) to ranking in lexical simplification. Our results show that the DSSM can effectively capture fine-grained features to perform semantic matching when ranking substitution candidates, outperforming the state-of-the-art on two standard datasets used for the task.

1 Introduction

Lexical simplification is the task of automatically rewriting a text by substituting words or phrases with simpler variants, while retaining its meaning and grammaticality. The goal is to make the text easier to understand for children, language learners, people with cognitive disabilities and even machines. Approaches to lexical simplification generally follow a standard pipeline consisting of two main steps: *generation* and *ranking*. In the generation step, a set of possible substitutions for the target word is commonly created by querying semantic databases such as Wordnet (Devlin and Tait, 1998), learning substitution rules from sentence-aligned parallel corpora of complex-simple texts (Horn et al., 2014; Paetzold and Specia, 2017), and learning word embeddings from a large corpora to obtain similar words of the complex word (Glavaš and Štajner, 2015; Kim et al., 2016; Paetzold and Specia, 2016a, 2017). In the ranking step, features that discriminate a substitution candidate from other substitution candidates are leveraged and the candidates are ranked with respect to their simplicity and contextual fitness.

The ranking step is challenging because the substitution candidates usually have similar meaning to the target word, and thus share similar context features. State-of-the-art approaches to ranking in lexical simplification exploit supervised machine learning-based methods that rely mostly on surface features, such as word frequency, word length and n-gram probability, for training the model (Horn et al., 2014; Bingel and Søgaard, 2016; Paetzold and Specia, 2016a, 2017). Moreover, deep architectures are not explored in these models. Surface features and shallow architectures might not be able to explore the features at different levels of abstractions that capture nuances that discriminate the candidates.

In this paper, we propose to use a Deep Structured Similarity Model (DSSM) (Huang et al., 2013) to rank substitution candidates. The DSSM exploits a deep architecture by using a deep neural network (DNN), that can effectively capture contextual features to perform semantic matching between two sentences. It has been successfully applied to several natural language processing (NLP) tasks, such as machine translation (Gao et al., 2014), web search ranking (Huang et al., 2013; Shen et al., 2014; Liu et al., 2015), question answering (Yih et al., 2014), and image captioning (Fang et al., 2015). To the best of our knowledge, this is the first time this model is applied to lexical simplification. We adapt the original DSSM architecture and objective function to our specific task. Our evaluation on two standard datasets for lexical simplification shows that this method outperforms state-of-the-art approaches that use supervised machine learning-based methods.

2 Method

2.1 Task Definition

We focus on the ranking step of the standard lexical simplification pipeline. Given a dataset of tar-

[*] This research was conducted while the first author was a Post Doctoral Fellow at the City University of Hong Kong.

get words, their sentential contexts and substitution candidates for the target words, the goal is to train a model that accurately ranks the candidates based on their simplicity and semantic matching.

For generating substitution candidates, we utilize the method proposed by Paetzold and Specia (2017), which was recently shown to be the state-of-art method for generating substitution candidates. They exploit a hybrid substitution generation approach where candidates are first extracted from 550,644 simple-complex aligned sentences from the Newsela corpus (Xu et al., 2015). Then, these candidates are complemented with candidates generated with a retrofitted word embedding model. The word embedding model is retrofitted over WordNet's synonym pairs (for details, please refer to Paetzold and Specia (2017)).

For ranking substitution candidates, we use a DSSM, which we elaborate in the next section.

2.2 DSSM for Ranking Substitution Candidates

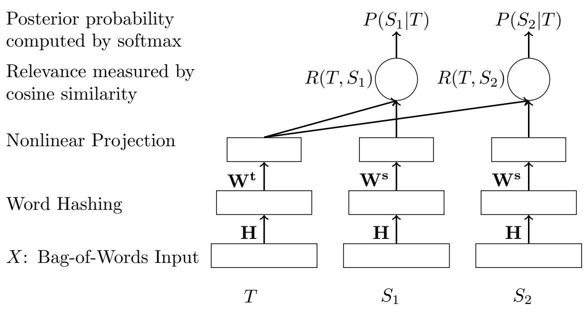

Figure 1: **Architecture of the Deep Structured Similarity Model (DSSM):** The input X (either a target word or a substitute candidate and their sentential contexts, T and S, respectively) is first represented as a bag of words, then hashed into letter 3-grams H. Non-linear projection W_t generates the semantic representation of T and non-linear projection W^s constructs the semantic representation of S. Finally, the cosine similarity is adopted to measure the relevance between the T and S. At last, the posterior probabilities over all candidates are computed.

Compared to other latent semantic models, such as Latent Semantic Analysis (Deerwester et al., 1990) and its extensions, Deep Structured Similarity Model (also called Deep Semantic Similarity Model) or DSSM (Huang et al., 2013) can capture fine-grained local and global contextual features more effectively. The DSSM is trained by optimizing a similarity-driven objective, by projecting the whole sentence to a continuous semantic space. In addition, it is is built upon characters (rather than words) for scalability and generalizability (He, 2016). Figure 1 shows the architecture of the model used in this work. It consists of a typical DNN with a word hashing layer, a nonlinear projection layer, and an output layer. Each component is described in the following:

Word Hashing Layer: the input is first mapped from a high-dimentional one-hot word vector into a low-dimentional letter-trigram space (with the dimentionality as low as 5k), a method called *word hashing* (Huang et al., 2013). For example, the word *cat* is hashed as the bag of letter trigram *#-c-a, c-a-t, a-t-#*, where # is a boundary symbol (Liu et al., 2015). The word hashing helps the model generalize better for out-of-vocabulary words and for spelling variants of the same word (Liu et al., 2015).

Nonlinear Projection Layer: This layers maps the substitution candidate and the target word in their sentential contexts, S and T respectively, which are represented as letter tri-grams, into d-dimension semantic representations, S^{S_q} and T^{S_q} respectively:

$$l = tanh(Wx) \tag{1}$$

where the nonlinear activation $tanh$ is defined as: $tanh(z) = \frac{1-e^{-2z}}{1+e^{-2z}}$.

Output Layer: This layer computes the semantic relevance score between S and T as:

$$R(T,S) = cosine(T^{S_q}, S^{S_q}) = \frac{{T^{S_q}}^T S^{S_q}}{\|T^{S_q}\|\|S^{S_q}\|} \tag{2}$$

2.3 Features for DSSM

As baseline features, we use the same n-gram probability features as in Paetzold and Specia (2017), who also employ a neural network to rank substitution candidates. As in Paetzold and Specia (2017), the features were extracted using the SubIMDB corpus (Paetzold and Specia, 2015). We also experiment with additional features that have been reported as useful in this task. For each target word and a substitution candidate word we also compute: cosine similarity, word length, and alignment probability in the sentence-aligned Normal-Simple Wikipedia corpus (Kauchak, 2013). The cosine similarity feature is computed using the SubIMDB corpus.

2.4 Implementation Details and Training Procedure of the DSSM

Following previous works that used supervised machine learning for ranking in lexical simplification (Horn et al., 2014; Paetzold and Specia, 2017), we train the DSSM using the LexMTurk dataset (Horn et al., 2014), which contains 500 instances composed of a sentence, a target word and substitution candidates ranked by simplicity (Paetzold and Specia, 2017). In order to learn the parameters W^t and W^s (Figure 1) of the DSSM, we use the standard backpropagation algorithm (Rumelhart et al., 1988). The objective used in this paper follows the pair-wise learning-to-rank paradigm outlined in (Burges et al., 2005).

Given a target word and its sentential context T, we obtain a list of candidates $\mathbf{L}$. We set different positive values to the candidates based on their simplicity rankings. E.g., if the list of the candidates is ordered by simplificity as, $L = \{A^+ > B^+ > C^+\}$, the labels are first constructed as $L = \{y_{A^+} = 3, y_{B^+} = 2, y_{C^+} = 1\}$. The values are then normalized by dividing by the maximum value in the list: $L = \{y_{A^+} = 1, y_{B^+} = 0.667, y_{C^+} = 0.333\}$. If the target word was not originally in $\mathbf{L}$, we add it with label 0. This enables the model to reflect the label information in the similarity scores. We minimize the Bayesian expected loss as: $\sum_{l=1}^{L} \ell(S_l, T)$, where $\ell(S_l, T)$ is defined as:

$$-\{y_l lnP(S_l|T) + (1 - y_l)ln(1 - P(S_l|T))\} \quad (3)$$

Note that $P(S_l|T)$ is computed as:

$$P(S_l|T) = \frac{exp(\gamma R(S_l, T))}{\sum_{S_i \in L} exp(\gamma R(S_i, T))} \quad (4)$$

here, γ is a tuning factor.

We used 5-cross validation approach to select hyper-parameters, such as number of hidden nodes. We set the gamma factor as 10 as per Huang et al. (2013). The selected hyper-parameters were used to train the model in the whole LexMTurk dataset. We employ early-stopping and select the model whose change of the average loss in each epoch was smaller than 1.0e-3. Since the training data is small (only 500 samples) we use a smaller number of hidden nodes, $d = 32$, in the nonlinear projection layer and adopt a higher dropout rate (0.4). The model is optimized using Adam (Kingma and Ba, 2014) with the learning rate fixed at 0.001, and is trained for 30 epochs. The mini-batch is set to 16 during training.

3 Experiments

3.1 Datasets and Evaluation Metrics

To evaluate the proposed model, we conduct experiments on two common datasets for lexical simplification: BenchLS (Paetzold and Specia, 2016b), which contains 929 instances, and NNSEval (Paetzold and Specia, 2016a), which contains 239 instances. Each instance is composed of a sentence, a target word, and a set of gold candidates ranked by simplicity (Paetzold and Specia, 2017). Since both datasets contain instances from the LexMturk dataset (Horn et al., 2014), which we use for training the DNN, we remove the overlap instances between training and test datasets [1]. We finally obtain 429 remaining instances in the BenchLS dataset, and 78 instances in the NNEval dataset, which are used in our evaluation.

We adopt the same evaluation metrics featured in Glavaš and Štajner (2015) and Horn et al. (2014): 1) **precision**: ratio of correct simplifications out of all the simplifications made by the system; 2) **accuracy**: ratio of correct simplifications out of all words that should have been simplified; and 3) **changed**: ratio of target words changed by the system.

3.2 Baselines

We compared the proposed model (DSSM Ranking) to two state-of-the-art approaches to ranking in lexical simplification that exploit supervised machine learning-based methods. The first baseline is the Neural Substitution Ranking (NSR) approach described in (Paetzold and Specia, 2017), which employs a multi-layer perceptron neural network. We reimplement their model as part of the LEXenstein toolkit (Paetzold and Specia, 2015). The network has 3 hidden layers with 8 nodes each. Unlike the proposed model, they treat ranking in lexical simplification as a standard classification problem. The second baseline is SVM*rank* (Joachims, 2006) (linear kernel

[1]Running on the original test datasets leads our system to obtain higher results than the ones reported here. Therefore, in order to avoid bias, we removed the overlap instances from both test datasets.

Substitution Candidates Ranking	Features	**BenchLS**			**NNSEval**		
		Prec.	Acc.	Changed	Prec.	Acc.	Changed
NSR	n-gram probs.	0.313	0.233	0.743	0.153	0.102	0.666
SVM^{rank}	n-gram probs.	0.354	0.261	0.736	0.166	0.102	0.615
DSSM Ranking	n-gram probs.	0.337	0.284	0.841	0.216	0.166	0.769
DSSM Ranking	all	**0.375**	**0.319**	**0.850**	**0.306**	**0.243**	**0.794**
Selection Step + Substitution Candidates Ranking	Features	**BenchLS**			**NNSEval**		
		Prec.	Acc.	Changed	Prec.	Acc.	Changed
NSR	n-gram probs.	0.304	0.214	0.703	0.204	0.102	0.500
SVM^{rank}	n-gram probs.	0.357	0.263	0.736	0.187	0.115	0.615
DSSM Ranking	n-gram probs.	0.355	0.286	0.806	0.259	0.179	0.692
DSSM Ranking	all	**0.383**	**0.328**	**0.857**	**0.333**	**0.269**	**0.807**

Table 1: Substitution candidates ranking results. *n-gram probs.* denotes the n-gram probability features described in Paetzold and Specia (2017), and *all* denotes all features described in Section 2.3. All values marked in bold are significantly higher compared to the best baseline, SVM^{rank}, measured by t-test at p-value of 0.05.

with default parameters) for ranking substitution candidates, similar to the method described in (Horn et al., 2014). All the three models employ the n-gram probability features extracted from the SubIMDB corpus (Paetzold and Specia, 2015), as described in (Paetzold and Specia, 2017), and are trained using the LexMTurk dataset.

3.3 Results

The top part of table 1 (Substitution Candidates Ranking) summarizes the results of all three systems. Overall, both SVM^{rank} and DSSM Ranking outperform the NSR Baseline. The DSSM Ranking performs comparably to SVM^{rank} when using only n-gram probabilities as features, and consistently leverages all features described in Section 2.3, outperforming all systems in accuracy, precision and changed ratio. We experimented with adding all features described in Section 2.3 to the baselines as well, however, we obtained no improvements compared to using only n-gram probability features.

We also tried running all ranking systems on selected candidates that best replace the target word in the input sentence. We follow the Unsupervised Boundary Ranking Substitution Selection method described in Paetzold and Specia (2017), which ranks candidates according to how well they fit the context of the target word, and discards 50% of the worst ranking candidates. The bottom part of the table 1 (Selection Step + Substitution Candidates Ranking) summarizes the results of all ranking systems after performing the selection step on generated substitution candidates. We obtain similar tendency in the results, with the DSSM Ranking outperforming both baselines. The results indicate the advantage of using a deep architecture, and of building a semantic representation of the whole sentence on top of the characters. To illustrate by examples, Table 2 lists the top candidate ranked by the systems for different input sentences. In the examples, the DSSM Ranking correctly ranked a substitute for the target word, while the two baselines either left the target word unchanged, or ranked an incorrect substitute.

Input = ”things continued on an **informal**, personal basis, by phone, I [remained] close friends with two of them, but Izzat al Gazawi died last year.”	
NSR	informal
SVM^{rank}	cozy
DSSM Ranking	**casual**
Input = ”perhaps the effect of West Nile Virus is sufficient to extinguish endemic birds already **severely** stressed by habitat losses.”	
NSR	severely
SVM^{rank}	severely
DSSM Ranking	**seriously**

Table 2: Top candidate ranked by the systems for different input sentences. The word in bold is the word to be simplified. The word highlighted denotes a correct answer.

4 Conclusions

We presented an effective method for ranking in lexical simplification. We explored the application of a DSSM that builds a semantic representation of the whole sentence on top of characters. The DSSM can effectively capture fine-grained features to perform semantic matching when ranking substitution candidates, outperforming state-of-art approaches that use supervised machine learning to ranking in lexical simplification. For future work, we plan to examine and incorporate a larger feature set to the DSSM, as well as try other DSSM architectures, such as the Convolutional DSSM (C-DSSM) (Shen et al., 2014).

Acknowledgments

We thank Gustavo Paetzold and Kevin Duh for valuable help.

This work was supported by the Innovation and Technology Fund (Ref: ITS/132/15) of the Innovation and Technology Commission, the Government of the Hong Kong Special Administrative Region; and by CityU Internal Funds for ITF Projects (no. 9678104).

References

Joachim Bingel and Anders Søgaard. 2016. Text simplification as tree labeling. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 337.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*, pages 161–173.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1482.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *ACL*. Citeseer.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 63–68.

Xiaodong He. 2016. Deep learning for natural language processing machine learning. In *"Machine Learning Summer School"*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *ACL (2)*, pages 458–463.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *ACL (1)*, pages 1537–1546.

Yea-Seul Kim, Jessica Hullman, Matthew Burgess, and Eytan Adar. 2016. Simplescience: Lexical simplification of scientific terminology. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1066–1071.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921.

G.H. Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. In *EACL*, pages 34–40.

Gustavo Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. In *ACL (System Demonstrations)*, pages 85–90.

Gustavo H Paetzold and Lucia Specia. 2016a. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3761–3767. AAAI Press.

Gustavo Henrique Paetzold and Lucia Specia. 2016b. Benchmarking lexical simplification systems. *Proceedings of the 10th LREC*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. "learning semantic representations using convolutional neural networks for web search". In *"Proceedings of the 23rd International Conference on World Wide Web"*, pages "373–374". "ACM".

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL (2)*, pages 643–648. Citeseer.