PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 108 JUNE 2017 133–145

A Neural Network Architecture for Detecting Grammatical Errors in Statistical Machine Translation

Arda Tezcan, Véronique Hoste, Lieve Macken

LT³, Language and Translation Technology Team
Department of Translation, Interpreting and Communication
Ghent University

Abstract

In this paper we present a Neural Network (NN) architecture for detecting grammatical errors in Statistical Machine Translation (SMT) using monolingual morpho-syntactic word representations in combination with surface and syntactic context windows. We test our approach on two language pairs and two tasks, namely detecting grammatical errors and predicting overall post-editing effort. Our results show that this approach is not only able to accurately detect grammatical errors but it also performs well as a quality estimation system for predicting overall post-editing effort, which is characterised by all types of MT errors. Furthermore, we show that this approach is portable to other languages.

1. Introduction

Despite the recent improvements in machine translation (MT), the task of producing grammatically correct sentences remains challenging for MT systems and post-editing is still necessary to obtain high quality translations. Furthermore, Statistical Machine Translation (SMT) systems seem to suffer more from grammatical errors than Neural Machine Translation (NMT) systems (Bentivogli et al., 2016), which pushed ahead the state of the art and challenged the dominance of SMT systems in recent (Bojar et al., 2016). The accurate detection of grammatical errors at the word level can be used as a major component for estimating the quality and post-editing effort in machine-translated texts. Moreover, such systems can assist post-editors by high-

Corresponding author: arda.tezcan@ugent.be

Cite as: Arda Tezcan, Véronique Hoste, Lieve Macken. A Neural Network Architecture for Detecting Grammatical Errors in Statistical Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 108, 2017, pp. 133–145. doi: 10.1515/pralin-2017-0015.

lighting errors, can inform MT developers about the strengths and/or weaknesses of MT systems and can further be developed as Automatic Post-Editing (APE) systems.

In this paper we present a Recurrent Neural Network (RNN) architecture for word-level detection of grammatical errors in SMT output by using word vectors that represent the *PoS*, *morphology* and *dependency relation* of words within surface and syntactic context windows. We test this approach for the English-Dutch (EN-NL) language pair and show that it can be used to detect grammatical errors with high accuracy, even when a relatively small data set is provided. Our results also indicate that, to detect grammatical errors in MT output, morpho-syntactic word representations are more informative than word embeddings, which capture precise syntactic and semantic word relationships (Mikolov et al., 2013). Furthermore, we apply this approach to predict overall post-editing effort for the EN-NL and English-German (EN-DE) language pairs by only relying on monolingual morpho-syntactic word representations, which do not provide any information about the semantic properties of words.

2. Related Work

Quality Estimation (QE) is the task of providing a quality indicator for machine-translated text without relying on reference translations (Gandrabur and Foster, 2003). Word-level QE, which can identify and locate problematic text fragments within a given MT output, has gained more attention in recent years (Bojar et al., 2016).

The detection of grammatical errors in MT output, without relying on reference translations, can be considered as a QE task that caters for a particular MT error type. Stymne and Ahrenberg (2010) used a rule-based grammar checker to assess and post-edit grammatical errors of their English-Swedish SMT system. Ma and McKeown (2012) decomposed parse trees of Chinese-English MT output into elementary trees and reconstructed the original parse trees using attribute value matrices that define the syntactic usage of each node in each tree. They considered reconstruction failures as indicators of grammatical errors. Recently, Tezcan et al. (2016) obtained dependency parse trees on English-Dutch MT output and queried the sub-trees of each parse tree against a treebank of correct sentences in the target language. The number of matching constructions were then used to mark words as grammatically incorrect.

Neural Networks (NNs) have been applied to many tasks in the Natural Language Processing (NLP) community, with language modelling (Bengio et al., 2003) and MT (Bahdanau et al., 2014) being two examples. In recent years, NNs have also shown promising results for sentence and word-level QE in different languages and domains (Kreutzer et al., 2015; Patel and Sasikumar, 2016). Moreover, focusing on the detection of grammatical errors from a different perspective, Liu and Liu (2016) proposed to derive positive and negative samples from unlabelled data, by generating grammatical errors artificially, and showed that RNNs outperform Support Vector Machines (SVMs) in judging the grammaticality of each word. All these NN-based systems utilized distributed word embeddings within context windows that preserve the original word sequence of given texts.

3. Morpho-syntactic Word Representations

Our assumption is that syntactic, morphological and dependency-related information about words provides useful information for detecting grammatical errors made by MT systems. Therefore, we have transformed each word in a given MT output into a feature vector using *multi-hot encoding*, which represents three types of information at the same time: *PoS*, *morphology* and *dependency relation*. These binary vectors are the same length as the size of the total vocabulary of all three types of information. In each word vector, all elements are assigned the value of 0, except the elements representing the linguistic features of each word, which are assigned 1. As a result, in this representation, each word is accurately represented with respect to its morpho-syntactic features, while avoiding the data sparsity issue, given the small vocabulary sizes of *PoS*, *morphology* and *dependency* labels.

Another approach to word representations is learning a distributed representation (word embeddings) for each word, which is dense and real-valued. Each dimension in distributed word representations, which are called word embeddings, represent a latent feature of a word, hopefully capturing useful syntactic and semantic properties (Turian et al., 2010). Unlike word embeddings, the morpho-syntactic representation strips out semantic features from words, which can be considered as unnecessary information for the task of detecting grammatical errors in MT output. Figure 1 shows an example source sentence (EN), its machine-translated version (NL) and the morpho-syntactic representation for the word *'zijn (are)'*. The MT output in this figure contains a grammatical error in the form of subject-verb agreement in number between the words *'zijn (are)'* (plural) and *'kans (chance)'* (singular). We obtain the morpho-syntactic features for Dutch using the Alpino parser (Van Noord, 2006).

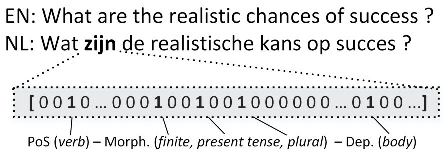
EN: What are the realistic chances of success ?
NL: Wat **zijn** de realistische kans op succes ?
[0 0 **1** 0 … 0 0 0 **1** 0 0 **1** 0 0 **1** 0 0 0 0 0 0 … 0 **1** 0 0 …]
PoS (*verb*) – Morph. (*finite, present tense, plural*) – Dep. (*body*)

Figure 1. Binary vector for 'zijn (are)' consisting of 1s for its PoS, morphology and dependency features and 0s for the remaining items in the vocabulary.

4. Surface and Syntactic N-Grams

Surface n-grams are sequences of words as they appear in texts, with n corresponding to the number of words in the sequence. While surface n-grams have been used effectively in various types of NLP tasks, they primarily rely on local context and are not informative on a syntactic level. Dependency trees, on the other hand, represent words in a sentence as nodes and grammatical relations between the words as edges. Unlike the surface n-grams, syntactic n-grams, which can be constructed by using paths in dependency trees, offer context windows based on syntactic neighbours of words and are able to capture long-distance dependencies.

Given each target word in the MT output, we consider four different fixed-sized context windows, which are based on the following surface and syntactic n-grams:

Surface n-gram (n): Sequence of words as they appear in MT output, centered around the target word (n=5)

Syntactic n-grams (sn):

- **Parents (sn_p)**: Vertical sequence of parent nodes in a given dependency tree for a given target node (n=3)
- **Siblings (sn_s)**: Horizontal sequence of sibling nodes sharing the same parent in a given dependency tree, centered around the target node (n=5)
- **Children (sn_c)**: Sequence of children nodes for a given target node (depth 1), containing the target node in the centre (n=5)

We include additional placeholder tokens in the vocabulary of the morpho-syntactic features to indicate boundaries (namely '<s>' to indicate a sentence boundary, '[ROOT]' to indicate the root of the dependency tree and '[NA]' to indicate horizontal boundaries in the sub-trees). Moreover, we preserve the original word order in each syntactic n-gram, with the aim of capturing word ordering errors in machine-translated texts. The n values for the four n-gram types are chosen as the best values between 3 up to 5, which maximized the estimation performance when all n-gram types are used together[1]. The four different context windows extracted for the word '*zijn (are)*' are illustrated in Figure 2.

One difficulty of using dependency parsers on MT output is that the syntactic relationships between words can only be accurately captured provided that a correct dependency parse tree is obtained to start with. This can be illustrated in the example in Figures 1 and 2. In this example, the surface 5-gram context window is unable to capture the dependency relation between the two words generating the grammatical error: '*zijn (are)*'[2] (plural) and '*kans (chance)*' (singular). While the syntactic n-gram (children) is able to capture it, the disagreement cannot be directly observed

[1]The 99-percentile for the number of parents (up to the root), siblings and children over all tokens in the dependency trees generated for the EN-NL data set are observed as 10, 4 and 4, respectively.

[2]In the example given in Figures 1 and 2, the dependency label *body* refers to the body of a verbal projection within a WH-phrase, headed by the word 'wat', which is marked as *ROOT*. Detailed information (in Dutch) about the syntactic annotations used by the Alpino parser can be found at `http://www.let.rug.nl/vannoord/Lassy/sa-man_lassy.pdf`

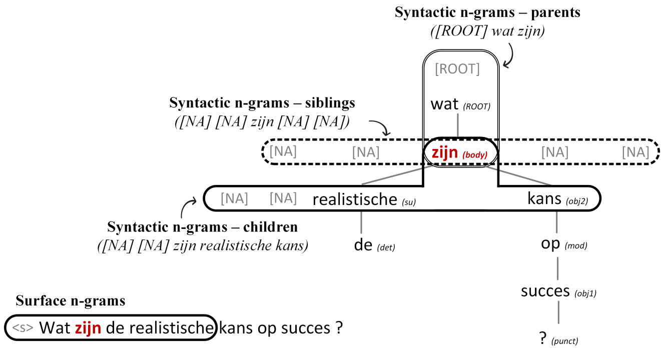
Syntactic n-grams – parents
([ROOT] wat zijn)
[ROOT]
wat (ROOT)
Syntactic n-grams – siblings
([NA] [NA] zijn [NA] [NA])
[NA] [NA] zijn (body) [NA] [NA]
[NA] [NA] realistische (su) kans (obj2)
Syntactic n-grams – children
([NA] [NA] zijn realistische kans)
de (det)
op (mod)
succes (obj1)
? (punct)
Surface n-grams
<s> Wat zijn de realistische kans op succes ?

Figure 2. A machine-translated sentence (lower left), its dependency parse tree (upper right) and the four different context windows used for the target word 'zijn (are)'.

in the parse tree since the parser (incorrectly) labels '*realistische (realistic)*' as the subject of the sentence. Tezcan et al. (2016) show that dependency parsers can nevertheless be useful to detect grammatical errors due to the unusual dependency structures they produce on MT output that contains errors. Similarly, our motivation for using syntactic n-grams is to learn such unusual structures by exploiting morpho-syntactic word representations in combination with dependency structures.

5. Neural Network Architecture

We propose a neural network architecture that uses Gated Recurrent Units (GRUs) (Cho et al., 2014). Similar to Long Short Term-Memory (LSTM)(Hochreiter and Schmidhuber, 1997), GRU is a variant of RNNs that are well suited to learn from history to process time series. Despite their similarities, LSTMs and GRUs have been shown to outperform each other in particular NLP tasks. LSTMs, for example, seem to be a better approach for language modelling (Irie et al., 2016). GRUs, on the other hand, have been shown to perform better in the task of word-level quality estimation of machine translation (Patel and Sasikumar, 2016).

We provide four different context vectors (as described in Section 4) as inputs to four GRU layers, which are concatenated before they are connected to the output layer, which consists of two units. The softmax over the activation of these two units is taken as a score for the two classes OK and BAD, which represent the correct and erroneous words, respectively. To reduce overfitting, we apply dropout (Srivastava et al., 2014) within the GRU layers (for the input gates and the recurrent connections) and after the concatenated hidden units. Figure 3 illustrates the proposed NN architecture.

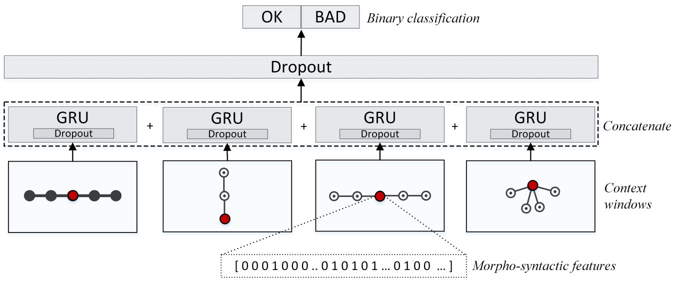

Figure 3. The proposed neural network architecture.

In all of our experiments, we have used binary cross-entropy as loss function and RMSProp (Tieleman and Hinton, 2012) as optimiser. We fixed the mini-batch size to 50 and trained each model for 50 epochs[3]. We implemented all models using the TensorFlow framework (Abadi et al., 2016). We adjusted the sizes of the GRU layers according to the sizes of the two data sets we used in our experiments, which are detailed in Section 6.

6. Experiments

We evaluated the proposed method on two tasks: detecting grammatical errors and predicting post-editing effort in SMT. In the first experiment we evaluate the performance of the proposed method on detecting grammatical errors for EN-NL. The second experiment aimed to find out if the same method could successfully be applied to a different language pair (EN-DE) and whether it could be used to predict overall post-editing effort. In both experiments, we considered F1_MULTI as the primary evaluation metric, which is the multiplication of F1 scores for the OK and BAD classes.

6.1. Detecting Grammatical Errors

To evaluate the performance of the proposed method in the task of detecting grammatical errors, we used the SCATE corpus of SMT errors (Tezcan et al., in press), which consists of 2967 sentence pairs. The source sentences in this data set were extracted from three different text types of the Dutch Parallel Corpus (Macken et al., 2011). The translations in this data set were obtained from Google Translate[4]. This corpus contains manual error annotations, which are classified based on the distinction between

[3]The mini-batch size of 50 has been selected as the best value after training the system with different sizes from 25 to 'full batch'. In all out experiments, each network converged to a point of minimal error after 40 epochs.

[4]http://translate.google.com (June, 2014)

fluency and accuracy by referring to the type of information that is needed to detect them. According to this taxonomy, fluency errors are detected on the target text alone (monolingual level), while to detect both the source and target text need to be analyzed (bilingual level). The fluency errors are further divided into the following sub-categories: *grammar*, *lexicon*, *orthography*, *multiple errors* and *other fluency errors*. To evaluate the proposed method we used the annotations of *grammar* and *multiple errors*. The label *multiple errors* was used when different fluency errors occurred at the same time, e.g. a word order combined with a wrong lexical choice. It is safe to say that most of the words labelled as *multiple errors* contain grammatical problems. As a result, the data set that we used in this experiment consisted of 58002 words, with an OK to BAD ratio of approx. 3.4:1. All systems in this experiment were evaluated using the average 10-fold cross-validation scores. To handle the issue of skewed distribution of labels, during training, we assigned class weights that are inversely proportional to their frequency in each training fold. For the EN-NL experiments, we trained the NN systems with GRU layer sizes of 50. We used the Alpino parser to extract the morpho-syntactic features for each word in a given MT output. The resulting word vectors consist of 128 features.

In the first part of this experiment we compared the proposed NN architecture (NN-MS) and the impact of using different morpho-syntactic features in this architecture to a baseline system proposed by Tezcan et al. (2016). The baseline system is based on querying subtrees of the dependency trees obtained on the MT output against a treebank of dependency trees built from correct sentences[5]. Considering their ability to capture syntactic and semantic properties of words, we also compare the effectiveness of word embeddings (NN-*word2vec*) to morpho-syntactic features as alternative word representations. For this purpose, we pre-trained 200-dimensional *word2vec* word embeddings (Mikolov et al., 2013) using 328M words from the SoNaR corpus (Oostdijk et al., 2008)[6]. In this experiment, we evaluated all NN systems using the surface 5-gram context windows (n).

The results in Table 1 clearly show that the NN architectures perform better than a simple frequency-based method (Baseline). We see that using only PoS features in the NN architecture is enough to beat this baseline system. Moreover, introducing additional morpho-syntactic features further improves the system. The positive effect of using dependency labels supports our hypothesis that they provide useful information for learning grammatical errors, even though the parser makes mistakes (as shown in Figure 2). Finally, we see that the performance of this NN architecture drastically improves when all three morpho-syntactic features are used instead

[5]Even though this system is evaluated on a subset of the data set used in this paper, it can safely be compared to the proposed method, given that it uses the same annotation set (*grammar* and *multiple errors*) and the assumption that it would achieve similar results on a larger test set because it is not based on machine-learning methods.

[6]We replace singleton words in the training data with <*unk*> to handle unknown words and apply zero padding to the n-grams containing sentence boundaries

	F1_BAD	F1_OK	F1_MULTI
Baseline (Tezcan et al., 2016)	0.3811	0.6789	0.2587
NN-MS - PoS (n)	0.4343	0.7493	0.3253
NN-MS - PoS+Morph (n)	0.4561	0.7951	0.3626
NN-MS - PoS+Morph+Dep (n)	**0.4729**	**0.8138**	**0.3848**
NN-*word2vec* (n)	0.4110	0.7779	0.3204

Table 1. Performance of the baseline system and the NN systems using different word representations.

of word embeddings. This observation suggests that the semantic and syntactic relationships captured by word embeddings are not as informative as the proposed morpho-syntactic features for this task.

In the second part of this experiment, we analyzed the predictive power of the surface and syntactic n-grams as context windows. Table 2 provides an overview of the performance of the different systems using the same three morpho-syntactic features with different combinations of context windows.

	F1_BAD	F1_OK	F1_MULTI
NN-MS (n)	0.4729	0.8138	0.3848
NN-MS (sn_p)	0.4053	0.7806	0.3162
NN-MS (sn_s)	0.4079	0.7861	0.3255
NN-MS (sn_c)	0.4077	0.7865	0.3203
NN-MS ($n + sn_p + sn_s + sn_c$)	**0.4799**	**0.8338**	**0.3998**
NN-MS ($sn_p + sn_s + sn_c$)	0.4383	0.8135	0.3565

Table 2. Performances of the NN systems using the three morpho-syntactic features with different combinations of context windows.

As is evident from the results of the four different context windows in isolation, the surface n-gram context window provides the most useful information when used alone. The syntactic n-grams seem to contain extra useful information and maximize the performance of the system when they are used in combination with the surface n-gram windows. Furthermore, removing a specific type of context window from the combined set reduces the performance in all cases. The largest drop in performance occurs when the surface n-grams are removed, which confirms the usefulness of the information provided by this context window.

6.2. Predicting Post-editing Effort

Applying the proposed method to a different language requires the use of a different set of language-specific NLP tools and/or models. To compare the performance over different languages, we applied the proposed method to predict post-editing effort to two different language pairs, namely to EN-NL and EN-DE. For EN-DE we tested this method on the WMT'16 data set, which has been used in the shared task on word-level QE. This data set consists of 15K source-target sentence pairs (279976 words in the target language) in the IT-domain with target sentences being the machine-translated version of the source sentences by a phrase-based SMT system. The data was partitioned into 12K, 1K and 2K sentence pairs as training, tuning and test sets, respectively. All words in this data set have been automatically annotated for errors with binary word-labels (OK and BAD) using the alignments between the MT output and its post-edited version provided by the TER tool[7] (Snover et al., 2006). In all three data sets, the OK to BAD ratio is approx. 4:1. Prior to training the NN, we obtained PoS, morphology and dependency labels for each German word in the MT output (in CoNLL-U format), using the Mate tools (Bohnet and Nivre, 2012). The resulting word vectors consist of 127 features. During training, we assigned class weights that are inversely proportional to their frequencies in the training set. For the EN-DE experiments, we trained the NN system with GRU layer sizes of 100 (instead of 50) and increased the complexity of the NN, given the relatively larger data set compared to the EN-NL language pair. For the EN-NL language pair, we used the same NN architecture and the data set as detailed in Section 6.1 with one difference: instead of using the gold-standard error annotations for grammatical errors, for this experiment, we automatically annotated the words for errors using the same procedure in the shared task of QE (WMT'16), by using the TER tool. For this purpose, we used the post-edited version of the MT output from a Master student in translation studies.

We evaluated the EN-NL system with regard to the average cross-validation results. The evaluation of the EN-DE system, on the other hand, was conducted on the test set made available by the organizers. This approach allows us to additionally compare the performance of the EN-DE system with the competing systems in the shared task. We trained both systems using the morpho-syntactic features consisting of *PoS*, *morphology* and *dependency* features and the four context windows consisting of surface and syntactic n-grams. Table 3 provides an overview of the performance of the proposed method for the two language pairs.

	F1_BAD	**F1_OK**	**F1_MULTI**
EN-NL (SCATE) - avg. cross val.	0.4335	0.8649	0.3749
EN-DE (WMT'16) - held out test set	0.4224	0.8319	0.3514

Table 3. Performance of the NN systems for predicting post-editing effort.

[7] The settings used are: tokenized, case insensitive, exact matching only. *Deletions* are not annotated as they cannot be associated with any word and *shifts* are disabled, but rather annotated as edits in the form of *deletions* and *insertions*.

From Table 3, we can see that, despite the difference between the data sizes and the tools we used, both systems obtained similar results. Furthermore, by comparing the results obtained for the EN-NL system on the two tasks, we can see that the proposed method performs better on detecting grammatical errors (F1_MULTI = 0.3998, as provided in Figure 2) than predicting overall post-editing effort (F1_MULTI = 0.3749), which represents all types of MT errors. We can gain a better picture of the performance of the proposed method on predicting post-editing effort when we compare the EN-DE system with the systems that participated in the shared task of word level QE in WMT'16 (Bojar et al., 2016). Three of these systems (out of 14) are provided in Table 4.

	Rank	**F1_BAD**	**F1_OK**	**F1_MULTI**
UNBABEL/ensemble	1	0.5599	0.8845	0.4952
CDACM/RNN	8	0.4192	0.8421	0.3531
EN-DE (WMT'16)	-	0.4224	0.8319	0.3514
BASELINE	11	0.3682	0.8800	0.3240

Table 4. Performances of the proposed NN architecture in comparison to three competing systems (and the ranks they achieved) in WMT'16 shared task on QE.

The proposed system outperforms the baseline system used in this shared task, consisting of 22 features representing monolingual and bilingual properties of each translated text. Moreover, it performs slightly worse than another GRU-based NN system (CDACM/RNN), which uses *word2vec* word embeddings within monolingual context windows of surface n-grams (Patel and Sasikumar, 2016). This observation shows that the morpho-syntactic features can provide almost as useful information as word embeddings for learning overall post-editing effort.

7. Conclusion

We have proposed an RNN architecture for word-level detection of grammatical errors in SMT that utilizes monolingual features in context windows of surface and syntactic n-grams. Our approach relies on PoS, morphological and dependency information of the MT output and uses multi-hot encoding to represent the morpho-syntactic properties of words as word vectors. We showed that this approach achieves high performance on EN-NL SMT output, even when a relatively small training set is available. Moreover, our results suggest that word embeddings, despite their informativeness on syntactic and semantic properties of words, should not be considered as a one-size-fits-all approach in the QE task. For detecting grammatical errors in SMT output, we achieved a marked improvement in performance by using accurate morpho-syntactic features over word embeddings. By applying the proposed approach on the task of predicting post-editing effort, we demonstrated its ability to learn all MT error types on two language pairs, EN-NL and EN-DE. This observation shows the applicability of the proposed method across languages and reveals the amount of valuable monolingual information that can be employed for estimating overall quality in machine-translated texts.

Building separate error-detection systems that are trained on different types of MT errors can be considered as an alternative approach to existing QE systems, which try to make a direct estimation of overall quality. By combining such specialized systems, we would like to build a single QE system that does not only achieve high performance on the QE task, but can be informative about the reasons of the estimated quality and the types and the location of errors MT systems make. We would also like to adapt this approach with a view to detecting common errors in NMT systems, which seem to make fewer grammatical errors compared to SMT systems.

Acknowledgements

This research has been carried out in the framework of the SCATE project funded by the Flemish government agency IWT (IWT-SBO 130041).

Bibliography

Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT@EMNLP 2015, 17-18 September 2015, Lisbon, Portugal, 2015. The Association for Computer Linguistics. ISBN 978-1-941643-32-7. URL `http://aclweb.org/anthology/W/W15/`.

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR*, abs/1603.04467, 2016.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003. ISSN 1532-4435.

Bentivogli, Luisa, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus Phrase-Based Machine Translation Quality: a Case Study. *CoRR*, abs/1608.04631, 2016.

Bohnet, Bernd and Joakim Nivre. A Transition-based System for Joint Part-of-speech Tagging and Labeled Non-projective Dependency Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1455–1465. Association for Computational Linguistics, 2012.

Bojar, Ondrej, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, et al. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 131–198, 2016.

Cho, Kyunghyun, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

Gandrabur, Simona and George Foster. Confidence Estimation for Translation Prediction. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 95–102. Association for Computational Linguistics, 2003.

Hochreiter, Sepp and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, 1997.

Irie, Kazuki, Zoltán Tüske, Tamer Alkhouli, Ralf Schlüter, and Hermann Ney. LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition. In Morgan (2016), pages 3519–3523. doi: 10.21437/Interspeech.2016. URL `http://dx.doi.org/10.21437/Interspeech.2016`.

Kreutzer, Julia, Shigehiko Schamoni, and Stefan Riezler. QUality Estimation from ScraTCH (QUETCH): Deep Learning for Word-level Translation Quality Estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT@EMNLP 2015, 17-18 September 2015, Lisbon, Portugal* DBL (2015), pages 316–322. ISBN 978-1-941643-32-7. URL `http://aclweb.org/anthology/W/W15/`.

Liu, Zhuoran and Yang Liu. Exploiting Unlabeled Data for Neural Grammatical Error Detection. *CoRR*, abs/1611.08987, 2016.

Ma, Wei-Yun and Kathleen McKeown. Detecting and Correcting Syntactic Errors in Machine Translation Using Feature-Based Lexicalized Tree Adjoining Grammars. *IJCLCLP*, 17(4), 2012.

Macken, Lieve, Orphée De Clercq, and Hans Paulussen. Dutch parallel corpus: a balanced copyright-cleared parallel corpus. *Meta: Journal des traducteursMeta:/Translators' Journal*, 56 (2):374–390, 2011.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

Morgan, Nelson, editor. *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, 2016. ISCA. doi: 10.21437/Interspeech.2016. URL `http://dx.doi.org/10.21437/Interspeech.2016`.

Oostdijk, N., M. Reynaert, P. Monachesi, G. Van Noord, R. Ordelman, and I. Schuurman. From DCoi to SoNaR: a reference corpus for Dutch. In *In Proceedings of the Sixth International Conference on Language Resources and Evaluation*, 2008.

Patel, Raj Nath and M. Sasikumar. Translation Quality Estimation using Recurrent Neural Network. *CoRR*, abs/1610.04841, 2016.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

Stymne, Sara and Lars Ahrenberg. Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), 2010. ISBN 2-9517408-6-7.

Tezcan, Arda, Véronique Hoste, and Lieve Macken. Detecting grammatical errors in machine translation output using dependency parsing and treebank querying. *Baltic Journal of Modern Computing*, 4(2):203–217, 2016.

Tezcan, Arda, Véronique Hoste, and Lieve Macken. SCATE Taxonomy and Corpus of Machine Translation Errors. In *Trends in e-tools and resources for translators and interpreters*. Brill, in press.

Tieleman, T. and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio. Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394. Association for Computational Linguistics, 2010.

Van Noord, Gertjan. At last parsing is now operational. In *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, pages 20–42, 2006.

Address for correspondence:
Arda Tezcan
`arda.tezcan@ugent.be`
LT^3 Language and Translation Technology Team
Department of Translation, Interpreting and Communication
Faculty of Arts and Philosophy
Ghent University
Groot-Brittanniëlaan 45, B-9000 Ghent, Belgium