SimpleNER Sentence Simplification System for GEM 2021

K V Aditya Srivatsa **Monil Gokani** **Manish Shrivastava**
Language Technologies Research Center (LTRC)
Kohli Center on Intelligent Systems
International Institute of Information Technology, Hyderabad
{k.v.aditya, monil.gokani}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

This paper describes SimpleNER, a model developed for the sentence simplification task at GEM-2021. Our system is a monolingual Seq2Seq Transformer architecture that uses control tokens pre-pended to the data, allowing the model to shape the generated simplifications according to user desired attributes. Additionally, we show that NER-tagging the training data before use helps stabilize the effect of the control tokens and significantly improves the overall performance of the system. We also employ pretrained embeddings to reduce data sparsity and allow the model to produce more generalizable outputs.

1 Introduction

Sentence simplification aims at reducing the linguistic complexity of a given text, while preserving all the relevant details of the initial text. This is particularly useful for people with cognitive disabilities (Evans et al., 2014), as well as for second language learners and people with low-literacy levels (Watanabe et al., 2009). Text and Sentence simplification also play an important role within NLP. Simplification has been utilized as a preprocessing step in larger NLP pipelines, which can greatly aid learning by reducing vocabulary and regularizing of syntax.

In our model, we use control tokens to tune a Seq2Seq Transformer model (Vaswani et al., 2017) for sentence simplification. We take character length compression, extent of paraphrase, and lexical & syntactic complexity as attributes to gauge the transformations between complex and simple sentence pairs. We then represent each of these attributes as numerical measures, which are then added to our data. We show that this provides a considerable improvement over as-is Transformer approaches.

The use of control tokens in Seq2Seq models for sentence simplification has been explored before (Martin et al., 2020). But this approach has shown to add data sparsity to the system. This is because the model is required to learn the distribution of the various control tokens and the expected outputs across the ranges of each control token. To mitigate this sparsity, we process our data to replace named entities with respective tags using an NER tagger. We show that this reduces the model vocabulary and allows for greater generalization. To further curb the data sparsity, we make use of pre-trained embeddings as initial input embeddings for model training. Our code is publicly available here. [1]

2 Background

2.1 Sentence Simplification

Past approaches towards sentence simplification have dealt with it as a monolingual machine translation(MT) task (specifically Seq2Seq MT (Sutskever et al., 2014)). This meant training MT architectures over complex-simple sentence pairs, either aligned manually (Alva-Manchego et al., 2020; Xu et al., 2016) or automatically (Zhu et al., 2010; Wubben et al., 2012) using large complex-simple repository pairs such as the English Wikipedia and the Simple English Wikipedia.

Some implementations also utilize reinforcement learning (Zhang and Lapata, 2017) over the MT task, with automated metrics such as SARI (Xu et al., 2016), information preservation, and grammatical fluency constituting the training reward.

2.2 Controllable Text Generation

A recent approach towards sentence simplification involves using control tokens during machine translation (Martin et al., 2020). For simplification, it

[1] https://github.com/kvadityasrivatsa/gem_2021_simplification_task

Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021), pages 155–160

Control Attribute	Control Measure	Control Token
Amount of compression	Compression ratio	<NbChars_x.xx>
Paraphrasing	Levenshtein similarity	<LevSim_x.xx>
Lexical complexity	Avg. third-quartile of log-ranks	<WordRank_x.xx>
Syntactic complexity	Max dependency tree depth	<DepTreeDepth_x.xx>

Table 1: Control Tokens used for Modelling

encodes and enforces changes in certain attributes of the text. Similar approaches for controlling generated text have been explored in other domains: Filippova (2020) uses control tokens to estimate and control the amount of hallucination in generated text, Fan et al. (2018) explored pre-pending control tokens to the input text for summarization, providing control over the length of the output, and customizing text generation for different sources.

Our model makes use of control tokens similar to Martin et al. (2020) to tailor the generated simplifications according to the extent of changes in the following attributes: character length, extent of paraphrasing, and lexical & syntactic complexity. These attributes are represented by their respective numerical measures (see 3.1), and then pre-pended to the complex sentences using in specific formats (Table 1). Alongside this, we use NER tagging and pre-trained input embeddings as a method to curb data sparsity and unwanted named entity (NE) replacements.

3 System Overview

3.1 Control Attributes

Following Martin et al. (2020), we encode the following attributes during training and attempt to control them during inference time. Eg:

Complex: *"<NbChars_0.80> <LevSim_0.76> <WordRank_0.79> it is particularly famous for the cultivation of kiwifruit ."*

Simple: *"It is mostly famous for the growing of kiwifruit ."*

3.1.1 Amount of compression

Compression in sequence length has been shown to be correlated with the simplicity and readability of text (Martin et al., 2019). Since compression as an operation directly involves deletion, controlling its extent plays a crucial role in the extent of information preservation. We make use of the **compression ratio** (control token: '**NbChars**') between the character lengths of the simple and complex sentences to encode for this attribute.

3.1.2 Paraphrasing

The extent of paraphrasing between the complex and simple sentences ranges from a near replica of the source sentence to a very dissimilar and possibly simplified one. The measure used for this attribute is **Levenshtein similarity** (Levenshtein, 1966) (control token: '**LevSim**') between the complex and simple sentences.

3.1.3 Lexical Complexity

For a young reader or a second language learner, complex words can decrease the overall readability of the text substantially. The average **word rank** (control token: '**WordRank**') of a sequence has been shown to correlate with the lexical complexity of the sentence (Paetzold and Specia, 2016). Therefore, similar to Martin et al. (2020), we use the average of the third-quartile of log-ranks of the words in a sentence (except for stop-words and special tokens), to encode for its lexical complexity.

3.1.4 Syntactic Complexity

Complex syntactic structures and multiple nested clauses can decrease the readability of text, especially for people with reading disabilities. To partially account for this, we make use of the maximum **syntactic tree depth** (control token: '**DepTreeDepth**') of the sentence as a measure of its syntactic complexity. We use SpaCy's English dependency parser (Honnibal et al., 2020) to extract the depth. The deeper the syntax tree of a sentence, the more likely it is that it involves highly nested clausal structures.

3.2 NER Replacement

Using control tokens contribute to the overall performance of the model, but it also gives rise to an added data sparsity. It divides the sentences of the train set into different ranges of the control tokens. This results in some control values having little to no examples, which adds the task of learning and generalizing over the control token values for the model. Additionally, the model can learn to ad-

Raw (Complex)	*"Sergio PÃ rez Mendoza (born January 26 , 1990 in Guadalajara , Jalisco) , also known as "Checo" PÃ rez , is a Mexican racing driver ."*
NER Replaced	*"person@1 (born date@1 in gpe@1) , also known as " person@2 " , is a norp@1 racing driver ."*

Table 2: NER Tagging input sentence

here to the control requirement, while still failing to correctly simplify the sentence. Eg:

Source: *<NbChars_0.95> <LevSim_0.75> <WordRank_0.75> oxygen is a chemical element with symbol o and atomic number 8 .*

Prediction: *It has the chemical symbol o . It has the atomic number 8 .*

Here, the proper noun *"Oxygen"* is replaced by the pronoun *"it"*. Although the model follows the requirement of bringing down the word rank of the sentence and remains grammatically sound, it doesn't help with the simplification.

To address the issue of data sparsity as well that of unwanted NE-replacement, we propose NER mapping the data before training, and replacing the NE-tokens back after generation. We make use of the Ontonotes NER tagger (Yu et al., 2020) in the Flair toolkit (Akbik et al., 2019). We identify named entities in the complex halves of all three of the data splits and replace them with one of 18 tags (from the NER tagger) with a unique index (Table 2). NER replacement for simplification was previously explored by Zhang and Lapata (2017), but consisted of fewer classes. The large number of tags allow for a fine division between different named-entity types, which helps the model to encode the contexts of each of the types better while still reducing the NE-vocabulary size substantially.

The tagged data is then used for training and subsequent generation on the test set. Then any tags in the simplified output are located in the saved NER-mapping and reverted back to the original token or phrase. This step not only prevents proper nouns from getting replaced, but also greatly reduces the model vocabulary (allowing for greater generalizability).

3.3 Pre-Trained Embeddings

The vocabulary of a model trained on a corpus like WikiLarge is quite small, which prevents the model from predicting better fitting tokens. To address this, we use FastText's pre-trained embeddings (Bojanowski et al., 2016) (dimensionality: 300) as input embeddings for our model. The embeddings significantly boost the vocabulary size of usable content words for the model.

4 Experimental Setup

4.1 Architecture

Our architecture is a Transformer Model (Vaswani et al., 2017), and we make use of the Transformer Seq2Seq implementation from FairSeq (Ott et al., 2019). To understand the impact of each of the proposed methods, we train a total of four models:

- **T**: Vanilla Transformer (Vaswani et al., 2017), with control tokens, used as a baseline model.
- **T+Pre**: Transformer trained with FastText's pretrained embeddings.
- **T+NER**: Transformer trained on NER mapped data.
- **SimpleNER (T+Pre+NER)**: Transformer trained on NER mapped data with FastText's pretrained embeddings.

For ease of comparison, all four models were trained with an input embedding dimensionality of 300, fully connected layers with a dimensionality of 2048, 6 layers and 6 attention heads on both, the encoder and the decoder. During training , we are using Adam optimizer (Kingma and Ba, 2015) ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), with a learning rate of 0.00011 and 4000 warm-up updates, while dropout is set at 0.2.

4.2 Datasets

For training, we make use of the WikiLarge dataset (Zhang and Lapata, 2017), with 296,402 automatically aligned complex-simple sentence pairs obtained from the English Wikipedia and Simple English Wikipedia.

For validation and testing, we use the evaluation sets of the two tracks we participated in, namely: ASSET (Alva-Manchego et al., 2020) and TurkCorpus (Xu et al., 2016). Both have the same source sentences in their test (359 sentence pairs) and validation sets (2000 sentence pairs). ASSET provides

Model	test_asset		val_asset		test_turk		val_turk	
	BLEU	SARI	BLEU	SARI	BLEU	SARI	BLEU	SARI
T (Baseline)	68.815	36.707	72.561	35.992	71.167	37.801	74.339	37.604
T + Pre	62.488	38.845	71.536	37.700	63.861	38.139	73.627	38.196
T + NER	59.215	39.380	70.433	37.985	58.985	38.996	72.181	**38.375**
SimpleNER	59.324	**39.551**	70.202	**38.897**	59.586	**39.777**	68.622	38.231

Table 3: Scores obtained by the trained models on different test and validation sets (best scores are bolded)

1.	**Source**	*"orton and his wife were happy to have alanna marie orton on july 12 , 2008."*
	Baseline (T)	*"orton and his wife , dorothy marie orton on july 12 , 2007 ."*
	SimpleNER	*"orton and his wife supported alanna marie orton on july 12 , 2008."*
2.	**Source**	*"aracaju is the capital of the state."*
	Baseline (T)	*"it is the capital city of the country ."*
	SimpleNER	*"aracaju is the capital city of the country ."*
3.	**Source**	*"yoghurt or yogurt is a milk-based food made by bacterial fermentation of milk."*
	SimpleNER	*"yogurt is a type of food that is made by bacterial fermentation of product@1."*
4.	**Source**	*"entrance to tsinghua is very very difficult."*
	SimpleNER	*"the entrance to tsinghua is very very simple ."*

Table 4: Sample outputs of the baseline(T) and SimpleNER models on the TurkCorpus-testset

10 human-annotated simplifications for each of the 2359 source sentences, whereas TurCorpus provides 8.

Apart from lower-casing all three splits of the data, the data pairs of the trainset with token length lower than 3 were removed, and sentence pairs with compression ratio ($len(target)/len(source)$) beyond the bounds [0.2, 1.5] were omitted.

4.3 Evaluation Metrics

Our model is evaluated on both BLEU (Papineni et al., 2002) and SARI (Xu et al., 2016). But as Martin et al. (2020) points out, BLEU favours directly replicating the source sentence because of a high N-Gram similarity between the source and target sentences in most sentence simplification datasets. Therefore we only use SARI to rate and compare the models. We also make use of SARI to choose the best performing checkpoints on the validation sets of each of the tracks for evaluation on their respective test sets.

4.4 Training

All models were trained on 4 Nvidia GeForce GTX 1080 Ti GPUs with 64 GB of vRAM. Training was carried out for 20 epochs, and took roughly 11 hours for each model. For all four models, we set the control tokens to NbChars: `0.95`, LevSim: `0.75`, and WordRank: `0.75`. We have omitted DepTreeDepth as Martin et al. (2020) shows that using all four tokens brings down the overall performance.

5 Results

We report the BLEU and SARI scores on the test and validation splits of the ASSET & TurkCorpus datasets for each of the four models (Table 3). All three variants outperform the baseline model (T) across evaluation sets. Using pretrained embeddings (T+Pre) and NER tagged data (T+NER) individually boosts the baseline SARI scores substantially, with the latter approach providing a larger increment in the performance. Using both methods together, further improves the overall SARI score (SimpleNER). Also note how the general BLEU score of the models reduce as the SARI score improves, indicating an increasingly dissimilar and simplified generation.

SimpleNER shows a better retention of named entities from the source sentence than the baseline model (Example 1, Table 4). The contrast is clearer between T+Pre and SimpleNER, as the standalone use of pretrained embeddings in T+Pre allows for unwanted switching between two named entities with similar vector representations (eg. *"2007"* & *"2008"*). Also, NER tagging prevents the unwanted shift from proper nouns to pronouns as observed in the baseline model (Example 2, Table 4).

We also noted that using NER tagging can hamper certain outputs: While decoding, if the model generates an NER-tag that either has a type or index mismatch with the original NE token, then the tag remains in the output even after NER-untagging (Example 3, Table 4). Also, using pretrained-embeddings can result in instances where a source gets replaced with another token having a similar vector representation. This was particularly observed when some tokens were replaced by their exact antonyms (Example 4, Table 4).

6 Social Impact

The following is a summary of the response submitted with our output and model card submission to the GEM 2021 modelling shared task.

6.1 Real World Use

Our model can be utilized to produce point-to-point simplifications for people with cognitive disabilities, to read and understand text. Additionally, it proves helpful for second language learners, especially in public service centres such as airports or health clinics. Although the use of NER-mapping improves our model performance, it can lead to certain pitfalls. Masking NERs before training assumes that named entities don't need to undergo simplification or elaboration. This may be true for most evaluation datasets like ASSET and TurkCorpus, however this isn't the case for many real world cases. High-ranked named entities are often part of domain specific texts, which may require further explanation to be clearly understood by the general public.

6.2 Measuring Impact

Elaboration and replacement of NEs are both crucial for simplification and also the pitfalls of our model. This shows that there is more linguistic information and knowledge of the named entities required to build the model to perfection or evaluate its results. Thus, the best suited method would be a manual evaluation and it could be as simple as a filling a likert scale on how well the simplification and elaboration were.

Since this method is inefficient with respect to time and resources, there is a need for automated evaluation methods to approximate human judgment. A rudimentary measure to work on could take into account the NE's word rank (WR) and its average similarity (AS) to the other words in its sentence. Here, a high WR and a low AS would imply that the sentence does not contextualize the NE even when it might require elaboration. The other case would be when the NE has a relatively low WR and a high AS implying that the sentence contextualizes the NE aptly.

7 Conclusion

In this paper, we report the performance of four Seq2Seq Transformer models on the sentence simplification task of GEM 2021 under two tracks: ASSET and TurkCorpus. We show that individually using pre-trained embeddings and NER-replaced data substantially boosts the performance of a Transformer model assisted by control tokens. The NER tagging prevents the model from replacing important NEs with low rank tokens Also, using pretrained embeddings lets the model access a larger and fine-grained content-word vocabulary for simplification, despite training the model on relatively small data. When put together, the two approaches give rise to a much higher overall performance on the task.

8 Future Work

Some pitfalls to be addressed are: The mismatch between the NER tags generated at the simplified end and the original NE tokens could be due to the exact string matching for NEs, the use of static embeddings (FastText) may have caused the unwanted swaps between highly similar tokens. Using finedtuned contextual embeddings may help. Additionally, since simplification datasets like TurkCorpus and ASSET might utilize different summarization styles, adding a control token to encode and control the output style could be explored.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, pages 4668–4679, Online. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Richard Evans, Constantin Orasan, and Iustin Dornescu. 2014. An evaluation of syntactic simplification rules for people with autism.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 864–870, Online. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Louis Martin, Éric de la Clergerie, Benoît Sagot, and Antoine Bordes. 2020. Controllable sentence simplification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4689–4698, Marseille, France. European Language Resources Association.

Louis Martin, Samuel Humeau, Pierre-Emmanuel Mazaré, Antoine Bordes, Éric Villemonte de la Clergerie, and Benoît Sagot. 2019. Reference-less quality estimation of text simplification systems. *CoRR*, abs/1901.10746.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: Reading assistance for low-literacy readers. In *Proceedings of the 27th ACM International Conference on Design of Communication*, SIGDOC '09, page 29–36, New York, NY, USA. Association for Computing Machinery.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China. Coling 2010 Organizing Committee.