CS230

An Edit-Based Text Simplification Game

Tom Saberhagen
Department of Computer Science
Stanford University
tsabe@stanford.edu

Abstract

Text simplification aims to make language easier to read while preserving semantics of the original text. Language is complex, so automated simplification has no straightforward algorithm. However, discrete simplifying actions such as splitting sentences, choosing simpler words, and dropping words are more tractable. We have framed text simplification as a game in which the player chooses how to simplify a text through a sequence of discrete simplifying actions. The game score balances dual objectives of simplicity and semantic preservation. We train a Reinforcement Learning agent to play the game.

1 Introduction

There is an extensive literature on automated text simplification (Al-Thanyyan, et al. 2021; Shardlow et al, 2014). It has long been recognized that simplified text leads to more efficient human understanding and reasoning. Simplification also may become an important pre-processing step in NLP applications, helping computers to learn and reason from text. It may be especially effective as a data augmentation strategy for NLP applications (Van et al., 2021).

Simplification has two main goals: to enhance readability while preserving semantics of the original text. These goals can conflict and must be balanced. For example, simplifying a text to a kindergarten reading level often results in substantial loss of meaning and nuance. There is no single 'correct' text simplification. One expert or user may favor extensive paraphrasing and restructuring, while another may favor minor editing for clarity. Ideally, simplification could be tuned to the needs of its audience.

Sequence-to-sequence models including large language models can produce effective simplifications. However, such models have shortcomings (Kumar et al., 2020). They are difficult to tune for specific audiences, and the supervised learning approach in a complex context such as NLP depends on availability of large aligned training corpora that are targeted for specific audiences. For example, a distinct training corpus would be needed for simplification to a 5th grade level versus to an 8th grade level. These challenges have created interest in encoder-decoder models in which the decoder is guided by reinforcement learning (Zhang and Lapata, 2017; Nakamachi et al, 2020; Yanamoto et al., 2022).

Edit-based approaches take a different path to text simplification. These approaches rely more on algorithms, databases of common simplifications, and scoring models. This reduces reliance on large, high quality aligned training corpus when compared to a purely supervised learning approach. However, it is a risky business to edit language sequences without the highly nuanced semantic and grammatical capability of an LLM.

We seek to combine the best aspects of edit-based approaches and sequence-to-sequence models. We frame text simplification as a multi-step game in which the player takes discrete edit-based actions.

CS230: Deep Learning, Autumn 2022, Stanford University, CA.

Discrete game actions are powered by LLMs that have been fine-tuned to a specific task, e.g., sentence splitting. We teach a reinforcement learning agent to play the game and coordinate a series of discrete simplifying actions. This approach enjoys the benefits of edit-based approaches (controllability and modest training data requirements) while still leveraging the proven strengths of LLMs (excellent semantic and grammatical nuance).

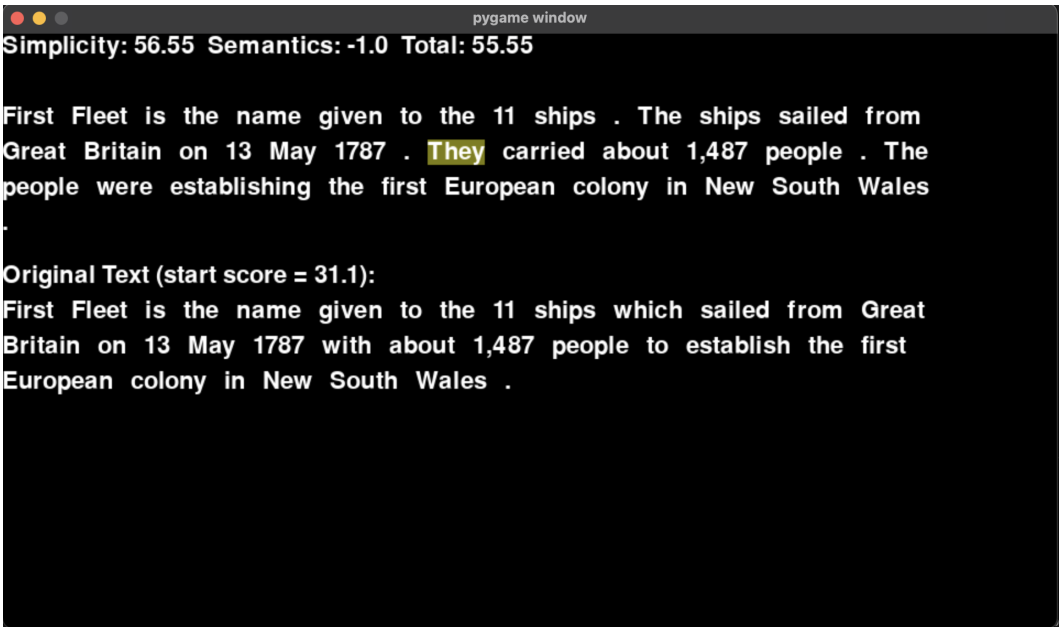

Figure 1 - The Text Simplification Game

Novelty. Our application of reinforcement learning to simplification differs from prior work. Zhang and Lapata (2017), Nakamachi et al. (2020), and Yanamoto et al. (2022) apply reinforcement learning to train encoder-decoder models, while our approach is more akin to edit-based methods of Kumar (2020) or Narayan and Gardent (2016). And unlike prior edit-based work, our approach relies on LLMs to ensure semantics and grammar are maintained.

2 Related work

This project is inspired by prior work that has sought to elaborate and clarify how discrete simplifying actions contribute to a holistic simplification.

Specifically, the project builds on lexical simplification work by Xu et al. (2016) who created the TurkCorpus. We also build on syntactic simplification work by Narayan et al. (2017) who created the WebSplit corpus and defined the "Split-and-Rephrase" sentence simplification task. We also build upon the ASSET corpus (Manchego et al., 2020). ASSET was created as "a dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations."

Narayan et al. (2017) provides an outline of additional related work with a longer list of possible rewriting operations: sentence compression, multi-sentence fusion, sentence paraphrasing, and sentence simplification. Our project architecture could be expanded to encompass any number of such discrete simplifying actions, each powered by its own dedicated fine-tuned model.

Kumar et al. (2020) proposes an "Iterative Edit-Based Unsupervised Sentence Simplification" approach. Candidate simplifications are generated by applying discrete edits and are evaluated with a robust scoring function. The spirit is very similar to our approach, however we use reinforcement learning alongside fine-tuned large language models to optimize editing.

Several approaches have been taken to apply reinforcement learning to text simplification. Zhang and Lapata (2017) introduced DRESS, an encoder-decoder model coupled to a reinforcement learning framework. Nakamachi et al. (2020) use a reward based on grammaticality, meaning preservation, and simplicity to train a BERT model. These holistic sequence-to-sequence models are a good approach, but the approach we offer has greater flexibility to tune and tailor simplifications.

3 Datasets, Features, and Model Preparation

ASSET Corpus for Training and Test. The ASSET corpus (Manchego et al., 2020) has 2000 validation and 359 test texts. We use the validation texts for game play during training. Our game environment extracts features from the text using the Spacy NLP (Honnibal et al., 2020) and textstat (https://github.com/textstat/textstat) packages. Features are used to provide game state and calculate game score.

The ASSET corpus includes 10 human simplifications for each original text. We have augmented the ASSET data with an eleventh simplification performed by GPT-3 davinci-text-002 (Brown, et al., 2020) using the prompt, "Rewrite the following text so a young child can understand it." This augmentation provides us with a 'zero shot' benchmark.

Benchmark simplifications of moderately complex text (human and zero-shot) often achieved around 5 points of improvement on Flesch-Kincaid Grade Level (Kincaid et al, 1975) and ten points on McAlpine EFLAW (McAlpine, 1997), with substantial variability. Please refer to the Appendix for discussion of metrics.

We used additional datasets to fine tune models that perform discrete actions during game play:

TurkCorpus for Word Simplification. TurkCorpus (Xu et al, 2016) includes 8 human simplifications of short texts drawn from Wikipedia. Human simplifiers were asked to preserve meaning "without losing any information or splitting sentence." We created a 'Simplify Words' action by fine-tuning GPT-3 davinci-text-002 using 885 selected TurkCorpus examples where average syllables per word was reduced by at least 0.2.

Wiki-BM for Split-and-Rephrase. The Split-and-Rephrase Wiki Benchmark or Wiki-BM (Zhang et al, 2020) corpus includes complex sentences paired with "a presumably meaning-preserving simplified rewrite containing multiple simpler sentences." Wiki-BM extends the work of Narayan et al (2017) who introduced the "Split-and-Rephrase" task. We created a Split-and-Rephrase generator by fine-tuning GPT-3 davinci-text-002 model using 720 examples from the Wiki-BM corpus.

CoLA Corpus for Grammar Checking. The CoLA training set (Warstadt et al, 2019) has 8551 examples of sentences that are labeled either grammatical ("1") or not grammatical ("0"). We used this data to create a binary classifier so that our game can assess grammaticality of dropping words. We used huggingface to autotrain a model (https://huggingface.co/autotrain). The resulting model achieved 0.88 accuracy, 0.90 precision, and 0.94 recall. We used PyTorch (Paszke et al, 2019) to implement a softmax activation function.

4 Methods

The game is implemented using PyGame (https://www.pygame.org/news) and is playable by humans. We packaged the PyGame code into a custom OpenAI Gym environment (Brockman, et al., 2016) to facilitate implementation of reinforcement learning. We implemented the agent and training loop in Python, basing our approach initially on the textbook "Deep Reinforcement Learning Hands-On" (Lapan, 2020) and later incorporating techniques from the cleanrl Python library (Huang et al., 2021).

4.1 Reward function

We chose a game score that captures dual goals of simplicity and semantic preservation.

Simplicity. We chose a simplicity measure derived from two traditional measures, Flesch-Kincaid Grade Level and McAlpine EFLAW. Both measures can be calculated without reference to any 'ground truth' human simplifications. The game is easily adaptable to other measures of simplicity, if desired. We use the textstat Python package to calculate FKGL and McAlpine EFLAW.

Semantics. We measure semantic preservation according to cosine similarity of large language model embeddings for the original text versus the "simplified" text. We use GPT-3 babbage model embeddings and Python numpy to calculate cosine similarity. Our metric is easily adaptable as semantic similarity measures evolve and improve.

$$Totalscore = \frac{(70 - EFLAW)}{2} + (30 - FKGL) + 100\left(\frac{\varepsilon^O \cdot \varepsilon^S}{\|\varepsilon^O\|_2 \|\varepsilon^S\|_2} - 1\right)$$

Grammar and fluency. Grammar and fluency could be scored (Kumar et al, 2020; Nakamachi et al, 2020; Napoles et al, 2017). However, our game treats grammatical acceptability as a satisficing metric. For Split-and-Rephrase and Simplify Words, GPT-3 is assumed to produce grammatical results. For word dropping, we use our own grammar checker trained on the CoLA dataset.

4.2 Action space

We implemented a simple action space as detailed in Table 1.

Game Action	Scope	Description
Split-and-Rephrase	Whole text	Use fine-tuned LLM to break up long complex sentences
Simplify Words	Whole text	Use fine-tuned LLM to simplify vocabulary
Drop Word	Single token	Remove word or token if resulting text is grammatical
Keep Word	Single token	Do nothing; advance to next token

Table 1: Text Simplification Game Action Space.

4.3 State space

We chose to adopt a compact state space to inform our reinforcement learning model. By design, our architecture leaves complex semantic judgment to the LLMs that execute actions. The state space reports features that have low computational cost yet plausibly signal the need for simplifications.

State Element	Description
1. Word length	Number of characters in current word
2. Lemma length	Number of characters in lemma of current word
3. Ancestor count	Number of grammatical ancestors of current word
4. Sentence length	Number of words in current sentence
5. Simplicity score	A scaled version of game simplicity score
6. Split-and-Rephrase count	Number of times Split-and-Rephrase executed in current game
7. Simplify Words count	Number of times lexical simplification executed in current game

Table 2: Game State Space.

4.4 Model architecture

Our architecture is modeled on the Deep Q-Network architecture (Mnih et al., 2015) and implemented in PyTorch. We have a more compact state space than many DQN variants. Unlike DQN, we do not convey state at the pixel level, so we do not use convolution layers. Our policy network consists of a 7x1 input reflecting our state space; three fully connected layers of 32 nodes each with ReLU activation; and an output layer of Q-values for each of our 4 actions.

We trained the network by minimizing the expected mean square error between a learning target and an approximation of the Q-function, using MSE loss function and Adam optimizer.

For hyperparameters, we used an epsilon decay of 1e-4, a learning rate of 1e-4, a gamma of 0.99, and batch size of 32, sampled uniformly at random from a replay buffer. Like DQN, we used an online network and a separate target network. We updated the target network every 128 game moves.

5 Experiments/Results/Discussion

In total, we logged more than 7100 games. A key metric to observe training progress is average score over the past 100 games. Our best model achieved a peak of more than +9 points in score on average over a 100-game span. Appendix Figure 3 depicts the training progress.

We used the EASSE package (Alva-Manchego et. al, 2019) to provide traditional simplification metrics for our best model predictions on the ASSET test set, comprising 359 test sentences that include human references for calculating BLEU and SARI scores. Taken as a whole, our metrics

are competitive, yielding especially strong FKGL and relatively weak SARI. For comparison, we present baselines of DMASS-DCSS (Zhao et al, 2018) and DRESS (Zhang and Lapata, 2017). Both are common benchmarks scored using EASSE. DMASS-DCSS is regarded for its strength in SARI metrics as it tends to make simplification choices similar to human simplifiers. DRESS applies reinforcement learning with a SARI reward to encourage simpler outputs from a sequence-to-sequence model.

EASSE Metrics:	BLEU	SARI	FKGL	Exact Copies	Lexical Complexity
Game Agent	80.39	35.93	5.8	0.23	8.28
EASSE Human Reference	69.2	44.56	6.64	0.01	8.08
DMASS-DCSS	71.44	38.67	8.08	0.05	8.07
DRESS	84.24	37.07	7.7	0.22	8.13

Table 3: Comparison of EASSE Metrics Using ASSET test; https://github.com/feralvam/easse

Original	Nevertheless, Tagore emulated numerous styles, including craftwork from northern New Ireland, Haida carvings from the west coast of Canada (British Columbia), and woodcuts by Max Pechstein.
Game agent	Tagore emulated numerous styles. These styles included craftwork from northern New Ireland. They also included Haida carvings from the west coast of Canada. They also included woodcuts by Max Pechstein.
Human reference	Tagore copied numerous styles including crafts from North New Ireland, Haida carvings from west Canada's coast (British Columbia) and woodcuts from Max Pechstein.

Table 4: Illustrative Original Text and Simplifications.

From our earliest iterations, it was evident to us (and to the RL agent) that the Split-and-Rephrase action is helpful in a strong majority of games. This is not surprising. Our simplicity score based on FKGL and McAlpine rewards short sentences, and Split-and-Rephrase rarely loses many semantic points. Also, we believe the training data for Split-and-Rephrase is of high quality.

We sought to balance the game by increasing effectiveness of the Simplify Words. Originally, Simplify Words acted at the token level with a synonym generator. However, context matters a great deal when simplifying words. We introduced a text-level action. We created a subset of TurkCorpus intended to exemplify lexical simplification, but qualitatively we think this training data can be improved.

We sought to encourage exploration of Drop Word by several means. We added the sixth and seventh state elements to encourage the agent to learn more quickly that both Split-and-Rephrase and Simplify Words actions have diminishing returns when executed repeatedly.

6 Future Work

Reward function. The present model favors a "low-FKGL style" characterized by short, choppy sentences. We think this is an effective simplification technique. In future work, we would consider additional rewards for lexical simplification and fluency.

Action space. Additional, finer-grained simplifying actions should be added to the toolbox. While our RL agent identifies some instances to "Drop Word" from its word-level optimization, this task may realistically be better handled at the text level by a fine-tuned large language model. A few hundred training examples focused on word dropping may be sufficient to produce a decent fine tune. Other actions that could be added include passive to active voice, paraphrases, and rephrasing to remove idioms, for example.

State space. One learning is that actions we initially thought could be executed at the word level (word simplification and word dropping) in fact entail substantial contextual nuance. We now think these actions are ideally performed in context of the whole text. This argues for a state space that is focused on providing the agent with metrics relevant to the whole text, with less emphasis on single words. Possibly, an embedding vector that encodes fluency and style could be designed and incorporated.

Model Architecture. Unless the state state is substantially expanded (perhaps by adding a text embedding vector or encoding of fluency and style), we do not think the parameter count or depth of the policy network is a constraint. Rather, we think the structure of the reward, action space, and state space is primary. However if a larger state space and network is adopted, it could become relevant for optimization that we will likely continue to observe a distribution of game scores characterized by many games at or near a score of zero (typically when original sentences are simple) and also many games with scores of +10 or more. In other words, outcomes can be characterized as bi-modal or multi-modal. As the network becomes more complex, Huber loss may be more efficient than MSE, and the C51 algorithm (Bellemare et al, 2017) might improve upon DQN.

Educational Applications. The game is playable by humans. A refined version of the game that includes additional simplifying actions, a broad array of texts to simplify, and a battle-tested scoring function could provide the engine for an educational tool useful for foreign language learners, children, or technical writers who need to write for a global audience of non-native speakers.

7 Conclusion

We frame text simplification as a game that can be played by a human or a reinforcement learning agent who applies discrete simplifying actions to text. We use fine-tuned large language models to execute discrete simplifying actions, such as sentence splitting and simplifying words. We show preliminary, promising results. Our system, which leans heavily into sentence splitting, achieves very low FKGL and largely fluent output. Future work can add insight and interest by incorporating variations on scoring and finer-grained simplifying actions.

8 Contributions

This was an individual project.

References

Al-Thanyyan, S. S., Azmi, A. M. (2021). Automated text simplification: a survey. ACM Computing Surveys (CSUR), 54(2), 1-36.

Alva-Manchego, F., Martin, L., Scarton, C., Specia, L. (2019). EASSE: Easier automatic sentence simplification evaluation. arXiv preprint arXiv:1908.04567.

Alva-Manchego, F., Martin, L., Bordes, A., Scarton, C., Sagot, B., Specia, L. (2020). ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. arXiv preprint arXiv:2005.00481.

Alva-Manchego, F., Scarton, C., Specia, L. (2021). The (un) suitability of automatic evaluation metrics for text simplification. Computational Linguistics, 47(4), 861-889.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J. (2021). CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. arXiv preprint arXiv:2111.08819.

Kumar, D., Mou, L., Golab, L., Vechtomova, O. (2020). Iterative edit-based unsupervised sentence simplification. arXiv preprint arXiv:2006.09639.

Lapan, M. (2020) Deep Reinforcement Learning Hands-On. Packt Publishing.

Kincaid, J. P., Fishburne Jr, R. P., Rogers, R. L., Chissom, B. S. (1975). Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Naval Technical Training Command Millington TN Research Branch.

McAlpine, Rachel. (1997). Global English for global business. Longman.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

Nakamachi, A., Kajiwara, T., Arase, Y. (2020, December). Text Simplification with Reinforcement Learning Using Supervised Rewards on Grammaticality, Meaning Preservation, and Simplicity. In Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop (pp. 153-159).

Napoles, C., Sakaguchi, K., Tetreault, J. (2017). JFLEG: A fluency corpus and benchmark for grammatical error correction. arXiv preprint arXiv:1702.04066.

Narayan, S., Gardent, C., Cohen, S. B., Shimorina, A. (2017). Split and rephrase. arXiv preprint arXiv:1707.06971.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

Shardlow, M. (2014). A survey of automated text simplification. International Journal of Advanced Computer Science and Applications, 4(1), 58-70.

Van, H., Tang, Z., Surdeanu, M. (2021). How May I Help You? Using Neural Text Simplification to Improve Downstream NLP Tasks. arXiv preprint arXiv:2109.04604.

Warstadt, A., Singh, A., Bowman, S. R. (2019). Cola: The corpus of linguistic acceptability (with added annotations).

Xu, W., Napoles, C., Pavlick, E., Chen, Q., Callison-Burch, C. (2016). Optimizing statistical machine translation for text simplification. Transactions of the Association for Computational Linguistics, 4, 401-415.

Yanamoto, D., Ikawa, T., Kajiwara, T., Ninomiya, T., Uchida, S., Arase, Y. (2022, November). Controllable Text Simplification with Deep Reinforcement Learning. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (pp. 398-404).

Zhang, X., Lapata, M. (2017). Sentence simplification with deep reinforcement learning. arXiv preprint arXiv:1703.10931.

Zhang, L., Zhu, H., Brahma, S., Li, Y. (2020). Small but mighty: New benchmarks for split and rephrase. arXiv preprint arXiv:2009.08560.

Zhao, S., Meng, R., He, D., Andi, S., Bambang, P. (2018). Integrating transformer and paraphrase rules for sentence simplification. arXiv preprint arXiv:1810.11193.

9 Appendix

9.1 ASSET corpus

Original Text	Adjacent counties are Marin (to the south), Mendocino (to the north), Lake (northeast), Napa (to the east), and Solano and Contra Costa (to the southeast).
Example Human Simplification	Neighboring counties are Marin, Mendocino, Lake, Napa, Solano, and Contra Costa.
GPT-3 Zero-Shot Simplification	The counties that are next to Sonoma County are Marin County (to the south), Mendocino County (to the north), Lake County (northeast), Napa County (to the east), Solano County, and Contra Costa County (to the southeast).

Table 5: An example from augmented ASSET validation set. Original and simplifications.

9.2 TurkCorpus

Original Text	His travel writings and his extensive diaries and correspondence have also been published.
Example Human Simplification	His writings from his travels, large amounts of diaries and letters have been published as well.

Table 6: An example lexical simplification extracted from TurkCorpus. Original and simplification.

9.3 Wiki-BM

Original Text	After Pinocchio accidentally sets Lorenzinis theatre on fire, Lorenzini changes career and begins luring unruly children to pleasure island, where they inevitably drink cursed water which turns them into donkeys.
Example Human Simplification	Pinocchio accidentally sets Lorenzini's theatre on fire. Afterwards, Lorenzini changes his career. Lorenzini begins luring unruly children to Pleasure Island. At Pleasure Island, the unruly children inevitably drink cursed water. The cursed water turns the children into donkeys.

Table 7: An example from Wiki-BM Split-and-Rephrase corpus. Original and simplification.

9.4 CoLA

Grammatical	Bill rolled out of the room. The gardener watered the flowers flat. Bill broke the bathtub into pieces. The professor talked us into a stupor.
Not Grammatical	They drank the pub. The professor talked us. Harry coughed us into a fit. They caused him to become angry by making him.

Table 8: Example sentences labeled grammatical and not grammatical.

9.5 Metrics

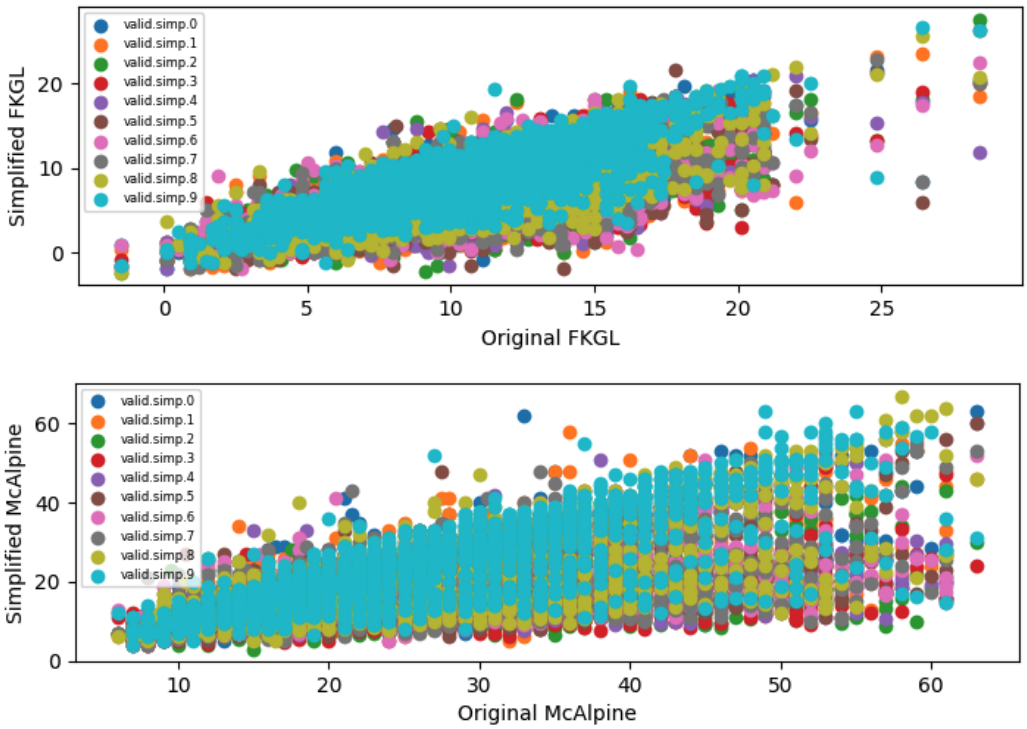

Figure 2 - Benchmark Original vs Simplified Metrics. Each dot represents a text.

9.6 Flesch-Kincaid Grade Level (FKGL)

FKGL is a score that relies on average sentence lengths and number of syllables per word. In earlier years of automated text simplification, the score was criticized for failing to take account of 1) grammatical errors introduced by automated systems and 2) semantic drift from an original text. In our view it remains a solid choice for a simplicity score in a setting where grammaticality is assured and semantic drift is separately measured.

$$FKGL = 0.39\left(\frac{totalwords}{totalsentences}\right) + 11.8\left(\frac{totalsyllables}{totalwords}\right) - 15.59$$

9.7 McAlpine EFLAW

The McAlpine EFLAW (McAlpine, 1997) is a simplicity score recommended by Rachel McAlpine in an article titled "From Plain English to Global English." The score is based on the insight that excessive use of "miniwords" (common words of one, two, or three letters) is a hallmark of convoluted structure and wordiness. Simpler texts (e.g., with sentences following Subject-Verb-Object) use few mini-words. The EFLAW score is defined as (W+M)/S where W = the number of words in a text, M = the number of mini-words in a text, and S = the number of sentences in a text. A score in the range of 1-20 generally indicates easy to understand text, while a score over 30 reflects confusing text. We believed that incorporating EFLAW to penalize mini-words might have a beneficial effect towards learning an effective Word Drop policy.

9.8 EASSE Metrics

We refer to Alva-Manchego (2019) for additional description of the EASSE system and the metrics it measures including BLEU, SARI, proportion of exact copies, and lexical complexity. On lexical complexity, the paper notes it is "computed by taking the log-ranks of each word in the frequency table. The ranks are then aggregated by taking their third quartile."

9.9 DQN Architecture Overview

https://web.stanford.edu/class/cs234/CS234Win2019/slides/lnotes6.pdf

9.10 Results, 100-Game Average Score

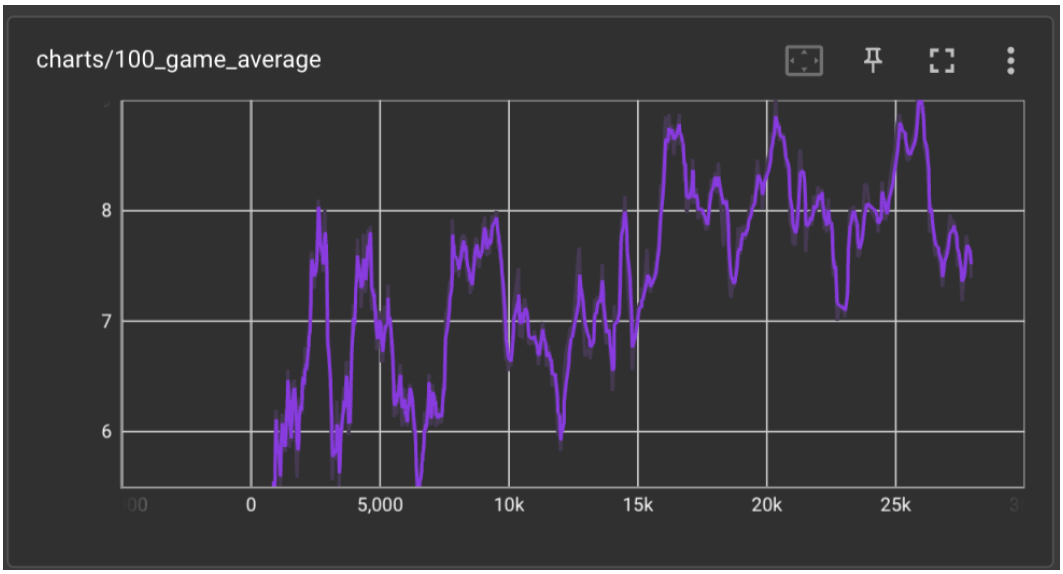

Figure 3 - 100-Game Average Score Improvement vs Game Moves