Leveraging Large Pretrained Models for WebNLG 2020

Xintong Li, Aleksandre Maskharashvili , Symon Jory Stevens-Guille and Michael White

Department of Linguistics
The Ohio State University
znculee@gmail.com maskharashvili.1@osu.edu
stevensguille.1@osu.edu mwhite@ling.osu.edu

Abstract

In this paper, we report experiments on fine-tuning large pretrained models to realize resource description framework (RDF) triples to natural language. We provide the details of how to build one of the top-ranked English generation models in WebNLG Challenge 2020. We also show that there appears to be considerable potential for reranking to improve the current state of the art both in terms of statistical metrics and model-based metrics. Our human analyses of the generated texts show that for Russian, pretrained models showed some success, both in terms of lexical and morpho-syntactic choices for generation, as well as for content aggregation. Nevertheless, in a number of cases, the model can be unpredictable, both in terms of failure or success. Omissions of the content and hallucinations, which in many cases occurred at the same time, were major problems. By contrast, the models for English showed near perfect performance on the validation set.

1 Introduction

The WebNLG Challenge has attracted increasing attention since 2017. It aims to promote research on realizing resource description framework (RDF) triples (Gardent et al., 2017) of various categories in fluent and accurate natural language. The RDF triple sets on which the WEBNLG dataset is best are from DBPedia (Auer et al., 2007) and consist of one or several triples, where each triple contains one subject, one predicate, and one object. The most challenging part of WebNLG 2017 is to generate from data containing unseen categories, where many predicates and subject/object entities don't appear in the training data.

Consequently, for WebNLG 2020, we are especially interested in improving realization quality for data from unseen categories. Because the size of the training data in the WebNLG challenge is quite modest, we think it is important to introduce extra knowledge into the pipeline. Considering that large pretrained models have achieved tremendous success in various NLP tasks, including recently for data-to-text NLG (Kale, 2020; Kale and Rastogi, 2020), we investigate using pretrained models to enhance generation from unseen categories, and find they work remarkably well.

In addition, we try to further improve the fine-tuned large pretrained models by using reverse model reranking, which involves reranking the outputs of the beam of the original forward model for a given source input using the perplexity of that input (conditioned on the output) according to the reverse model. As shown in Shen et al. (2019); Yee et al. (2019), techniques similar to the reverse model reranking method we use here, under the guise of RSA or noisy channel models, have shown significant benefits for generation tasks when there is room for improvement. However, in our experiments, reverse model reranking not only provides no benefit but also slightly harms realization quality. By looking at the BLEU oracle reranking scores, we see a considerable potential for reranking, so we surmise that the quality of the reverse model must be too low to be useful. This could indicate that it is more difficult for large-scale pretrained models to learn to reconstruct meaning representations than to generate natural language text from them.

2 System Description

Data The English dataset contains 13,211 and 1,666 triple sets for training and validation, respectively. The Russian dataset contains 5,573 and 789 triple sets for training and validation, respectively. Each triple set has two versions, original and modified. We use the modified version because it is a cleaner version. Each triple set is also associated

RDF:	`"Aarhus_Airport | cityServed | \"Aarhus, Denmark\""`
MR:	`__subject__ Aarhus Airport __predicate__ cityServed __object__ Aarhus, Denmark`
Lex:	`The Aarhus is the airport of Aarhus, Denmark.`

Table 1: The format of original and preprocessed data. RDF is the original triple from DBPedia. MR is our preprocessed meaning representation. Lex is the reference realisation of the RDF triple.

with one to several surface realizations. Therefore, we have 35,426 and 4,461 parallel data items for training and validation, respectively, on the English task, and 14,630 and 2,062 ones for training and validation, respectively, on the Russian task.

Preprocessing We linearize each triple set into a sequence with three delimiters, which are `__subject__`, `__predicate__`, `__object__`, as shown in Table 1. Additionally, we remove the underscores inside the subject/object (_), quotes surrounding subject/object (\"), and the beginning and end quotes ("), in order to reduce noise in MRs.

Pretrained Models The training data of the WebNLG challenge is too modest in size to sufficiently train a large neural model to generalize very well. In our preliminary experiments on WebNLG 2017 (Gardent et al., 2017), we found that models trained from scratch were very bad at generating with unseen categories, even though the outputs with seen categories were mostly acceptable. This is due to the unseen categories frequently introducing unseen predicates and noun phrases, on which models, having never seen these expressions, struggle to consistently produce text from. Considering that large pretrained models have achieved tremendous success in many areas, we think they should also be helpful for generating unseen content with small training datasets. To test the conjecture that pretrained models would improve performance on unseen content, we use T5 (Kale, 2020) for the English task and mBART (Liu et al., 2020) for the Russian task. These models are chosen because T5 is designed for the monolingual task and mBART is designed for multi-lingual tasks. To be more specific, we use the pretrained T5-Large HuggingFace transformer model (Wolf et al., 2019) and the mBART-Large fairseq model (Ott et al., 2019).

Fine-tuning[1] Considering that both these models tokenize the sentences into subwords and the triple delimiters (e.g. `__predicate__`) are supposed to be indivisible, we add the three special delimiters to the vocabulary of the pretrained models.[2] The embeddings of these three special delimiters are randomly initialized. After extending the vocabularies, the total parameters to be fine-tuned for T5 and mBART are 737,643,008 and 610,851,840, respectively.

The T5 model is fine-tuned using cross entropy loss without label smoothing. The learning rate is constantly 2×10^{-5} and the batch size is 8 samples. We found that an overly large learning rate and batch size could hurt performance. The optimizer is Adam (Kingma and Ba, 2014) where $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and the weight decay is 0. In our preliminary experiments, we found that L2 normalization did not help, whether we used the weight decay of Adam or AdamW (Loshchilov and Hutter, 2017). The best checkpoint is selected by validation with patience of 10 training epochs. With this setting, the best checkpoint is at the end of the 7th epoch.

The mBART model is fine-tuned with different hyper-parameters than T5 because we followed the recommended fine-tuning guidelines of the mBART authors. It is optimized according to cross entropy loss with label smoothing of 0.2. The batch size is 2048 tokens. The learning rate is 3×10^{-5} and the scheduler is polynomial decay with 2500 warmup updates. The optimizer is Adam where $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1 \times 10^{-6}$. The best checkpoint is selected by validation with patience of 20 training epochs. The best checkpoint is at the end of the 32nd epoch.

3 Experiments

3.1 Evaluation Metrics

BLEU We use BLEU-4 (Papineni et al., 2002) implemented by the e2e-metrics (Dušek et al., 2018). Although we can evaluate with multiple references by combining the same MR, we still

[1]Code: https://github.com/znculee/webnlg2020

[2]We only extend the vocabulary for the T5-Large of HuggingFace transformer by the `add_token()` function, because fairseq does not support this feature currently to the best of our knowledge.

Models	BLEU	BLEURT	Fluency	Grammar	Semantics
T5	46.51	0.464176	2.58516	2.70785	2.53218
T5-RMR	45.40	0.443449	2.56646	2.69307	2.51706
T5-RMRB	45.31	0.435909	2.55813	2.68477	2.51426
T5-RMRF	45.16	0.439485	2.56026	2.68461	2.51901
T5-ORAC	55.13	0.538256	2.62836	2.74878	2.57138

Table 2: Evaluation scores on the English validation set. BLEURT is the pretrained BLEURT-base-128 model without any fine-tuning on WebNLG data. Fluency, grammar, and semantics are the scores from BLEURT fine-tuned on the WebNLG 2017 three-fold human evaluation scores. See text for model descriptions.

report the single reference BLEU here to be consistent with other metrics which cannot consider multi-reference, e.g. the following BLEURT metric.

BLEURT As an n-gram metric, BLEU may not fairly judge the correctness of the generated texts, and since human evaluation is expensive, we investigate using a recently developed model-based metric, known as BLEURT (Sellam et al., 2020), for automatic evaluation. BLEURT's evaluation shows state-of-the-art consistency with human judgements on the WMT Metrics Shared Task (Bojar et al., 2017). And it is also shown to be well adapted to other domains with its pretrained model, such as WebNLG 2017.[3] Considering we do not have the human evaluation data for WebNLG 2020 now, the best we can do is to use WebNLG 2017[4] as the data to fine-tune BLEURT to evaluate our models for WebNLG 2020. The human evaluation in WebNLG is three fold, including fluency, grammar correctness, and semantics adequacy. Each of them has one of three possible values (1 or 2 or 3) for the human annotator. Each sentence is assigned up to three annotators. The range of average score is from 1 to 3. Then, we fine-tuned the recommended BLEURT-base-128 on these three different tasks. Specifically, we used all 5,363 items and randomly sample 1,000 of them as validation data and the others are regarded as training data, because the BLEURT paper concludes that using more training data leads to better consistency with human judgements. Following Sellam et al. (2020), the fine-tuning stops at 40,000 steps.

[3]The checkpoints of BLEURT fine-tuned on WebNLG 2017 are not released publicly.

[4]https://gitlab.com/webnlg/webnlg-human-evaluation

Models	BLEU
mBART	24.33
mBART-RMR	23.90

Table 3: BLEU scores on Russian validation set. mBART-RMR indicates mBART with reverse model reranking.

3.2 Validation Performances

The main results for the English task and Russian task are shown in Table 2 and Table 3, respectively. More model comparisons are conducted on the English task only. These models share the same forward model but use an extra reverse model to rerank the generated text after beam search.

- T5-RMR is reverse model reranking with forced decoding perplexity.
- T5-RMRB is reverse model reranking using the BLEU scores of the reconstructed MRs from the reverse model against the original MRs.
- T5-RMRF is reverse model reranking with the sum of the reverse model forced decoding perplexity and the forward model perplexity.
- T5-ORAC is reranking using oracle BLEU scores.

From Table 2, we can see that the vanilla fine-tuned T5 is generally better than its reranking versions. It's surprising that these reranking methods don't work. For the English task, it could be because the model was near ceiling for correctness, however this is not the case with Russian. Although these three reranking methods do not improve upon the baseline, the BLEU oracle strongly suggests that there is considerable potential for reranking

Task	Test Subset	BLEU	BLEU NLTK	METEOR	CHRF++	TER	BERT PRECISION	BERT RECALL	BERT F1	BLEURT
En	All	53.54	0.532	0.414	0.688	0.416	0.958	0.955	0.956	0.61
En	Seen Categories	61.24	0.607	0.434	0.727	0.393	0.964	0.960	0.962	0.61
En	Unseen Categories	47.40	0.474	0.397	0.652	0.437	0.953	0.951	0.951	0.57
En	Unseen Entities	52.37	0.520	0.416	0.694	0.398	0.963	0.960	0.961	0.65
Ru	All	47.29	0.477	0.616	0.622	0.453	0.897	0.882	0.888	N/A

Table 4: Official evaluations of T5 on English (En) and mBART on Russian (Ru) test sets

to improve the outputs. It would be interesting to investigate more effective methods to select more correct generated text in the beam search.

3.3 Official Evaluation

Since the test set references are not available to the challenge participants, the evaluation of test performance is done by the organizers. Table 4 shows the automatic evaluation scores across various measures (Moussalem et al., 2020) for T5 on the English task and mBART on the Russian task, respectively. For the English task, our T5 model is always ranked in the top two over all 35 submissions across all test subsets, and only has a small gap from the first ranked system. For the Russian task, our mBART is ranked third over all 12 submissions. The official report (Castro-Ferreira et al., 2020) should have more comprehensive comparisons and human evaluations for all submissions.

4 Methodology for Analyses

Analyzing the performance of a model is an important aspect of research pursued with neural networks, because even though their architecture is explicit and transparent, their behavior roughly can be compared to a black box.

One of the strategies we relied on in the current work is to analyze success and failure of a model qualitatively by comparing given examples to related proportions of the training data.

We tested our Russian models against the test set, even though it had no labels. Based on our linguistic knowledge and using observations from the training and validation data, we hypothesize a possible text for a given source (meaning representation), which allows us to see whether a model generated a plausible/acceptable text for a given source. We also ran our models for Russian against the validation set. We discuss our findings in Section 5.1.

In addition to Russian, we also analyzed the model performance on English, though we used only the validation set. Our models worked almost flawlessly on the data given for the validation set, both in terms of content realization and lexical and morpho-syntactic selections. The models exhibit high performance on aggregation of content across sentences and also within the same sentence, using coordinating conjunctions for both multiple noun and multiple verb phrases.

5 Performance and Observations: Case of Russian

5.1 mBART and mBART-RMR

We make use of two models for Russian. The first one is the mBART model fine-tuned for our task. The other one is the fine-tuned mBART model with reverse model reranking. We refer to the fine-tuned mBART by mBART and its reverse model reranked version by mBART-RMR.

On the Russian test set, we find that out of 1,102 texts generated by mBART and mBART-RMR, all but 174 were the same. We have analyzed those 174 generated texts for each model and checked their correctness against the source. We find that on those 174 cases, mBART performs better then mBART-RMR. In particular, among those 174 cases, on 82 of them, both of the models were successful, whereas on 33 cases both of them failed. On the rest of the cases, 41 times mBART was the better of the two, whereas mBART-RMR prevailed in 18 cases.

Since mBART showed better performance than mBART-RMR, we tested mBART against the validation set of the WebNLG 2020 dataset. We randomly selected 120 examples and annotated the problematic ones. We found that on average, only 1 out of 3 qualified as a decent Russian text.

5.2 The Same Realizations

We performed our analyses on the cases where both of the models, mBART and mBART-RMR, generate the same text.

We randomly selected 70 short and 61 long cases, where we call an item short if the source has 3 or fewer tuples and long otherwise (the short cases consists of: 15 1-tuples, 37 2-tuples, and 18 3-tuples; while the long cases consists of 13 4-tuples, 21 5-tuples, 14 6-tuples and 13 7-tuples). We found that among the short items, the ratio of correct realizations is 46 to 70. For the long item, the ratio is 9 to 61. (These success rates are close to the numbers we got while analyzing the performance of our models on the validation set of WebNLG).

In many cases, the success of a generated text could be attributed to the fact that a very similar source (MR) has been encountered in the training data.

If MR_1 is part of MR_2 but the model has only seen MR_2, in certain cases we observe that it is still able to correctly generate from MR_1, skipping the parts associated with $\mathrm{MR}_2/\mathrm{MR}_1$ (where $\mathrm{MR}_2/\mathrm{MR}_1$ denotes the part of MR2 that is not in MR1). This is due to model seeing a number of cases in the training data similar to MR1 and MR2, thereby learning how to differentiate between the two.

Hallucinations together with omissions are one of the major problems of the model. They usually occur with sparse or unevenly distributed data. One of the issues we find is that the model may incorrectly associate an entity from the source with a name in a text, even though the correct realization would also be present in the training data (the model has the needed information to be able to select the correct realization of an entity, but it fails to do that). For the sake of illustration we discuss details of such errors. The correct realization of an entity occurred 31 times in the training data. Let us denote the correct realization by cr. However, in certain cases, the model associated the entity with a realization of another entity, which we call the incorrect realization and denote by ir. We find that ir occurred 42 times in the training data. These two realizations, cr and ir occurred 10 times together (i.e. within in the same text). In the context where the model made the mistake, the frequencies of cr and ir are more or less homogeneous (cr appeared 7 times and ir 10 times). However, they differ with respect to how they appear in the training data in general: ir exclusively appears within realizations of 5-tuples, whereas cr appears as a part of texts that are realizations of tuples of various lengths. The source on which the model makes the error of associating the entity with ir also happens to be a 5-tuple; and moreover, this source is very similar to the sources whose realizations contain ir. (Even though cr also appears within texts serving as realizations of 5-tuples, there are only 3 such cases and in all of these cases, these texts also contain ir together with cr.)

In certain cases we also saw that several triples were omitted when the source consisted of 7 tuples. However, this was also true in certain cases where we had only 3-triples. Let us consider the following example:

Example 5.1 Арроз негре родом из региона Каталония в Испании.
EN: Arròs negre comes from Catalonia in Spain.
Source: `__subject__ Arròs negre __predicate__ country __object__ Spain __subject__ Arròs negre __predicate__ region __object__ Catalonia __subject__ Arròs negre __predicate__ ingredient __object__ Squid`

As we see in Example 5.1, the source states squid is an ingredient in Arròs negre, but this is not generated by the model. We look into the training set to find out how many times squid appears as an object to the predicate ingredient. It turns out that squid appears as an ingredient 4 times in the training set, and in all of these MRs, it is as an ingredient of Arròs negre. Given that the model failed to generate this fact, one may conclude that its occurrence in the corpus is too infrequent to be learned. However, the model generates the following:

Example 5.2 Арроз негре родом из Испании, одним из ингредиентов - кальмары.
EN: Arròs negre comes from Spain, one of its ingredients is squid.
Source: `__subject__ Arròs negre __predicate__ country __object__ Spain __subject__ Arròs negre __predicate__ ingredient __object__ Squid`

To make sure that Example 5.2 is not simply copied from the training set, we check that it does not appear in the training set. Moreover, the second clause "одним из ингредиентов - кальмары" (*EN: one of its ingredients - squids*) as a sequence has no occurrence in the training set.

The pair of Example 5.1 and Example 5.2 illustrate an unstable behavior of our models. One might attribute this to the low number of training examples with certain words engaged in those examples. While this might be true, we do not know how to augment the data to overcome those problems.

For further exposition we show examples with similar issues in cases where the number of training examples is relatively high. We discuss a case where the model failed to produce realization of a source consisting of three triples, even though it was successful separately on each of them in other contexts.

Example 5.3 HOK SVE был архитектором 3Arena, который был построен в декабре 2008 года.
EN: HOK SVE was the architect of 3Arena, which was built in December 2008.
Source: __subject__ 3Arena __predicate__ location __object__ Dublin __subject__ 3Arena __predicate__ architect __object__ HOK SVE __subject__ 3Arena __predicate__ completionDate __object__ December 2008
Fails on: *__subject__ 3Arena __predicate__ location __object__ Dublin*
(3Arena, December 2008 = 37 times in training set); (3Arena, Populous = 30 times in the training set); (3Arena, Populous, December 2008 = 20 times in the training set).

We write ($w_1, \ldots, w_n = m$ *times in the training set*) to denote that the words $w_1, \ldots, w_n$ co-occur in m sources (i.e. meaning representations) in the training set. We report these numbers in order to facilitate analysis in each of the cases.

Let us mention that in the training set, we have a very similar text to the one we would like to have generated in Example 5.3, which is the following: HOK SVE была архитектором 3Arena, завершенным в декабре 2008 года и расположенным на Норт-Уолл в Дублине. (*EN: HOK SVE was the architect of 3Arena, completed December 2008 and located at North Wall, Dublin.*)

To analyze why the model fails on a particular test example, we look for other test items with the same predicates and/or entities. For Example 5.3, we find some good realizations of locations in the following cases:

Example 5.4 3Arena расположена в Дублине, Республика Ирландия.
EN: 3Arena is located in Dublin, Republic of Ireland.
Source: __subject__ Dublin __predicate__ isPartOf __object__ Republic of Ireland __subject__ 3Arena __predicate__ location __object__ Dublin
Success!
(3Arena, Dublin = 69 times in the training set) (3Arena, Dublin, Republic of Ireland = 27 times in the training set)

Example 5.5 HOK SVE был архитектором 3Arena, расположенной на набережной Норт-Уолл и завершенной в декабре 2008 года.
EN: HOK SVE was the architect of 3Arena, located on North Wall Quay waterfront and completed in December 2008.
Source: __subject__ 3Arena __predicate__ location __object__ North Wall Quay __subject__ 3Arena __predicate__ architect __object__ HOK SVE __subject__ 3Arena __predicate__ completionDate __object__ December 2008
Success!
(3Arena, December 2008 = 37 times in training set) (3Arena, HOK SVE = 36 times in the training set) (3Arena, HOK SVE, 2008 = 26 times the training set) (3Arena, HOK SVE, 2008, North Wall Quay = 6 times the training set)

Hence, the model successfully realized location and its arguments in Example 5.4 and Example 5.5, even though it failed to produce this text in the very similar Example 5.3. The model indeed seems to learn how to realize triples built using location as their predicate, and the following example can be used to support this conclusion:

Example 5.6 HOK SVE был архитектором 3Arena, который находится на мосту Истлинк.
EN: HOK SVE was the architect of 3Arena, which is on the East Link Bridge.
Source: __subject__ 3Arena __predicate__ location __object__ East Link Bridge __subject__ 3Arena __predicate__ architect __object__ HOK SVE
Success!
(3Arena, HOK SVE, East Link Bridge = 2 times the training set)

The training corpus contains no exact match to this example, either content-wise (source) or realization-wise (text). In particular, every time terms 3Arena, HOK SVE, and East Link Bridge appear in the same source (and are realized in the corresponding text), they are accompanied by the completionDate. The model apparently is capable of learning how to generate text without completionDate when the latter is not present in a source MR.

Our model is also successful in the following example, where it realizes the location by *Dublin* (which it failed to produce in Example 5.3). Furthermore it correctly realizes `Populous` (entity name) and the predicate `architect`.

Example 5.7 Компания Populous была архитектором 3Arena, расположенной в Дублине.
EN: Populous is the architect of 3Arena located at Dublin.
Source: `__subject__ 3Arena __predicate__ location __object__ Dublin __subject__ 3Arena __predicate__ architect __object__ Populous (company)`
Sucsess !
(`3Arena, Dublin` = 69 times in the training set)
(`3Arena, Populous` = 30 times in the training set)
(`3Arena, Populous, Dublin` = 10 times in the training set)

As we see in the case of Example 5.7, the model realized correctly the triple `__subject__ 3Arena __predicate__ architect __object__ Populous (company)`. But in Example 5.8, the model fails on the very same triple. In fact, it does not generate any textual realization corresponding to that triple. We cannot explain why this happens, even by looking in the training set (the numbers are close across both the examples on which the model failed and on which it succeeded).

Example 5.8 3Arena, построенная в декабре 2008 года, расположена в Дублине.
EN: The 3Arena, completed in December 2008, is located in Dublin.
Source: `__subject__ 3Arena __predicate__ location __object__ Dublin __subject__ 3Arena __predicate__ architect __object__ Populous (company) __subject__ 3Arena __predicate__ completionDate __object__ December 2008 predicate completionDate object December 2008`
Fails on: *`__subject__ 3Arena __predicate__ architect __object__ Populous (company)`*
(`3Arena, Populous` = 30 times in the training set) (`3Arena, Populous, December 2008` = 20 times the training set) (`3Arena, Populous, December 2008, Dublin` = 7 times the training set)

6 Conclusions

For Russian, we have seen that in certain cases the models successfully realize meaning representations in text, both with respect to fluency and content, but they fail to do the same in very similar cases. To analyze such cases, we looked into distributions of the relevant data in the training set. While considering distributions provides some insight into the models output, further investigation is needed to explain a number of issues.

It is common knowledge that more data means better performance, and our observations are consistent with this view. Nonetheless, we observed that the same frequency of two entities or predicates in the training data doesn't determine whether the model consistently produces the text corresponding to such content. It is not impossible augmenting the data would induce more errors; this could make the model more biased towards producing text we would like to avoid. Thus it is not straightforward to say whether simple data augmentation would help.

Pretrained models excel due to their accumulating valuable knowledge about the world (semantic and pragmatic) and the language (lexical and morpho-syntactic). To analyze the performance of a pretrained model that is fine-tuned on some specific data, for a specific task, is not straightforward. The possible sources of success or failure are numerous — it might be the pretrained model is the source of success, but contradictory results could be due to inherent biases in the pretrained model.

To conclude, we believe that both the architecture of the model and the data needs to be scrutinized to understand why the model and the data do or do not produce good results.

Nevertheless, on the whole we find that large-scale pretrained models work remarkably well on the WebNLG tasks, especially T5 for English. To our surprise though, we find that reverse model reranking does not appear to work well with pretrained models. This could indicate that it is easier for a pretrained model to learn to generate natural language text from meaning representations than the reverse task of reconstructing meaning representations from text. In the future, we aim to investigate better reranking scorers to realize the potential of reranking shown by the BLEU oracle. Additionally, it would also be interesting to leverage the shape information among the triples and employ constrained decoding (Balakrishnan et al., 2019) to guarantee the correctness of realizations.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Ondřej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the WMT17 metrics shared task. In *Proceedings of the Second Conference on Machine Translation*, pages 489–513, Copenhagen, Denmark. Association for Computational Linguistics.

Thiago Castro-Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussalem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional webnlg+ shared task: Overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG Challenge. In *Proc. of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg, The Netherlands. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv preprint arXiv:2005.10433*.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*. To appear.

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

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Diego Moussalem, Paramjot Kaur, Thiago Castro-Ferreira, Chris van der Lee, Anastasia Shimorina, Felix Conrads, Michael Röder, René Speck, Claire Gardent, Simon Mille, Nikolai Ilinykh, and Axel-Cyrille Ngonga Ngomo. 2020. A general benchmarking framework for text generation. In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Kyra Yee, Yann Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5696–5701, Hong Kong, China. Association for Computational Linguistics.