From Tabular to Vocabular: Data-To-Text Generation

Hussameddine Al Attar
Dept. of Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
hya15@mail.aub.edu

Clark Abou Rjeily
Dept. of Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
cna15@mail.aub.edu

Emilio Bitar Zankoul
Dept. of Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
eab16@mail.aub.edu

Jana Salameh
Dept. of Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
jbs12@mail.aub.edu

Fawzia Zeitoun
Dept. of Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
fhz02@mail.aub.edu

ABSTRACT

Data-to-Text generation is a widely used technique to convert tabular or graphical data into natural language. The generation of natural informative sentences is a challenging task, especially when it comes to human conversation. Natural Language Processing is a considerable endeavor due to the complexity of human speech as considerable attention should be given to preserving the meaning of the data, all the while using correct grammar and structure. In this work, our goal is to fine-tune the T5-base model, an encoder-decoder model that transforms data on a text-to-text basis. Preliminary preprocessing was done on the WebNLG 3.0 dataset, and performance was evaluated by comparison to the state-of-the-art model DataTuner. Our work resulted in an improvement of the input size and the implementation of special character generation, with an average BLEU score of 34.8, a testing loss of 28% and a validation loss of 4%.

I. INTRODUCTION

Over the years, Natural Language Generation (NLG) has been linked to human-machine exchanges [1]. To generate fluent and accurate text, Data-to-Text generation is used. The latter is a technique that allows the conversion of structurally formatted information, such as tables or graphs, into natural language [2]. Hence, it is used by virtual assistants such as Apple's Siri, Amazon Alexa, and Google Assistant to provide users with access to a wide array of information and services from the web, through a natural language interface [3]. The outcome can either be read or listened to upon the user's request and needs, including displaying weather forecasts, finding restaurants, etc. [4]

Conversational Artificial Intelligence is one of the persistent challenges in computer science and artificial intelligence [5]. Since human conversation is complex and ambiguous by nature, learning a conversational AI that can perform a random task set by the user is still under development [5] [7]– we want to generate naturally flowing sentences that convey accurate information.

Traditional Natural Language Generation systems are based on the use of templates to ensure good control over the output [3]. However, with the growth of the digital world and new services needed by users, defining templates has become a more challenging task [3]. Hence, the need for a more generalized approach to diverse topics.

Moreover, traditional Data-to-Text approaches use a pipeline machine learning methodology. The problem is divided into multiple sub-problems including content selection, text structuring, sentence aggregation, and lexicalization [6]. The method has been proven to be labor-intensive to create and might pose risks of compounding errors [6].

In this context, transfer learning, in which a model is pre-trained on a data-rich task to later be fine-tuned to perform more specific tasks, has recently emerged as a powerful Natural Language Processing (NLP) technique [8]. The latter equips the model with general-purpose abilities, thus making it easy to transfer them to downstream tasks. Effectively, pretraining in the form of end-to-end transformer-based models has been shown to provide better performance than the pipelined architectures [2].

Data-to-text generation still struggles to maintain semantic fidelity in its output as well as generalize its output to previously unseen domains, despite the great amount of work being done in the industry [6]. Another challenge faced concerns grammar and word placement in a sentence: while the sentence may be correct and somewhat understandable, it might sound heavy, or include unnecessary redundancies, ultimately reducing the quality of the sentence generated. In fact, the main problems faced are the selection and division of the input data (what to say), and the phrasing of the generated sentence (how to say it). [9]

In our work, we aim to fine-tune the T5-base model, an end-to-end transformer-based pre-training model, to perform the Data-to-Text generation task and address the aforementioned research challenges.

After preliminary preprocessing tasks on the WebNLG 3.0 dataset, we trained and tested the fine-tuned T5-base model. We then assessed the model's performance against the state-of-the-art "DataTuner" model, trained on the same dataset.

We analyzed T5's performance on its own, improved some earlier limitations such as the input size and generation of special letter characters and determined some limitations of the model within NLG with an average BLEU score of 34.8, a testing loss of 28%, and a validation loss of 4%

The remainder of the paper is organized as such: the proposed method and implementation are presented in section II. The performed experiments are detailed in section III, then the results and their analysis are discussed in sections IV and V respectively. Finally, section VI concludes our work and we added section VII for better visualization of our results.

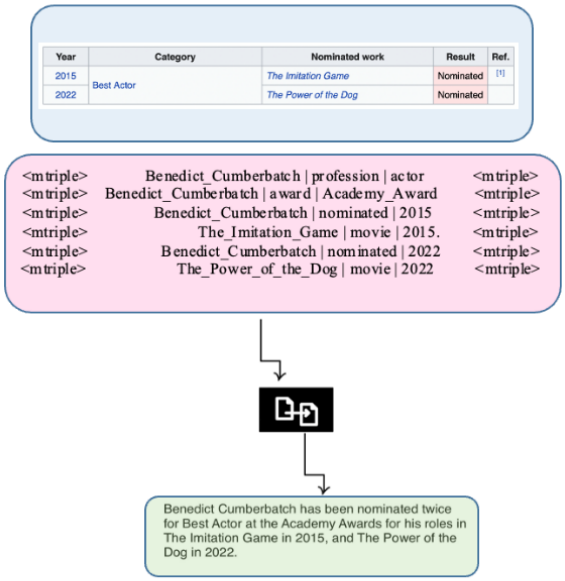

Fig. 1 Black Box Diagram of sample input & output. (Information is courtesy of Wikipedia)

II. Proposed Method and Implementation

A. Model Description

The Text-To-Text Transfer Transformer (T5) developed by Google, is a transformer-based architecture model. It is a pre-trained model that can be fine-tuned to perform a variety of downstream tasks, including Data-to-Text generation [11]. T5 was pre-trained on the C4 dataset (Colossal Clean Crawled Corpus), a data-rich set of web scrapes of size 750 GB [10]. This model proposes the reframing of all NLP tasks into a text-to-text format (input and output are text strings), the latter framework allows the use of the same model, loss function, and hyperparameters on any NLP task [10].

The T5 model comes in 5 sizes: small, base, large, 3b, and 11b, all differentiated by their number of incorporated parameters. [11]. We chose to work with the base model containing 220 million parameters. [12]

As previously mentioned, the T5 base model is based on the Transformer architecture which consists of building blocks and different layers. The primary building block relies on self-attention where the model processes each sequence and replaces each element by the weighted average of the rest. [13] It also includes an encoder and a decoder, each realized by a stack of encoders and decoders. Both have a Multi-Head Attention layer preceding a Feed Forward Neural Network with the Decoder having an added layer of Masked Multi-Head Attention. In addition, each encoder contains a normalization layer following the subcomponents. [13]

The input/output data format is a text-to-text format, where the model is fed text and then asked to produce text. The input sequence will then be tokenized followed by a mapping to embeddings to be finally passed to the encoder. [13] Next, the skip connection adds the input to the output of subcomponents to later be passed to the decoder, which uses autoregressive forms to attend to previous outputs. [13] The output of the final decoder will be fed to a layer with a SoftMax activation function where the weights are shared with the embedding matrix input. It is important to keep in mind that all the heads' outputs are concatenated before proceeding to the next layer and that the layer norm of the T5 base model was removed compared to standard transformer architecture. [13]

The T5 architecture is represented in the below diagram:

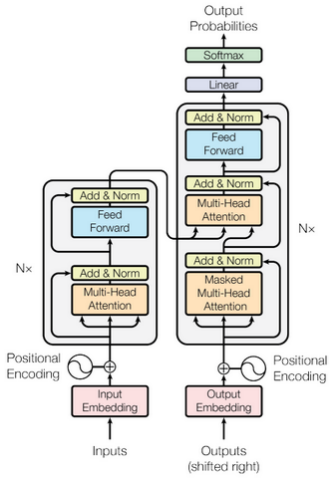

Fig. 2 T5 model architecture [13]

B. Dataset Definition

The WebNLG 3.0 dataset represents sets of RDF triples, ranging from 1 to 7 per entry, and mapped to an English text.

The RDF triples were extracted from DBpedia, a platform that allows users to extract structured content from the information on Wikipedia [14]. They adopt a subject-predicate-object structure with each component separated by vertical separators: "Arròs_negre | country | Spain",

where *Arròs_negre* is the subject, *country* is the predicate, and *Spain* is the object [14].

Each entry in the dataset is a tree characterized by its category, size, and shape type: sibling (triples have a shared subject), chain (object of one triple is the subject of the other), or mixed (both types) [14]. Moreover, each entry consists of three sections: <originaltripleset>, <modifiedtripleset>, and <lex>. Where the first represents the original tripleset as extracted from DBpedia, the second represents the triples after being modified by crowdworkers, and the third represents the natural language text mapped to the triples (each tripleset can have more than 1 natural language description).

The overall dataset is stored in a hierarchical format where everything is wrapped under the root tag <benchmark>, followed by <entries> tag which wraps all the <entry> units.

The following figure describes the organization of a single WebNLG data entry:

```
</entry>
  <entry category="SportsTeam" eid="Id2" shape="(X (X) (X) (X))" shape_type="sibling" size="3">
    <originaltripleset>
      <otriple>1._FC_Köln | manager | Peter_Stöger</otriple>
      <otriple>1._FC_Köln | season | 2014–15_Bundesliga</otriple>
      <otriple>1._FC_Köln | capacity | 50000</otriple>
    </originaltripleset>
    <modifiedtripleset>
      <mtriple>1._FC_Köln | manager | Peter_Stöger</mtriple>
      <mtriple>1._FC_Köln | season | 2014–15_Bundesliga</mtriple>
      <mtriple>1._FC_Köln | numberOfMembers | 50000</mtriple>
    </modifiedtripleset>
    <lex comment="good" lid="Id1">Peter Stöger is manager of 1. FC Köln who were in the 2014–15
        Bundesliga season and have 50000 members.</lex>
    <lex comment="good" lid="Id2">The manager of 1. FC Koln is Peter Stoger. They have 50000 members and
        played the 2014–15 season in the Bundesliga.</lex>
    <lex comment="good" lid="Id3">1. FC Köln were in the 2014–15 Bundesliga season, its manager is Peter
        Stoger and have 50000 members.</lex>
```

Fig. 3 Organization of a WebNLG data entry (snapshot taken from dataset file)

WebNLG is divided into training, development, and test sets as detailed in the table below:

	Train	Dev	Test (D2T)
RDF triple sets	13,211	1,667	1,779
Texts	35,426	4,464	5,150
Properties	372	290	220

Table 1: WebNLG data statistics [15]

The test data spans 16 domains seen in training & 3 unseen domains introduced in the testing set, as described in the table below:

Seen	Airport, Astronaut, Building, City, ComicsCharacter, Food, Monument, SportsTeam, University, and WrittenWork; Athlete, Artist, CelestialBody, MeanOfTransportation, Politician, Company
Unseen	Film, Scientist, and MusicalWork

Table 2: WebNLG domains [15]

C. Preprocessing

The WebNLG 3.0 dataset, downloaded from GitHub [source code], was preprocessed in several stages. Since the data is in hierarchal .xml format, we parsed through the data and extracted the triples and their corresponding text to insert into a DataFrame. The connected triples were joined with a double ampersand symbol (&&) to emulate the selection of multiple rows of a table at once. The && connected triples were stored under "input_text" and their corresponding natural language mappings were stored under "target_text" as outputs. The WebNLG dataset was already split for training, validation, and testing, and the previous procedure was implemented on all 3 sets.

Additionally, we performed a type of character normalization on the dataset. We wrote a Python script that removes words with diacritical marks from the dataset and replaces them with their normalized versions while storing the original words in a dictionary. For example, if the dataset contains "São Paulo", it will be modified to "Sao Paulo", while the word "São" is stored in a dictionary as a value of the key "Sao". This has been done since the model cannot handle non-ASCII characters and keeping diacritical marks will result in an output such as "S<unk>o Paulo", where <unk> refers to the accented character. By storing the words in a dictionary, we can then iterate over the output and change the normalized words back to their original form. The output "Sao Paulo" would be reverted to "São Paulo". This script was applied to the DataFrame for training and test runs to examine the output of our model and to avoid any shortcomings in the evaluation of the BLEU score in later steps.

Finally, we tokenize the input and target texts with the T5 tokenizer before being fed into the model for training. The T5 tokenizer is based on Google SentencePiece, which considers input text as Unicode characters. This allows the tokenizer to separate the sequence while preserving certain elements such as spaces or underscores (_) between words, ultimately easing future detokenization. We used the batch_encode_plus() method for large batches of text to convert the tokens into token IDs – a numerical representation of the meaning of the words.

D. Planned Experiments

To begin with, we will preprocess the WebNLG 3.0 dataset using the preprocessing methods described above. After that, the T5 model will be trained and tested accordingly. We will then assess the performance of the model using the BLEU metric and compare this score with the state-of-the-art model (DataTuner) on this dataset.

The model we will be comparing T5's performance to is DataTuner: a neural end-to-end generation system designed by the Amazon Alexa AI team [1]. It is the successor to Open AI GPT-2: a multilayer autoregressive model constructed with a transformer decoder block [2]. DataTuner aims for higher generalization accuracy by making minimal assumptions about the type of data, the target domain, and the MR's structure.

Firstly, the model's architecture is purposely generic to generalize to different semantic representations and to allow future modifications and improvements with reduced work following data processing [2]. Secondly, its most

accurate version, DataTuner_FC, includes a low-level supervised Semantic Fidelity Classifier (SFC) which detects some generation errors and increases the semantic accuracy, by which is meant to improve the understanding of the information provided [1]. Finally, a beam search decoder, which uses conditional probability to make the best choice and updates ranking every iteration using scores is found in the architecture.

The model also contains multiple types of embeddings indicating relationships between data through their spatial links such as token embeddings that encode semantics about individual words while the other indicates the position of words in the text [2]. This version of the model also includes well fine-grained state embeddings that identify the data item's structural links.

BLEU (bilingual evaluation understudy) is an algorithm used to measure the quality of machine-generated text. This quality is defined as the similarity between the machine's generated text and professional human speech [3]. The more similar the output text is to human speech, the higher the score. BLEU scores are within the 0 to 1 range: 1 represents very high correspondence between output text and human text, and 0 represents great dissimilarity. It is unnecessary to attain a BLEU score of 1 since there are multiple ways to write the same sentence correctly, some of which may not have been included in the dataset [4]. Thus, adding more reference phrases may increase the BLEU score of the model.

BLEU scores are evaluated by comparing the word choice between the output text and the reference text(s). Words present in the machine translation but never in the reference data would negatively affect the BLEU score. This may lead to considering the use of synonyms as an error. Hence, variety in reference data would increase BLEU score reliability.

Groups of consecutive words are also evaluated at once to avoid high scores for texts with correct word choice but wrong sentence structure. This approach guarantees that random placement of correct words would not be rewarded as they only match the reference data when they are placed in a certain order [5].

BLEU scores are efficiently calculated and easily understandable. It is also language-independent and thus very practical for widespread use.

The interpretation of different ranges of the BLEU score is provided in the table below for reference:

BLEU Score	Interpretation
< 10	Almost useless
10-19	Hard to get the gist
20-29	The gist is clear, but has significant grammatical errors
30-40	Understandable to good translations
40-50	High quality translations
50-60	Very high quality, adequate, and fluent translations
>60	Quality often better than human

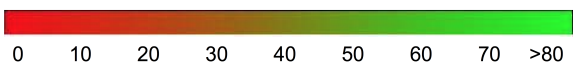

Table 3: BLEU score interpretation [16]

III. EXPERIMENTS

For training the model, we used the recommended Adafactor optimizer, a stochastic optimization method based on the popular Adam optimizer that is able to reduce memory usage significantly [17]. This memory usage, regarding an *n x m* matrix, is reduced from a quadratic space complexity of O(*nm*) to a linear O(*n+m)*.

We randomized the occurrence of the data entries in the training set to ensure optimal training and trained our model over 5 epochs. Each epoch took about 30 minutes to complete, resulting in a total of 2.5 hours of training. Validation took another 30 minutes. We then plotted the losses and studied the decreasing error as a function of epochs.

Before running our trained model on the entire test set, we generated single sentences to further investigate potential problems and track the model's sentence generation. We came across the following problems:

1. The generated sentences were truncated (stopping abruptly in mid-sentence).
   → After further analyzing the parameters of the encode and decode methods, the reason was that the encoder and decoder had different initial "max-length" parameter values. The decoder had a shorter max-length, hence it stopped before reaching the EOS token which indicates the end of the sentence.

2. The model was replacing characters with diacritical marks with <unk> in the generated sentences.
   →This issue was later resolved with the implementation of a type of character normalization, which we mentioned in the preprocessing steps above.

3. There was loss of information in the generated sentences for most triplesets of size >3.
   →We addressed this problem after further analysis of the generate() method parameters and adding beam search. Which reduces the risk of missing hidden high-probability word sequences by keeping a specified number of most likely hypotheses at each step and make a final decision by choosing the overall highest probability. We also used the early stopping parameter to ensure that the generation is finished when all the beam hypotheses reach the EOS token </s>.

As we ran our model on the test set, we kept track of the model's predictions and the corresponding input triples. Given that every input triple has several reference sentences included in the dataset, we maintained a dictionary in which the keys were the input triples, and the values were the array of sentences corresponding to each input triple.

Next, we imported the BLEU score module from the nltk library. Since the BLEU score code requires a certain format for the input, we prepared the data accordingly: first, we split each sentence into separate words in a list. We

created two arrays that would store the hypothesis and references, respectively, for every input triple. We apply the corpus_bleu function on these arrays and obtain our results.

IV. Results (view appendix for examples)

When training the model and checking the sentence generation for single testing triple sets at a time, we noticed decent semantic fidelity. Triples pertaining to most domains, and most sizes, generated correct sentences. We noticed some loss in information, incorrect word placement, and/or redundancy in the sentences generated from the MusicalWork and Film domains when the triple size exceeded 4. For other domains, we identified minimal loss of information for some triples over the size of 5 instead of initially 3.

In addition, when replacing the accented characters, there was a minor improvement in results, with the model being able to construct more coherent sentences.

As for the training and validation results, we noticed for the first, a gradual decrease in running loss from 50% at the first epoch to 28% at the fifth, while the validation error was quite low, reaching a decent 4% loss.

Finally, we evaluated the model's performance with the previously mentioned BLEU score, in which our T5-base application on the WebNLG dataset resulted in an average score of 34.8 after calculating the score on multiple training runs and test set predictions.

We compare this score with state-of-the-art "DataTuner" in the table below:

Model	BLEU score
DataTuner	52.9
Fine-tune T5 for Data-to-Text	34.8

Table 4: WebNLG domains [15]

V. Analysis

After generating the previous results, we first look at T5's standalone performance. The model is now able to correctly handle five triples at a time compared to the previous three but exhibits loss of data for six and more. More research should be done in the future to determine whether using a T5 model of a larger size than the base with noticeably more parameters (T5-large, 3b, 11b) would provide better performance.

In addition, the fact that the triplesets of the MusicalWork and Film domains performed more poorly than the rest (size >4) is worthy of analysis. These two domains are part of 3 unseen in training, yet only the Scientist domain triples lead to satisfying results. This might be due to the more complex vocabulary in artistic domains; therefore, a more diversified dataset with additional complex domains in the training set should be implemented in the future.

Regarding the runtime loss, it reached a low of 28% after 5 epochs with a seemingly decreasing trend. The model's optimizer algorithm seems to be converging. However, with each epoch taking 30 minutes of time, it was difficult to realize more than five per training to validate this point. As for the validation error, we obtained a decent 4% loss which we believed was sufficient to claim that the hyperparameters of T5 were well-tuned.

Finally, we compared T5's performance to the state-of-the-art DataTuner's performance on the same dataset. Using Bleu score, we noticed DataTuner outperformed T5 with a score of 52.9 compared to T5's 34.8. This points to DataTuner's ability to better generalize and produce satisfying results with a new dataset while our fine-tuned T5 lags behind in terms of both generation and flexibility when faced with unseen data.

VI. Conclusion

In this paper, we have presented a fine-tuned version of the T5 model and its performance in data-to-text generation, using the WebNLG dataset. The performance of the fine-tuned model was measured using the BLEU score metric. The model achieved a score of 34.8, which is less than that of DataTuner, the state-of-the-art model that achieved a score of 52.9. As per our analysis, the model specifically underperformed with unseen data.

Hence, future work may focus on improving model performance in generalizing to novel and unseen data, as well as testing it on several datasets with more distinct domains. Further interesting expansion of this work may be to study the performance of GPT-3, a successor of the T5 model by OpenAI, on this dataset specifically, and in the task of data-to-text generation generally.

VII. APPENDIX – OUTPUT EXAMPLES

Category: Company
Size=5
Og; Hypermarcas is a pharmaceutical S.A. corporation that produces healthcare products. They employ 10,252 people and have a subsidiary called Mantecorp. They have a net income of $108,600,000.

Gen: Hypermarcas is an S.A. corporation in the pharmaceutical industry. They sell healthcare products and own the subsidiary Mantecorp. They employ 10,252 people and have a net income of $108,600,000.

Category: Scientist
Size=4
Gen:Piotr Hallmann, whose height is 175.26 metres, was born in Gdynia, Poland on the 25th of August, 1987 and weighs 70.308.

Og:Piotr Hallmann was born on August 25, 1987 in Gdynia Poland and his weight is 175.26 cm. His weight is 70.308 kilograms.

Category: University
Size=4
Gen:The Acharya Institute of Technology is located in Soldevanahalli, Acharya Dr. Sarvapalli Radhakrishnan Road, Hessarghatta Main Road, Bangalore – 560090. It is affiliated with Visvesvaraya Technological University.

Og:The Acharya Institute of Technology campus is located in Soldevanahalli, Acharya Dr. Sarvapalli Radhakrishnan Road, Hessarghatta Main Road, Bangalore – 560090 in the state of Karnataka, and it's affiliated to Visvesvaraya Technological University.

Category: MusicalWork
Size=4
Og: Aaron Turner is a post-metal musician from Massachusetts who plays the electric guitar. He started performing in 1995 and is a musician in the Lotus Eaters band.

Gen:Aaron Turner is an electric guitar player from Massachusetts who started performing in 1995 and is a member of the Lotus Eaters band. He is a performer of the Post-metal genre.

Category: Monument
Size=6
Og: The 11th Mississippi Infantry Monument was established in 2000 and categorised as contributing property. It is located in Gettysburg, Adams County, Pennsylvania, United States.

Gen: The 11th Mississippi Infantry Monument is located in Gettysburg, Adams County, Pennsylvania, United States. It was established in 2000 and is categorised as a contributing property.

Category: Company
Size=7

Og: GMA New Media is a mass media company which makes mobile applications. They were founded on Jan 1, 2000 in the Philippines. They are headed up by Felipe Gozon, and are headquartered in the GMA Network Center.

Gen: GMA New Media Inc founded on january 1 2000 is a philippine media company that offers products such as mobile applications. It’s key person is Felipe Gozon and it is located in the GMAA Netwerk center

Category: Musical Work
Size=6
Og: The Train song Mermaid, which is an example of the reggae genre, was released under the record labels of Columbia Records and Sony Music Entertainment. It was written by Espen Lind and Amund Bjørklund and has a total runtime of 3 minutes and 16 seconds.

Gen: the musical genre of Mermaid (Trainsong) is Reggae and it was written by Amund Bjørklund. It has a runtime of 3.16 and was signed to Sony Music Entertainment.

REFERENCES

[1] I. Wigmore, "What is natural language generation?," *TechTarget*, Jul-2021. [Online]. Available: https://www.google.com/amp/s/www.techtarget.com/searchenterpriseai/definition/natural-language-generation-NLG%3Famp%3D1. [Accessed: 07-Oct-2022].

[2] M. Kale and A. Rastogi, "Text-to-text pre-training for data-to-text tasks," *ACL Anthology*, Dec-2020. [Online]. Available: https://aclanthology.org/2020.inlg-1.14/. [Accessed: 07-Oct-2022].

[3] M. Kale and A. Rastogi, "Template guided text generation for task-oriented dialogue," *ACL Anthology*, Nov-2020. [Online]. Available: https://aclanthology.org/2020.emnlp-main.527/. [Accessed: 07-Oct-2022].

[4] I. Groves, "Automatically generating text from structured data," *Amazon Science*, 08-Apr-2021. [Online]. Available: https://www.amazon.science/blog/automatically-generating-text-from-structured-data. [Accessed: 07-Oct-2022].

[5] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic, "Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling," *ACL Anthology*, 2018. [Online]. Available: https://aclanthology.org/D18-1547/. [Accessed: 07-Oct-2022].

[6] H. Harkous, I. Groves, and A. Saffari, "Have your text and use it too! end-to-end neural data-to-text generation with Semantic Fidelity," *ACL Anthology*, Dec-2020. [Online]. Available: https://aclanthology.org/2020.coling-main.218/. [Accessed: 07-Oct-2022].

[7] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "Pretrained language models for TEXT GENERATION: A survey," *arXiv.org*, 13-May-2022. [Online]. Available: https://arxiv.org/abs/2201.05273. [Accessed: 07-Oct-2022].

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research 21*, 28-Jul-2020. [Online]. Available: https://arxiv.org/abs/1910.10683v3?_hsenc=p2ANqtz--1pb_5H15EiMOYFDHJ_q735TeJ1zleTnMMhat0zfi7KZykOmRv2VgkKIWwN5AhgXsiU5Hc. [Accessed: 28-Nov-2022].

[9] S. Wiseman, S. M. Shieber, and A. M. Rush, "Challenges in Data-to-Document Generation," *Harvard Library*, 11-Sep-2017. [Online].Available: https://dash.harvard.edu/bitstream/handle/1/33927876/challenges-inline-appendix.pdf?sequence=1. [Accessed: 28-Nov-2022].

[10] A. Roberts and C. Raffel, "Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer," *Google AI Blog*, 24-Feb-2020. [Online]. Available: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html. [Accessed: 28-Nov-2022].

[11] T5. [Online]. Available: https://huggingface.co/docs/transformers/model_doc/t5. [Accessed: 28-Nov-2022].

[12] "T5-base · Hugging Face," t5-base · Hugging Face. [Online]. Available: https://huggingface.co/t5-base. [Accessed: 28-Nov-2022].

[13] P. Joshi, "Transformers in NLP: State-of-the-art-models," Analytics Vidhya, 23-Jul-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/. [Accessed: 28-Nov-2022].

[14] "What is WebNLG Challenge?," *WebNLG Challenges*. [Online]. Available: https://webnlg-challenge.loria.fr/. [Accessed: 28-Nov-2022].

[15] T. C. Ferreira, C. Gardent, N. Ilinykh, C. van der Lee, S. Mille, D. Moussallem, and A. Shimorina, "The 2020 bilingual, bi-directional webnlg+ shared task: Overview and evaluation results (webnlg+ 2020)," *ACL Anthology*, 2020. [Online]. Available: https://aclanthology.org/2020.webnlg-1.7/. [Accessed: 28-Nov-2022].

[16] "Evaluating models | AutoML Translation Documentation," *Google Cloud*. [Online]. Available: https://cloud.google.com/translate/automl/docs/evaluate#bleu. [Accessed: 28-Nov-2022].

[17] N. Shazeer and M. Stern, "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost," *Cornell University*, 11-Apr-2018. [Online]. Available: https://arxiv.org/abs/1804.04235. [Accessed: 28-Nov-2022].