Investigating Disfluency Generation for the Creation of Humanlike Utterances in Conversation

Custom Project, Mentor: Siyan Liu

Ayushi Gupta[*]
Computer Science
Stanford University
`ayushig@stanford.edu`

Alice Zhang
Computer Science
Stanford University
`alicebz@stanford.edu`

Zuyi Liz Zhao
Computer Science
Stanford University
`zuyi@stanford.edu`

Abstract

Developing conversational agents that communicate in a natural and human-like manner is a challenging task in natural language processing (NLP). Existing NLP models often produce artificially fluid utterances that lack the disfluencies of natural speech, such as interruptions and repetitions. Our explorations reveal a discrepancy between the goals of generating human-like disfluencies and disfluency generation models intended to augment data for disfluency detection models. Current models primarily focus on detecting complex disfluencies and avoid simple ones, which is not reflective of human behavior. In this paper, we present a novel application of an existing BiLSTM-based disfluency generation NLP model, trained on the Switchboard Dialog Act Corpus (SwDA) dataset, to address this gap. Furthermore, we integrate our disfluency generation model with DialoGPT and SpeechT5 for Text-to-Speech (TTS) to create SpokenDisfloGPT, a vocal chatbot.

1 Introduction

One of the ultimate objectives of dialogue systems is to create coherent utterances indistinguishable from human conversation, and many current conversational frameworks achieve some spectrum of this goal [1]. The pursuit of such artificially-generated language development has therefore spawned many disfluency detection and classification methods to remove disfluency from input data and increase ease and effectiveness in speech recognition and processing [2].

If our goal has been to mimic human behavior, however, then many modern agents have overshot the window of fluidity exhibited by humans and have returned full circle to artificiality. Technology has impressive potential towards greater access to human-oriented services but cannot properly do so with obviously artificial responses. With that in mind, we set out to investigate the generation of disfluent text from fluent inputs to create more natural utterances similar to typical human conversation through the incorporation of disfluency that promotes natural language but does not extend to incoherence.

2 Related Work

Disfluency detection data augmentation tools have been developed to improve the performance of disfluency detection models. The LARD tool as developed by Passali et al. [3] is a non-neural

[*]Authors are ordered alphabetically by last name.

approach that performs disfluency detection by annotating a large corpus of speech data. A newer version of LARD tool, while not publicly available, utilizes natural language processing techniques to generate disfluencies [4]. Yang et al. [5] proposed a neural approach for data augmentation in disfluency detection, using two-part model to generate disfluent sentences. These methods have shown promising results in improving the accuracy of disfluency detection models.

Despite the successes of such models with natural generation, however, we must recognize that most augmentation methods explored by other researchers prioritize disfluency augmentation to improve accuracy over artificial language tasks such as disfluency detection, classification, and extraction, and put less emphasis on disfluency generation for human consumption with the intent of eventually mimicking human-to-human conversation. As we cannot determine the successes of such models on natural generation per our specific purposes, we aim to develop a model that considers this issue as the main focus of its generated disfluencies.

There also exists work in combining conversational GPT models with TTS models to create vocal chatbots. To put our model into practical use, we extensively studied such models and integrated Microsoft's SpeechT5 and DialoGPT (large) models to our proposed model which serves as our conversational vocal chatbot with human-like disfluency generating capabilities not seen in any existing chatbots.

3 Approach

3.1 Baselines.

LARD. LARD [3] is an automated method for generating a vast amount of synthetic disfluencies based on existing corpora containing fluent text. LARD can handle repetitions, replacements, and restarts. Passali et al. categorize a repetition as a sequence of words preceding a fluent utterance that consists of the same content, a replacement as a disfluent phrase being corrected or replaced by the fluent phrase, and a restart as a disfluency that cause a speaker to begin a completely new utterance rather than correct or replace the disfluency in the original sentence. Passali et al. have developed a newer version of LARD [4], but it is not publically available. The newer LARD model utilizes a BERT-based language model to select the reparandum candidate of a disfluency rather than randomly selecting it as the publically available LARD [3] model does. For our research purposes we only make use of LARD's ability to produce repetitions and replacements as LARD's restart algorithm requires the use of two or more fluent sequences into one disfluent output, which could affect our quantitative metrics when averaging disfluency frequency and length over the resulting utterances and result in more variance during model comparison.

Planner-Generator Model. The Planner-Generator Model [5] generates augmented disfluent texts by leveraging neural methods. The Planner decides on where to insert disfluent segments and the Generator generates the corresponding disfluent segments based off the results from the Planner. The Planner incorporates a bidirectional LSTM encoder in its architecture while the Generator consists of a bidirectional LSTM encoder and an RNN decoder. The Planner uses the softmax of its hidden states to compute decision probabilities which the Generator then uses to determine the input value of each step of its decoder. The final disfluent sentence is therefore generated based on these decision probabilities where it would either directly copy a step of the input fluent sentence as the output sentence or generate a sequence of words to act as a disfluency or reparandum for the resulting utterance. Although Yang et al. originally trained the model on the Switchboard-1 Telephone Speech Corpus, we train the model on the Switchboard Dialog Act Corpus instead. See **Appendix 7.4** for a visualization of the model architecture.

3.2 Our Models.

Disfluency Generator. Transformers have traditionally outperformed RNNs for a variety of NLP tasks (with one relevant example being Karita et al. [6] finding greater performance of Transformer models than RNNs for speech applications through higher accuracy and faster convergence). With that in mind, we chose to change the Planner-Generator (PG) encoder from an RNN to a transformer template that was published alongside the RNN structures. As Yang et al. [5] chose to use RNN models over transformers as per resulting performance, we attempted to improve performance of their published transformer encoder and decoder classes by adding another normalization layer to both so as to reduce internal variance, improve convergence, and increase generalization of the model during training. We ran this model with 6 transformer layers and 8 transformer heads in the encoder as those were the default values provided by Yang et al., and we first wanted to measure transformer effectiveness without altering too many other variables. To further explore the effect transformer influence would have on a model's resulting performance, we then chose to implement transformers for both the encoder and decoder classes of our second transformer model, decreasing the number of layers and heads that were run on the previous model to 3 layers and 4 heads so as to prevent overfitting with the inclusion of two complex transformers.

Disfluency Generator with LARD. While the baseline PG model is effective in generating simple repetitions and interruptions, the limited number of complex disfluencies in the training set leads to difficulty in generating more complex disfluencies. To address this issue, we incorporated a heuristic replacement generator from LARD, which allowed for the generation of more complex disfluencies involving the correction of a previous word. LARD's replacement generator is called on top of the PG model's disfluent output at a probability of 11%, which is the calculated percentage of complex disfluencies, defined as any disfluency that is longer than two tokens in the SwDA test set.

SpokenDisfloGPT. In addition to developing our disfluency generation model, we integrate it with a GPT model fine-tuned on multiturn conversations and a TTS model to create a vocal chatbot we call SpokenDisfloGPT. The combination of these models allows for a seamless and natural conversation experience between the chatbot and the user. The conversational GPT model provides high-quality responses to the user's queries, while our disfluency generation model adds a layer of human-like disfluencies to the chatbot's speech, making it sound more natural and realistic. Then the TTS model converts the disfluent output into audio format, which allows SpokenDisfloGPT to produce a spoken response to the user's queries. We designed SpokenDisfloGPT to evaluate the performance of our disfluency generation model and demonstrate its practical use. See **Appendix 7.5** for an example flow of SpokenDisfloGPT.

4 Experiments

4.1 Data

The Switchboard Dialog Act Corpus (SwDA)[7] corpus extends the Switchboard-1 Telephone Speech Corpus, Release 2 (Switchboard) [8]. See **Appendix 7.1** for a data sample. Switchboard is a widely used disfluency detection dataset, although as Passali et al. [3] have noted, it and SwDA by extension are highly imbalanced, with less than 6% of tokens being disfluent. We do not use data augmentation to correct this imbalance because our model aims to imitate human-like disfluency generation, which should reflect a low percentage of disfluencies assuming that the SwDA corpus is representative of human disfluency. Following the conventions of the Dysfluency Annotation Stylebook for the Switchboard Corpus, we used the SwDA disfluency annotations to identify and separate fluent and disfluent segments of each utterance in order to create fluent inputs for our model at test time.

We perform the following preprocessing steps on our dataset:

1. Normalize the text by converting to lowercase, removing punctuation other than apostrophes, and tokenizing using the Natural Language Toolkit (NLTK).
2. Use the NLTK part-of-speech (POS) tagger to obtain POS information. We follow the part-of-speech tags used in the Penn Treebank Project. See **Appendix 7.2** for a full list.
3. Remove instances where the entire utterance is disfluent.
4. Format the data to fit the expected input format of our disfluency generator model. See **Appendix 7.3** for how the data is formatted.

4.2 Evaluation Method

Following Yang et al.[5], we use BLEU to compare the disfluent sentences generated by our model with reference disfluent sentences reserved for testing from the SwDA dataset. However, given that the disfluency generation task is relatively open-ended, we note that BLEU does not capture the full scope of our model's quantitative performance. Therefore, we leverage novel metrics to evaluate the quality of our generated disfluencies. Specifically, we evaluate the quality of our generated disfluencies by tracking the average number of separate disfluencies per utterance, the average number of disfluent words per disfluency, the top 3 insertion indices for each separate disfluency, and the top 3 grammatical patterns of each disfluency for both our generated outputs and the reference test set (including and excluding disfluencies of length 1). We then compare these metrics to assess the performance of our model. To fully account for the subjective and open-ended nature of disfluencies, we incorporate qualitative evaluation methods such as naturalness ratings and error analysis.

4.3 Experimental Details

Disfluency Generator. The Planner-Generator model provided by Yang et al. [5] acted as our baseline for disfluency generation and was developed with an architecture consisting of a bidirectional LSTM encoder for the Planner portion of the model as well as a bidirectional LSTM encoder and an RNN decoder for the Generator. It was run with a learning rate of 0.001, a gradient vector maximum norm of 0.1, and a layout weight of 1 with a batch size of 64 and an Adam optimizer. We trained this model for the default value of 30 epochs, which took a little over an hour. Each epoch trained around 156 seconds each, with the shortest training time being 154 seconds and the longest being 158 seconds.

Our first transformer model consisted of a transformer encoder for the Planner and Generator with two normalization layers in each encoder layer. To keep parameters consistent with Yang et al.'s baseline, this model was run on a 0.001 learning rate with a batch size of 64 and an Adam optimizer. The transformers consisted of the default values of 6 layers and 8 heads. We trained this model for 30 epochs and found that epoch took around 762 seconds each to train, with the shortest training time being 759 seconds and the longest being 765 seconds. We implemented our second transformer model with a transformer encoder for the Planner and a transformer encoder and decoder for the Generator with two normalization layers in each encoder layer and three normalization layers in each decoder layer. We ran this model on a 0.001 learning rate with a batch size of 64 and an Adam optimizer but decreased the number of layers and heads to 3 layers and 4 heads. We trained for 30 epochs and saw an average training time for each epoch of 462 seconds, with the shortest training time being 459 seconds and the longest being 468 seconds. We then incorporated LARD's replacement generator [3] to all three models' outputs.

In accordance to the metrics set by Yang et al. [5], the second transformer model (which we will now reference as the PG+Transformers Enc&Dec) began to show highest accuracy after around 10 epochs and lowest layer loss (the loss found during regularization of the model's parameters at each layer) around 7 epochs in but was comparable to the baseline for target loss (the loss found when comparing the model's generated disfluent sentences with the reference disfluent sentences from the SwDA dataset). For the test set it was either comparable to the first transformer model (PG+Transformer Enc) or performed worse after 5 epochs. The baseline was comparable to PG+Transformer Enc results over the training set but saw highest accuracy and lowest loss for the duration of the 30 epochs on the test set. The PG+Transformer Enc saw lowest accuracy and highest loss for the duration of training on the training set but was comparable to PG+Transformers Enc& Dec for the test set on most metrics aside from layer accuracy (the accuracy found for model regularization between layers) and layer loss, in which it was more comparable to baseline results. See **Appendix 7.6** for visualizations plotted over 25 epochs due to logging issues for the last 5 epochs with PG+Transformer Enc's training.

SpokenDisfloGPT. To integrate our disfluency generation model into a vocal chatbot, we thoroughly evaluated several conversational GPT models and TTS models. Our evaluation process was primarily qualitative in nature. We considered Microsoft's DialoGPT, (including small, medium, and large variants) [9], Facebook's BlenderBot (specifically the 90M-small variant) [10], and OpenAI's ChatGPT API. Additionally, we experimented with Microsoft's SpeechT5 for TTS model [11] and the gTTS python module that utilizes Google Translate's TTS API. Due to compute and funding limitations that restricted the use of ChatGPT [12] and the larger variants of BlenderBot, we chose to use the large variant of DialoGPT. It is important to note that the responses generated by DialoGPT are typically brief, which restricts the complete potential of disfluency generation. However, this aligns well with our model as it was primarily trained on short utterances. Following our qualitative assessment, we utilized SpeechT5 despite its occasional flaws in duplicating audio, as the SpeechT5 model's human-like voice quality was considered superior to that of Google Translate's TTS API. We also explored the possibility of a graphical user interface for the vocal chatbot using Flask, although due to time constraints, we were unable to fully develop this aspect of the project.

4.4 Results

Model	**Good Generated Disfluency**	**Bad Generated Disfluency**
LARD Repetitions & Replacements	which *seems to* seems to be	plastic cans and *meth I am sorry* glass
PG Model BRNN Enc	*some of the* some of it runs off right away in to the streams and rivers	*uh* back in those days they also use to give you *uh uh* good for *uh you know* you and gas and things like this
PG Model BRNN Enc w/ LARD	*that 's* that 's what *single well I actually meant* i was *uh*	*i 'm* i 'm hoping that this *uh* the flex will *uh uh* things a little bit
PG Model Transformer Enc	*i i 've never* i 've never seen an actual capsule	like *i* i say *the* the *acid* acid rock
PG Model Transformers Enc&Dec	*i mean* they played all the irish jigs and so forth	*and* i just got through reading *uh by* by *uh and* michener
PG Model Transformers Enc&Dec w/ LARD	and if you keep up with a consistent *you know* pace	our biggest electrical plants *in rhode island island island island island island* in rhode island

Dataset/Model	**BLEU**	**Avg. Num Disfl**	**Top 3 Num Disfl Dist**	**Avg. Disfl Len**
SwDA Test Set Full	N/A	0.4952	(0, 2800), (1, 1220), (2, 287)	0.5726
SwDA Test Set Filtered	N/A	1.3512	(1, 1220), (2, 287), (3, 72)	1.5622
LARD Repetitions	0.3723	1	(1, 4258)	1.678
LARD Replacements	0.3608	1	(1, 1642)	5.8825
PG Model BRNN Enc	**0.74589**	1.087	(1, 1343), (2, 61), (3, 19)	1.262
PG Model BRNN Enc w/ LARD	0.7407	1.0789	(1, 1351), (2, 58), (3, 17)	**1.618**
PG Model Transformer Enc	0.7400	1.0728	(1, 1339), (2, 78), (3, 7)	1.1144
PG Model Transformers Enc&Dec	0.7287	**1.232**	**(1, 1193)**, **(2, 189)**, **(3, 39)**	1.1118
PG Model Transformers Enc&Dec w/ LARD	0.7244	1.2204	(1, 1209), (2, 178), (3, 36)	1.4268

Dataset/Model	Top 3 Insertion Indices (I-I)	Top 3 POS Pattern (P-P)	Top 3 I-I Dist	Top 3 P-P Dist
SwDA Test Set Full & Filtered	0, 4, 5	CC, NN, RB	(0, 1351), (4, 85), (5, 72)	(‘CC’, 349), (‘NN’, 259), (‘RB’, 191)
LARD Repetitions	0,1,2	NN, JJ, RB	(0, 2018), (1, 583), (2, 410)	(‘NN’, 872), (‘JJ’, 353), (‘RB’, 196)
LARD Replacements	1, 0, 2	NN PRP VBP, NN DT NN, JJ NN	(1, 370), (0, 309), (2, 265)	(‘NN PRP VBP’, 23), (‘NN DT NN’, 20), (‘JJ NN’, 18)
PG Model BRNN Enc	0, 10, 14	**CC, NN, RB**	(0, 1421), (10, 10), (14, 10)	(‘CC’, 566), (‘NN’, 188), (‘RB’, 117)
PG Model BRNN Enc w/ LARD	0, 3, 5	**CC, NN, RB**	(0, 1345), (3, 20), (5, 18)	(‘CC’, 523), (‘NN’, 174), (‘RB’, 110)
PG Model Transformer Enc	**0, 4, 6**	**CC, NN, RB**	(0, 1409), (4, 16), (3, 16)	(‘CC’, 523), (‘NN’, 259), (‘RB’, 143)
PG Model Transformers Enc&Dec	0, 7, 3	CC, RB, NN	(0, 1388), (7, 44), (3, 39)	(‘CC’, 804), (‘RB’, 208), (‘NN’, 122)
PG Model Transformers Enc&Dec w/ LARD	**0, 4, 7**	CC, RB, NN	(0, 1316), (4, 47), (7, 46)	(‘CC’, 751), (‘RB’, 199), (‘NN’, 119)

Dataset/Model	Top 3 POS Pattern for Longer Disfl (P-P-Long)	Top 3 P-P-Long Dist
SwDA Test Set Full & Filtered	PRP VBP, JJ VBP, CC NN	(‘PRP VBP’, 134), (‘JJ VBP’, 63), (‘CC NN’, 42)
LARD Repetitions	DT NN, DT VBZ, PRP VBP	(‘DT NN’, 51), (‘DT VBZ’, 32), (‘PRP VBP’, 31)
LARD Replacements	Identical to P-P	Indentical to P-P Dist
PG Model BRNN Enc	JJ VBP, PRP VBZ, PRP VBP	(‘JJ VBP’, 66), (‘PRP VBZ’, 46), (’PRP VBP’, 34)
PG Model BRNN Enc w/ LARD	JJ VBP, PRP VBZ, PRP VBP	(‘JJ VBP’, 64), (‘PRP VBZ’, 43), (‘PRP VBP’, 29)
PG Model Transformer Enc	**PRP VBP**, **JJ VBP**, PRP VBZ	(‘PRP VBP’, 34), (‘JJ VBP’, 34), (‘PRP VBZ’, 33)
PG Model Transformers Enc&Dec	**PRP VBP**, **JJ VBP**, PRP VBZ	(‘PRP VBP’, 63), (‘JJ VBP’, 24), (‘PRP VBZ’, 13)
PG Model Transformers Enc&Dec w/ LARD	PRP VBP, JJ VBP, PRP VBZ	(’PRP VBP’, 62), (‘JJ VBP’, 23), (‘PRP VBZ’, 13)

Qualitative Results. We observed that for good results, the model tended to place disfluencies in natural locations within the sentence. Furthermore, the model appeared to use slightly complex disfluent phrases, rather than simple interruptions or hesitations, resulting in more varied output. However, we also observed a common problem in NLP models when generating text in general, which is the repetition of tokens. While the model was able to generate natural-sounding disfluencies in many cases, there were instances where it produced repetitive and unnatural phrases. In addition to the issues mentioned above, we also observed other unexpected behavior with the PG-based models, such as incorrect annotations for the disfluencies that it had generated itself (wrongly indicating that a disfluency is a fluent token) and special tokens that are included and decoded in the final output.

Quantitative Results. The SwDA Test Set Filtered is a subset of the SwDA Test Set that only includes inputs that contain at least one disfluency. This filtering ensured that we could more adequately test our disfluency generation models against the presence and accuracy of disfluencies in the data. In terms of BLEU score, the vanilla PG model developed by Yang et al. [5] but trained on the SwDA outperforms our other models. However, according to some of our other metrics, the PG+Transformer Enc, the PG+Transformers Enc&Dec, and the PG+Transformers Enc&Dec w/ LARD models perform better. PG+Transformers Enc&Dec achieves an average number of disfluencies per utterance similar to the filtered SwDA test set, and correspondingly has the most similar distribution. PG+Transformer Enc and PG+Transformers Enc&Dec achieve similar insertion indices for disfluencies. Many models achieve a similar parts-of-speech pattern when looking at single-token disfluencies.

5 Analysis

Trends. There is a significant predominance of single word disfluencies and a tendency towards one disfluency per utterance in the disfluencies generated by our model. However, this observation is consistent with the prevalence of single-word interregnums, such as “uh” or “um”, in human disfluencies. Additionally, the occurrence of multi-word disfluencies was found to be relatively low, in line with the ground-truth data. It is worth noting that the PG model outputs exhibited a low number of separate disfluencies per utterance, which could potentially be attributed to the segmentation of utterances within the dataset.

Since LARD is designed heuristically to only insert a disfluency at one location within a given utterance, all of its generated disfluent sentences will only have one disfluency each. Additionally, because LARD’s replacement generator is meant to repair and correct a misspoken word in the given sentence, the disfluencies tend to be longer.

Dataset Limitations. The limitations of our dataset must be acknowledged in order to properly contextualize our findings. Firstly, the dataset is segmented by dialogue tags rather than by sentence, which results in some inaccuracy in the insertion indices of disfluencies on a sentence level. Additionally, utterances with less than five tokens may not provide enough meaningful context for the model to properly generate disfluencies. Finally, the dataset does not contain many complex disfluencies, which limits the ability of the model to learn how to effectively generate these types of disfluencies.

6 Conclusion

In this work, we explored several architecture variants of the Planner+Generator (PG) model first proposed by Yang et al. [5], by using transformers instead of BiLSTMs and performing hypertuning, as well as using heuristic disfluency generation tools. While we saw dataset limitations in terms of dialogue tagging and underperformance from our models compared to the SwDA-trained PG model in terms of BLEU score, many of our models overperform the vanilla PG model in our novel metrics for disfluency analysis. Moreover, we demonstrated that pre-existing disfluency generation models, which were designed for the purposes of data augmentation for disfluency detection training, are not fully aligned with the aim of generating human-like disfluencies. Finally, we integrate the SwDA-trained PG model with DialoGPT and SpeechT5 for TTS to create SpokenDisfloGPT, a vocal chatbot that serves to both evaluate our disfluency generation model and to demonstrate its effectiveness. While this work constitutes an investigation of possible disfluency generation models, we have presented a novel application for disfluency generation outside of data augmentation. Our hope is to contribute to solving the challenge of creating believable conversational agents with human-like disfluencies.

References

[1] Tingchen Fu, Shen Gao, Xueliang Zhao, Ji-rong Wen, and Rui Yan. Learning towards conversational ai: A survey. *AI Open*, 3:14–28, 2022.

[2] Yang Liu, Elizabeth Shriberg, Andreas Stolcke, and Mary Harper. Comparing hmm, maximum entropy, and conditional random fields for disfluency detection. *Ninth European Conference on Speech Communication and Technology*, 2005.

[3] T. Passali, T. Mavropoulos, G. Tsoumakas, G. Meditskos, and S. Vrochidis. Lard: Large-scale artificial disfluency generation, 2022.

[4] T. Passali, T. Mavropoulos, G. Tsoumakas, G. Meditskos, and S. Vrochidis. Artificial disfluency detection, uh no, disfluency generation for the masses, 2022.

[5] Jingfeng Yang, Diyi Yang, and Zhaoran Ma. Planning and generating natural and diverse disfluent texts as augmentation for disfluency detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1450–1460, Online, November 2020. Association for Computational Linguistics.

[6] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. A comparative study on transformer vs rnn in speech applications, 2019.

[7] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Marie Meteer, and Carol Van Ess-Dykema. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–371, 2000.

[8] John J. Godfrey and Edward Holliman. Switchboard-1 Release 2, 2021.

[9] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation, 2020.

[10] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's wmt19 news translation task submission, 2019.

[11] Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. SpeechT5: Unified-modal encoder-decoder pre-training for spoken language processing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5723–5738, May 2022.

[12] OpenAI, Nov 2022.

7 Appendix

7.1 SwDA Hugging Face Data Instance

Taken from the Hugging Face dataset card information for the SwDA dataset. See the dataset card for more information about the data fields.

{'act_tag': 115, 'caller': 'A', 'conversation_no': 4325, 'damsl_act_tag': 26, 'from_caller': 1632, 'from_caller_birth_year': 1962, 'from_caller_dialect_area': 'WESTERN', 'from_caller_education': 2, 'from_caller_sex': 'FEMALE', 'length': 5, 'pos': 'Okay/UH ./.', 'prompt': 'FIND OUT WHAT CRITERIA THE OTHER CALLER WOULD USE IN SELECTING CHILD CARE SERVICES FOR A PRESCHOOLER. IS IT EASY OR DIFFICULT TO FIND SUCH CARE?', 'ptb_basename': '4/sw4325', 'ptb_treenumbers': '1', 'subutterance_index': 1, 'swda_filename': 'sw00utt/sw_0001_4325.utt', 'talk_day': '03/23/1992', 'text': 'Okay. /', 'to_caller': 1519, 'to_caller_birth_year': 1971, 'to_caller_dialect_area': 'SOUTH MIDLAND', 'to_caller_education': 1, 'to_caller_sex': 'FEMALE', 'topic_description': 'CHILD CARE', 'transcript_index': 0, 'trees': '(INTJ (UH Okay) (. .) (-DFL- E_S))', 'utterance_index': 1}

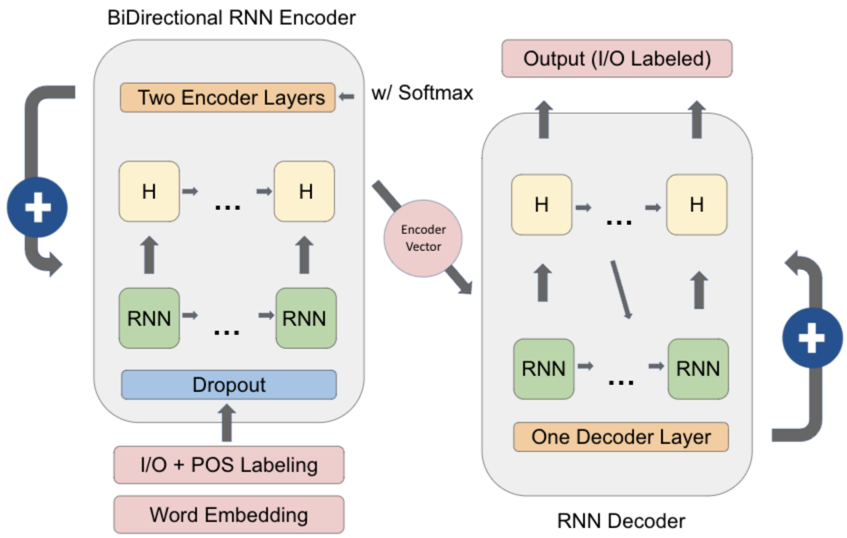
7.2 Alphabetical list of part-of-speech tags used in the Penn Treebank Project

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun

7.3 Sample Data Formatting

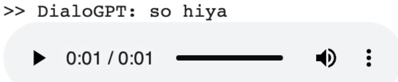
oh	they	did	n't	say	they	did	n't	say	what	though
IN	PRP	VBD	RB	VB	PRP	VBD	RB	VB	WP	IN
I	I	I	I	I	O	O	O	O	O	O

7.4 PG Model Architecture

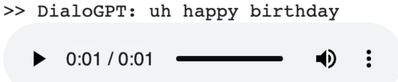


7.5 Flow of SpokenDisfloGPT

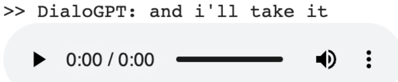
```
>> User: hello
>> DialoGPT: so hiya
```




```
>> User: it's my birthday tomorrow
>> DialoGPT: uh happy birthday
```

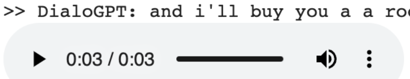



```
>> User: i want to invite you to my party but you will have to buy me an awesome gift
>> DialoGPT: and i'll take it
```




```
>> User: what are you going to give me for my birthday
>> DialoGPT: a a free trip to the moon
```


```
>> User: cool! are you going to buy me a rocket ship too?
>> DialoGPT: and i'll buy you a a rocket ship for your birthday
```



7.6 Training and Test Set Statistics During Model Training

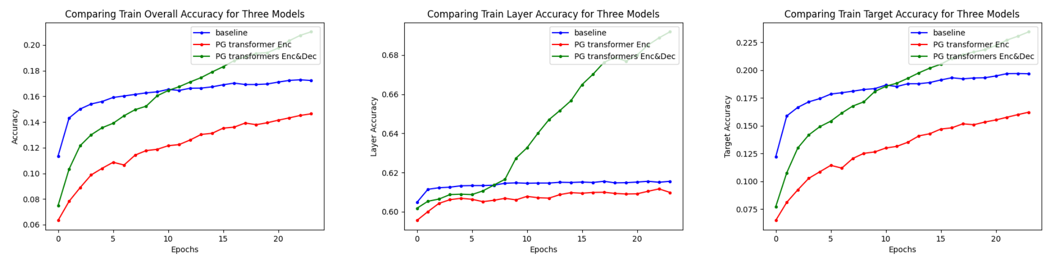


Figure 1: Accuracy comparison over 25 epochs for the training set

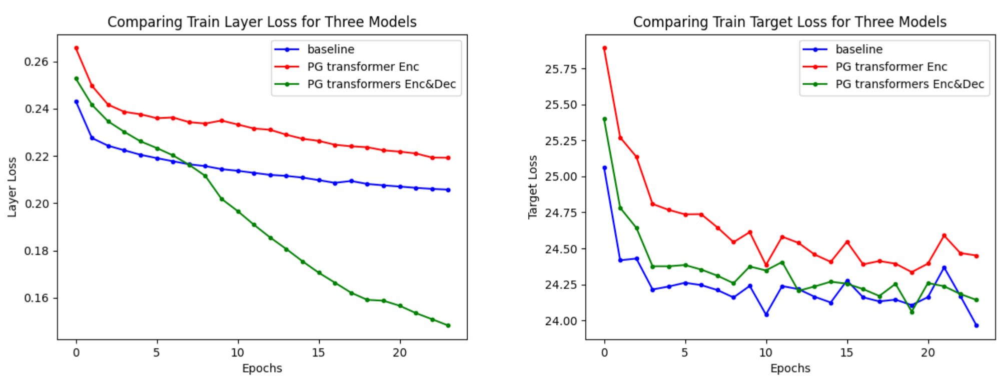


Figure 2: Loss comparison over 25 epochs for the training set

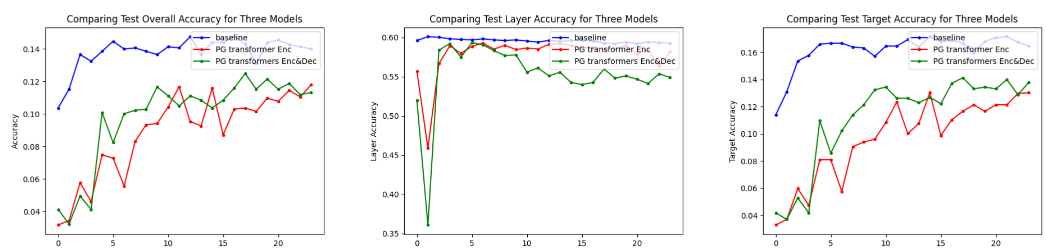


Figure 3: Accuracy comparison over 25 epochs for the test set

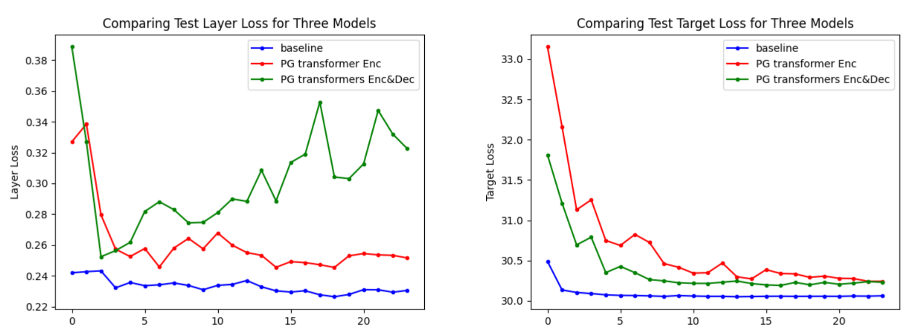


Figure 4: Loss comparison over 25 epochs for the test set