Detecting human written text from machine generated text by modeling discourse coherence

Ganesh Jawahar
Department of Comptuer Science
University of British Columbia
ganeshjw@cs.ubc.ca

Abstract

In this project, we focus on building a classifier that can accurately detect human written text from machine generated text. Such a classifier can be used in mitigating the risks posed by adversaries who can use the state-of-the-art language models (e.g. GPT-2) to create misinformation (e.g. fake news). Existing work treat the text to be detected as a flat sequence of tokens, thereby ignoring the sentence level structural information present in the text. We posit that the features based on sentence-sentence and paragraph-paragraph interactions can be very useful for this task. To this end, we employ various discourse coherence models, ranging from entity grid based models (basic) to neural models (advanced). This report presents our detection results and analysis.

1 Introduction

Neural language models (NLMs) have become ubiquitous in the field of natural language processing (NLP). NLMs are trained on large amounts of naturally occurring text by predicting a word given the surrounding words from a text sequence (e.g., sentence). NLMs have a wide range of use cases such as a starting point for learning supervised models (Peters et al., 2018; Devlin et al., 2019), analysis of dataset biases (Solaiman et al., 2019), generation of radiology reports (Liu et al., 2019), aiding literary art (AIW) and code auto-completion (TabNine). Unfortunately, NLMs are prone to misuse by adversaries, such as generating fake news (Zellers et al., 2019; Keskar et al., 2019), generating fake product reviews (Adelani et al., 2020), impersonating others in email, automating spamming/phishing and abusive social media content production (Radford). Hence, NLP researchers are focusing on building automated methods to prevent such misuse of NLMs.

The popular approach to solve this problem is to build automated ML based models that can detect text generated by NLMs from human written text. A good detector can help in removing harmful generated content in vulnerable platforms such as social media, email clients and e-commerce websites. The detector can be built using four different approaches. First, a detector can be a simple classifier (e.g., logistic regression) trained from scratch (Solaiman et al., 2019). Second, a detector can use pretrained generative model to detect generations from itself or similar models (e.g., using probabilities assigned by the model to strings of text) without additional training (Gehrmann et al., 2019; Zellers et al., 2019). Third, a detector can be obtained by finetuning a language model to detect itself or similar models (Solaiman et al., 2019). Fourth, a human can collaborate with an interpretable ML detector to perform the classification.

Existing work on building detection models treats the text to be classified as a flat sequence of tokens, throwing away the sentence level structural information. We believe that the sentence level cues are valuable for this problem as they capture how NLMs handle coherency. In this project, we unpack these sentence level cues by employing discourse coherence models. We experiment with both non-neural based approaches (such as entity grid (Barzilay and Lapata, 2008) and entity graph (Guinaudeau and Strube, 2013)) and neural based approaches (such as sentence average model, paragraph sequence model (Lai and Tetreault, 2018)). We work with the publicly available dataset containing the text generated by the state-of-the-art neural text generation model, GPT-2 (Radford et al., 2018) [1] and subset of human written articles from WebText (collection of online articles). We compare our discourse coherence based approaches with competitive baseline

[1] https://github.com/openai/gpt-2-output-dataset

models such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2018) (to detect itself). We present our detection results, analysis and future directions. The main contribution of our work is to explore discourse coherence based approaches to build useful features for the task of detecting machine generated text from human written text.

The report is organized as follows. Section 2 discusses the related work in building text generation models and detection models. Section 3 provides the description of the problem and the data used in this study. Section 4 presents the details of the discourse coherence based models employed in this project. Section 5 provides the settings of the baseline models, coherence models and our experimental results. We analyze our results further in Section 6 and jot down the lessons learned in this project. Section 7 concludes the report and lists interesting future research directions towards building accurate detection models.

2 Related Work

Our project deals with neural language models (NLMs), decoding techniques to generate text from NLMs and detectability of generative models. In this section, we will cover the key research works done in each of these sub fields.

2.1 Neural language models

Generative Pre-Training 2 (GPT-2) model (Radford et al., 2018) is the state-of-the-art NLM typically used to generate coherent paragraphs of text. Apart from coherency, the generated text exhibits other desirable properties such as grammaticality, consistency and usage of world knowledge. GPT-2 model can be used as a code auto-completion tool (TabNine), poetry generation tool (Solaiman et al., 2019), fake product reviews creator (Adelani et al., 2020) and so on. The main drawback with GPT-2 model as a text generator is that it does not provide knobs to control the aspects of a text (e.g. topic, sentiment). The authors of Zellers et al. (2019) present a controllable text generation model, called GROVER, that can generate a news article given the metadata of the news article (such as headline, author and date). They show that the GROVER model can create fake news that is harder for humans to identify than human-generated fake news. Similar to the spirit of the GROVER model, the CTRL model is developed to overcome the weakness of the GPT-2 model by allowing users to easily control particular aspects of the generated text. The CTRL model is a NLM that exploits naturally occurring control codes (like URL for a news article) to condition the text (e.g. news article body). These control codes govern style, content, and task-specific behavior.

2.2 Decoding text from NLM

Given a NLM and a prefix (e.g. starting of a news article), generating the text, that is computing the optimal continuation is not tractable. We generally use approximate deterministic or stochastic decoding technique to generate continuations. The two most commonly used deterministic decoding techniques are greedy search and beam search. In greedy search, we select the highest probability token at each time step. On the other hand, beam search maintains a fixed-size set of partially decoded sequences, called hypotheses. At each time step, beam search creates new hypotheses by appending each token in the vocabulary to each existing hypothesis, scoring the resulting sequences. Thus, greedy search can be seen as a special case of beam search. In practice, these deterministic decoding techniques depend highly on underlying model probabilities and suffer from generating degenerate continuation (uninteresting text often with repetitive tokens). In stochastic decoding techniques, we sample from a model-dependent distribution at each time step. We prevent sampling from low probability tokens by limiting sampling to a subset of the vocabulary at each time step. The two most effective stochastic decoding techniques are top-k sampling (Fan et al., 2018) and top-p (or nucleus) sampling (Holtzman et al., 2020). The top-k sampler limits sampling to the k most-probable tokens. On the other hand, the nucleus sampler limits sampling to the smallest set of tokens with total mass above a threshold $p \in [0, 1]$.

2.3 Detection models

Given the importance of detecting malicious text generated from NLMs, there has been a flurry of works recently from both NLP and ML communities to build accurate detectors. GLTR (Gehrmann et al., 2019) is a tool which consists of a suite of baseline statistical methods that can detect generation artifacts from NLMs such as the GPT-2 model. The tool lets human (including non-experts) study a piece of text by visualizing per-token probability, per-token rank in the predicted distribution and entropy of the predicted distribution. These simple tests rely on the assumption that text generation models over generate from a limited subset of the true distribution of natural language. The main advantage of GLTR is that it can facilitate untrained humans to accurately detect synthetic text (from 54% to 72% in terms of accuracy). The main weakness of this work is that the generative models can get better at producing good quality text that lacks statistical anomalies, which can make GLTR to be ineffective. Ippolito et al. (2019) conducts a thorough study on how choices such as sampling method, length of the text excerpt can impact the performance of human raters as well as automatic detection methods. They find that automatic detectors perform significantly better than human raters (similar to the result of the GLTR work). Notably, text generated through longer text excerpts are easier to identify for both automated and human raters. Automatic detectors are imperfect as they are poorly calibrated statistically and have very low correlation with expert human raters. The effectiveness of the automatic detector is highly dependent on the sampling method used to generate its training set, while some choices of sampling method leads to generalization as poor as random guessing.

3 Problem Description and Data

NLMs can be used by (low-skilled) adversaries for malicious use cases like generating fake news, impersonating others in email, automating spamming/phishing and abusive social media content production (Solaiman et al., 2019). Zellers et al. (2019) shows that NLM can be used to create fake news automatically and humans find fake news written by NLM to be more trustworthy than human-written fake news. Spammers can misuse NLMs to create fake product reviews (Adelani et al., 2020) that might bypass the traditional spam filters as the text generated using NLM closely matches human distribution. Weiss (2019) shows that NLM can be used to create fake comments in a federal public comment website that cannot be distinguished from human comments. To combat the threats posed by such adversaries, we need to build accurate models that can identify text generated by NLM from human written text. Such a model can be used to moderate content in vulnerable platforms such as social media, email clients and e-commerce websites.

Our task can be defined as follows. Given a set of human written articles and articles generated by a state-of-the-art NLM (like GPT-2), the goal is to build a binary classifier that can detect human written article from machine generated article. We use the publicly available corpora provided by OpenAI [2]. The corpora contains 250K articles from WebText (collection of human written articles). The corpora also contains text generated by GPT-2 model, specifically 250K articles each from three different stochastic sampling techniques such as pure sampling (no truncation), top-k sampling and nucleus sampling. Thus, the corpora allow us to frame three binary classification tasks:

- Detecting WebText articles from articles generated using pure sampling (PURE)
- Detecting WebText articles from articles generated using top-k sampling (TOPK)
- Detecting WebText articles from articles generated using nucleus sampling (NUCLEUS)

OpenAI also separately provide 5K examples each for creating validation and test set.

Given the large size of the dataset, we use only sample from the original dataset for our experiments, specifically 15K, 1.5K and 1.5K for creating training, validation and test set respectively for each class.

4 Proposed Approach

Gehrmann et al. (2019) reports that while humans would vary expressions in real texts, GPT-2 model rarely generate synonyms or referring expressions for entities, which does not follow the theory of centering in discourse analysis (Grosz et al., 1995). In our manual analysis of GPT-2 generation, we observe that the salient entity in sentences to be switching back and forth between multiple entities, which might indicate that the text is not locally coherent. Existing detection models treat the text to be classified as a flat sequence of tokens, ignoring the sentence level structural information present in the text. We hypothesize in the project that the sentence level cues can help us model how NLMs handle coherence. This in turn can act as useful features in the detection task. To this end, we propose to model the detection task using models from discourse coherence (Lai and Tetreault, 2018), which

[2] `https://github.com/openai/gpt-2-output-dataset`

can capture how sentences are connected in a document as well as how a document is organized. We assume the human written text is coherent in general while the machine generated text is incoherent in general. In other words, we are casting the detection problem as a task of identifying if the text is coherent (human written) or incoherent (machine generated). Below we will list down the discourse coherence models we have employed in this project. These discourse coherence models range from basic models that model entity centric coherence to advanced models based on neural networks that capture richer interactions (sentence-sentence and paragraph-paragraph interactions). [3]

4.1 Entity Grid (EGRID)

The entity grid model (Barzilay and Lapata, 2008) works on the assumption that the distribution of entities (noun phrases) in locally coherent texts exhibits certain regularities. [4] The entity grid is a two-dimensional matrix that tracks the distribution of entity mentions across sentences. Each row of the grid correspond to sentences and each column correspond to entities in the document. Each cell in the grid contains the syntactic role of an entity in a sentence. From this grid, we can extract the patterns of local entity transitions by considering continuous subsequences of each column. Each transition will have a certain probability in a given grid. We represent the document as a distribution over transition types. We treat the distribution as a feature vector and train a random forest classifier to perform the coherence prediction.

4.2 Entity Graph (EGRAPH)

The main disadvantages of EGRID model are data sparsity, domain dependence and computational complexity, especially in terms of feature space issues when building the model. Additionally, EGRID model is restricted to capture transition between adjacent sentences only. The entity graph model (Guinaudeau and Strube, 2013) overcomes the above-mentioned challenges by representing the information in entity grid in a graph. [5] In the graph, each node is a sentence in the document and edge exists between two nodes if the two sentences share at least one entity. Edge weights are computed based on the number of entities shared, the syntactic roles of the entities, or the distance between sentences. The coherence score of a document is based on the average outdegree of its graph (a centrality measure). This measure is computationally easier to calculate and captures the degree to which the sentences in a document are connected, that is, higher the outdegree, higher the coherence is. We use logistic regression by treating this measure as a feature to perform classification.

4.3 Sentence Averaging (SENTAVG)

We employ neural network based coherence models that is trained end-to-end on this classification task. In general, these models can capture richer interactions between sentences and paragraphs in a document. We experiment with the sentence averaging model proposed in Lai and Tetreault (2018) that can help us in investigating the importance of sentence order in the classification task by ignoring the sentence order. [6] Essentially, the model consists of a single LSTM that emits a sentence embedding (the last output embedding) by utilizing a sequence of GloVe word embeddings to represent the words in the sentence. The document embedding is obtained by averaging over all sentence embeddings in that document. The document embedding is then passed through a linear layer followed by a softmax layer to perform end-to-end classification. We choose this model as it helps us to measure the importance of sentence structure in detecting machine generated text from human written text.

4.4 Paragraph Sequence (PARSEQ)

We believe the role of paragraphs is crucial in our detection task as we find text generated by GPT-2 model tends to have *more* number of *longer* paragraphs than a human typically writes (as discussed in Section 6). To leverage the paragraph-paragraph interactions (in addition to sentence-sentence interactions) in a document, we employ the hierarchical document model, namely paragraph sequence model proposed in Lai and Tetreault (2018). [7] This model contains three LSTMs. The first LSTM takes a sequence of GloVe word emebddings to

[3]The authors of Lai and Tetreault (2018) provided us the implementation for all the models over email.

[4]We use dependency and constituency parser from Stanford CoreNLP tool (Manning et al., 2014) to get the syntactic roles and noun constituents respectively.

[5]We use the same tokenization, parsing tools used in EGRID model.

[6]We use `nltk` toolkit for sentence segmentation and tokenization.

[7]We use line breaks to identify paragraphs in the document and use *nltk* toolkit for sentence segmentation and tokenization.

produce a sentence vector. The second LSTM takes a sequence of sentence embeddings to produce a paragraph embedding for each paragraph in the document. The final LSTM takes a sequence of paragraph embeddings to produce a document embedding. Similar to SENTAVG model, we pass the document embedding through a linear layer followed by a softmax layer to perform end-to-end classification.

5 Experimental Setup and Results

In this section, we will lay out the details of our baseline models and coherence models (discussed in the previous section). We will also present our detection results. We make the code used in our experiments publicly available for reproducibility. [8]

5.1 Baseline models

We use the following baseline models to compare with existing work.

5.1.1 NGRAM model

NGRAM is a simple baseline model provided by OpenAI that represents a document using tf-idf representation (with unigrams and bigrams) and uses a logistic regression model to perform classification. We use the original code provided by OpenAI to implement this baseline model. [9] The document is tokenized using a white space tokenizer.

5.1.2 BERT model

BERT model (Devlin et al., 2019) is a strong baseline model that is pre-trained for bidirectional representations from large amounts of unlabeled text. The representation learned by BERT model has been shown to be useful for a diverse set of downstream NLP tasks including question answering, text classification, paraphrase detection. We use the implementation of BERT model provided by Hugging Face to build our detection classifier [10]. We make use of the `bert-large-uncased` variant of BERT model that consists of 24 layers, each having a hidden size of 1024 and 16 attention heads ($340M$ parameters) pre-trained on lower-cased English text. We finetune the pre-trained model on our task for 3 epochs, with learning rate of 2e-5 and batch size of 32.

[8] https://tinyurl.com/y9acnnzu

[9] https://github.com/openai/gpt-2-output-dataset/blob/master/baseline.py

[10] https://github.com/huggingface/transformers

5.1.3 GPT-2 model

We use the GPT-2 model (Radford et al., 2018) as a baseline model to detect generation produced by itself. Zellers et al. (2019) shows that detectors based on NLMs are better at detecting their own generation as they know the tail of the distribution well. Similar to our BERT model setup, we use the implementation of GPT-2 model provided by Hugging Face to build our detection classifier. We make use of the `gpt2-medium` variant of GPT-2 model that consists of 24 layers, each having a hidden size of 1024 and 16 attention heads ($345M$ parameters), which is of similar capacity to BERT model except that the representations learned by GPT-2 model are unidirectional (left to right) and pre-trained on a collection of online articles, WebText. Note that human written text for our detection task is taken from the test set of WebText.

5.2 Coherence models

We will now discuss the configuration for our coherence models. For EGRID and EGRAPH models, we assign each noun form to one of the three syntactic properties - subject (S), object (O) or miscellaneous (X) and noun forms used in different roles in a single sentence will be resolved in the following order of preference: S (most preferred), O and X. For EGRAPH model, we choose a weighted syntax-sensitive graph without discounting for the distance between sentences and the role weights for S, O and X are set to 3, 2 and 1 respectively. For neural models such as SENTAVG and PARSEQ, we use the validation set to tune the model hyperparameters such as hidden size of a LSTM cell (100, 300), word embedding size (100, 300) and dropout rate (0.1, 0.5). We use a batch size of 32 and train each model for 10 epochs.

5.3 Results

Table 1 furnishes the detection results of all our models in this study. Unsurprisingly, the pre-trained language models such as BERT, GPT-2 performs very well as their internal representations (except the linear classification layer) are pre-trained on large amounts of data. GPT-2 excels in PURE and TOPK tasks while BERT outperforms GPT-2 by a small margin in NUCLEUS task. This result is consistent with that of Zellers et al. (2019), where they show that GROVER model is best at detecting itself than BERT discriminator. We believe using the largest model of GPT-2 (`gpt2-xl` vari-

model	PURE	TOPK	NUCLEUS
Baseline models			
NGRAM	66.2	86.5	66.86
BERT (Devlin et al., 2019)	75.27	90.77	**70.33**
GPT-2 (Radford et al., 2018)	**76.2**	**92.8**	69.2
Entity based coherence models**			
EGRID (Barzilay and Lapata, 2008)	**68**	61	57
EGRID (with unigrams, bigrams)	64	70	56
EGRAPH (Guinaudeau and Strube, 2013)	57.04	63.64	52.6
EGRAPH (with unigrams, bigrams)	67.09	**84.8**	**62.38**
Neural based coherence models			
SENTAVG (Lai and Tetreault, 2018)	51	37.03	40.83
PARSEQ (Lai and Tetreault, 2018)	**60.3**	**76.8**	**59**

Table 1: Accuracy percentage of baseline, entity based and neural based coherence models. **The performance of entity based models are not comparable with the rest of the models as nearly 50% of the documents resulted in parsing errors and are excluded from the training, validation and test set.

ant) might have resulted in strong performance of GPT-2 model over BERT model as the generation indeed comes from the largest model of GPT-2. In practice, we might not have access to the parameters of the original NLM whose generated text we wish to detect. Additionally, classifiers based on BERT and GPT-2 models are not interpretable and hence might not be able to guide humans in detecting fake text reliably.

We face parsing errors with 50% of the documents in the dataset, which prevents us from directly comparing the numbers obtained for our entity based coherence models with other models considered in this study. Nevertheless, entity based models throws light on the coherence characteristics of the text generated using different sampling techniques. Strikingly, text generated using nucleus sampling are much harder to detect by the coherence based models in general. As seen in the next section, this result is due to the fact that the nucleus generations (compared to generations obtained through other sampling methods) are as coherent as human written text, which makes the coherence based detector easily confuse a machine generated text for human written text. EGRAPH model with n-gram features outperforms similar variant of EGRID model, mainly due to the richness of sentence-sentence interactions captured by the former. SENTAVG model performs the worst among other models, which clearly indicates that sentence structure plays a vital role in our detection task. However, neural based coherence models are not effective for this detection task, as these models are outperformed by NGRAM model by a large margin.

	WebText	PURE	TOPK	NUCLEUS
#para	12.11	14.69	13.05	15.43
#sentlen	19.61	20.41	21.67	18.87
#paralen	18.14	19.67	22.1	18.39

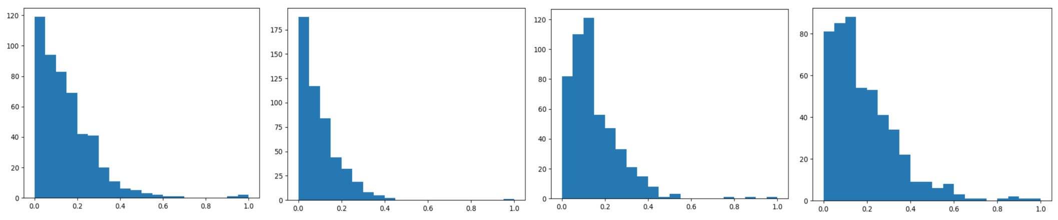
Table 2: Discourse statistics such as average number of paragraphs in a document ('#para'), average length of the sentence in a document ('#sentlen') and average length of the paragraph in a document ('#paralen') for different datasets in our study.

6 Analysis and Discussion

In this section, we provide extended analysis of the detection results along with the lessons we learned in this project.

6.1 EGRAPH Outdegree Scores

One of the key assumption of our approach of employing coherence based models for detecting machine generated text from human written text is that machine generated text is highly incoherent in general compared to human written text. To validate this assumption, we plot the histogram of the outdegree scores obtained using EGRAPH model on WebText, text generated using pure sampling, top-k sampling and nucleus sampling. As seen from the figure 1, we observe that the histogram obtained for nucleus sampling and WebText are highly similar (compared to other sampling), thereby highlighting that efficient sampler can make NLM generate text that is as coherent as human written text. This result is consistent with the results presented in table 1 that generations using nucleus sampling is harder to detect by coherence based models.



(a) WebText (b) PURE Sampling (c) TOPK Sampling (d) NUCLEUS Sampling

Figure 1: Histogram of average outdegree score obtained through EGRAPH model for different datasets.

	PURE	TOPK	NUCLEUS
uni+bi	62.5	81.4	**64.03**
#para	50.63	52.33	53.33
#sentlen	52.57	58.3	50.37
#paralen	52.47	50.27	51.2
uni+bi+#para	62.23	81.23	63.67
uni+bi+#sentlen	61.93	**81.67**	63.93
uni+bi+#paralen	62	81	63.83
all	**63.3**	80.73	63.73

Table 3: Accuracy percentage of detection models with features based on unigrams (uni), bigrams (bi), average number of paragraphs in a document ('#para'), average length of the sentence in a document ('#sentlen') and average length of the paragraph in a document ('#paralen').

6.2 Discourse Statistics

Another motivation for us to make the detector model coherency interactions such as sentence-sentence and paragraph-paragraph interactions is based on our observation through manual analysis that text generated by NLM tends to prefer more number of paragraphs and longer paragraphs in general than typical human writings. This observation is quantified in table 2 where we present three statistics for a dataset such as average number of paragraphs in a document ('#para'), average length of the sentence in a document ('#sentlen') and average length of the paragraph in a document ('#paralen'). Inspired by this striking phenomenon, we use these statistics as features to build detectors. As seen from table 3, we find that detectors based on these statistics in isolation performs at chance level. We augment the feature space of the detector with unigram and bigram features. In comparison to NGRAM baseline model, the detector with all the features performs well for PURE task while the detector with n-gram and '#sentlen' feature performs well for TOPK task.

6.3 Lessons Learned

We will now list the lessons we learned through this project.

- Utilizing coherency based features for a detector is questionable, especially if the sampling technique used by the generator is efficient (e.g., nucleus sampling) as those generated texts are highly likely to be coherent and can confuse the detector.
- NGRAM models exhibit strong detection performance than the neural based coherence models despite its simplicity.
- Detector based on the parameters of the model to be detected outperforms strong discriminators such as BERT model.
- EGRAPH model is better than EGRID model, only when the feature space also contains unigrams and bigrams. [11]
- Text generated by NLM tends to prefer more number of paragraphs and longer paragraphs in general than typical human writings.

7 Conclusion and Future Directions

In this report, we presented our experiences in employing coherence based models to detect machine generated text from human written text. To the best of our knowledge, our work is the first to explore discourse coherence based approaches for this task. In future, we plan to experiment with advanced coherence models such as entity grid with convolutions (Tien Nguyen and Joty, 2017), lexical coherence graph (Mesgar and Strube, 2016), discourse parsers (e.g. RST parser) to extract useful features for this task. We also plan to focus on

[11]This might also be due to the difference in the ML model as EGRID model uses random forest classifier while EGRAPH model uses logistic regression based classifier.

building an interpretable detector, which is also accurate and can help in aiding humans in detecting the generation from NLM.

References

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. 2020. Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human- and Machine-Based Detection. In *Advanced Information Networking and Applications - Proceedings of the 34th International Conference on Advanced Information Networking and Applications, AINA-2020*, volume 1151 of *Advances in Intelligent Systems and Computing*, pages 1341–1354.

AIW. GPT-2: It learned on the internet. https://aiweirdness.com/post/182824715257/gpt-2-it-learned-on-the-internet. Accessed on 04/22/2020.

Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics*, 34(1):1–34.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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 4171–4186.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical Detection and Visualization of Generated Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Camille Guinaudeau and Michael Strube. 2013. Graph-based Local Coherence Modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 93–103, Sofia, Bulgaria. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2019. Human and Automatic Detection of Generated Text. *CoRR*, abs/1911.00650.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *CoRR*, abs/1909.05858.

Alice Lai and Joel R. Tetreault. 2018. Discourse Coherence in the Wild: A Dataset, Evaluation and Methods. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, Melbourne, Australia, July 12-14, 2018*, pages 214–223. Association for Computational Linguistics.

Guanxiong Liu, Tzu-Ming Harry Hsu, Matthew B. A. McDermott, Willie Boag, Wei-Hung Weng, Peter Szolovits, and Marzyeh Ghassemi. 2019. Clinically Accurate Chest X-Ray Report Generation. In *Proceedings of the Machine Learning for Healthcare Conference, MLHC 2019, 9-10 August 2019, Ann Arbor, Michigan, USA*, volume 106 of *Proceedings of Machine Learning Research*, pages 249–269. PMLR.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mohsen Mesgar and Michael Strube. 2016. Lexical coherence graph modeling using word embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1414–1423, San Diego, California. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford. OpenAI. Better language models and their implications. hhttps://openai.com/blog/better-language-models/. Accessed on 04/22/2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release Strategies and the Social Impacts of Language Models. *CoRR*, abs/1908.09203.

TabNine. TabNine. Autocompletion with deep learning. `https://tabnine.com/blog/deep`. Accessed on 04/22/2020.

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1320–1330, Vancouver, Canada. Association for Computational Linguistics.

Max Weiss. 2019. Deepfake Bot Submissions to Federal Public Comment Websites Cannot Be Distinguished from Human Submissions. In *Technology Science*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending Against Neural Fake News. In *Advances in Neural Information Processing Systems 32*, pages 9054–9065.