A Survey on Neural Text Generation and Degeneration

Elham Madjidi *elham.madjidi@okstate.edu*
Department of Computer Science
Oklahoma State University
Stillwater, OK 74078 USA

Christopher Crick *chriscrick@cs.okstate.edu*
Department of Computer Science
Oklahoma State University
Stillwater, OK 74078 USA

Abstract

The evolution of text generation has been revolutionized by the rise of transformer-based models. This has brought about significant changes in various fields, including news, social media, and scientific research. However, there is a need for a comprehensive review that covers the historical evolution, challenges, and potential solutions in this domain. To address this gap, we have conducted a thorough survey that provides a comprehensive overview of text generation. We also investigate text degeneration, providing insights and mitigation strategies. Our survey sheds light on the current landscape of neural text generation, identifies forthcoming challenges, and highlights research areas that require exploration within the academic community.

Keywords: Natural Language Processing, Text Generation, Neural Text Degeneration, Large Language Models, Decoding Technique.

1. INTRODUCTION

The production process of any speech or writing is to know what to talk about and how to present that idea to the audience, so that the information is transferred effectively. Writers and speakers also optimize their speech or writing to avoid stating the obvious, which adds to the novelty of the discourse (Grice, 1975).

The two significant factors of computer-based text generation are the content of speech or written text and how to transform the message into a natural language to make the most human-like text (McKeown, 1992). Text generation is a family of sub-tasks such as text summarization, dialogue response generation, and storytelling in the natural language processing realm. Such tasks intend to generate an output text conditioned on some input information (Xie, 2017), (McEnery, 2023). Prior to neural networks, the main techniques for text generation were either based on template or rule-based systems or well-understood probabilistic models such as n-gram (Chen & Goodman, 1999) or log-linear models (Koehn & Knowles, 2017).

Recent improvements in processing power and deep learning algorithms have made automatic text production conceivable (Kilgarrif, 2001). Deep learning models used in neural text generation are in one of three categories (Fatima, Imran, Kastrati, Daudpota, & Soomro, 2022): 1) Vector-to-sequence models, which take a fixed-sized vector as an input and output is a sequence with varied sizes, image captioning is one example. 2) Sequence-to-vector, the model takes a variable-length text (such as a sentence or paragraph) as input and predicts a fixed-size vector, sentiment analysis is an example. 3)Sequence-to-sequence, these models take variable size input and generate variable size output, like text summarization. There are different deep learning architectural frameworks to implement deep learning models, such as Recurrent Neural Networks (RNN)(Rumelhart, Hinton, & Williams, 1986), Bidirectional RNN (Schuster & Paliwal,

1997), Long Short-Term Memory (LSTM)(Hochreiter & Schmidhuber, 1997), Generative Adversarial Networks (GAN)(Yu, Zhang, Wang, & Yu, 2017), and transformer based models (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, & Polosukhin, 2017).

There are three main approaches to evaluating the generated text by any model: human-centric, automatic metrics that require no training, and machine-learned metrics (Celikyilmaz, Clark, & Gao, 2020). The human-centric method is based on general and expert human evaluation of the context and grammar production which is expensive in both time and cost. Moreover, it is prone to human error. On the other hand, untrained automatic and machine-learned metrics are objective and cheap, but the quality of an evaluation depends heavily on the application. Metric for evaluation of translation with explicit ordering (METEOR) (Banerjee & Lavie, 2005), bilingual evaluation understudy (BLEU) (Papineni, Roukos, Ward, & Zhu, 2002), recall-oriented understudy for gisting evaluation (ROUGE) (Lin, 2004),and Word Perplexity (Brown, Della Pietra, Della Pietra, Lai, & Mercer, 1992) are examples of untrained automatic evaluation method. Human Unified with Statistical Evaluation (HUSE) is an example of a machine-learned evaluation metric (Hashimoto, Zhang, & Liang, 2019).

The development of neural text generation models has brought about an increase in awareness of their drawbacks and risks. The main flaws with these models are degeneration (Holtzman, Buys, Du, Forbes, & Choi, 2019); hallucination, a condition in which the model does not know the answer but still provides a wrong or nonsensical answer (Maynez, Narayan, Bohnet, & McDonald, 2020); bias, different degrees of preferences or tendencies concerning various groups (e.g., male vs. female) (Sheng, Chang, Natarajan, & Peng, 2019); toxicity, rude, disrespectful or unreasonable text; and miscalibration, plausible-sounding but incorrect information (Guo, Pleiss, Sun, & Weinberger, 2017). In this survey we introduce a comprehensive background on text generation from early works to early neural text generation models in section 2, and in section 3 we coverthe most recent neural text generation models. This is followed by an overview of the degeneration problem and solutions to alleviate it in section 4. Section 5 concludes the survey.

2. BACKGROUND

The earliest works used stored text and templates to communicate with the user. The system designer needed to know all the questions a user might ask the system and manually provide the answers to those questions. Such methods required considerable time and labor and could only handle common and foreseeable questions.

Later research focused on the tactical component, a grammar, used by a system to communicate its intent and determine the phrasing of the answer. The majority of tactical components were intended to generate English sentences from a form of semantic nets (Simmons & Slocum, 1972), write general purpose rules and heuristics using an explicit linguistic structure as the representation of the speaker's goals(McDonald, 1980), and develop a system based on the conceptual dependency system for meaning representation to understand the natural language and generate paraphrase and inference system (Riesbeck, Schank, Goldman, & Rieger III, 1975). Most of these works focused on the generation of single sentences; thus, the link of sentences to previous text was not considered. Thus, strategic components were developed to provide solutions to such problems. The works were to provide the knowledge needed to generate texts related to the context (Swartout, 1981), addressing the problem of determining the speech act to specify the agents involved and the propositional content of the act(Cohen, 1979), and enhancing the order of information in a given knowledge base.(Mann & Moore, 1981) Research systems designed with context were still limited by their linguistic bases' weakness. Their techniques often cannot be transferred to new knowledge domains. (Mann, 1983).

There is an important part of text generation which was often left out of consideration in early research, choosing lexical items and syntactic structures to best present the different facts. A theory of language generation based on planning makes it possible for one to account for noun phrases, which were proposed to address this problem (Appelt, 1985). One approach to designing a system that is able to adapt its behavior to different kinds of users with whom it

interacts, incorporates the user's domain knowledge into natural language generation (Paris, 2015).

Statistical language models (LM) are used in many natural language applications as a crucial component for improving a system's performance (Rosenfeld, 2000). A statistical language model represents a probability distribution over different granularity of text, such as words, sentences, and a whole document. N-grams were the conventional language modeling approach for its simplicity and good performance, although it has a problem of data spareness which can be addressed to some degrees with smoothing techniques (Chen & Goodman, 1999). However, the n-grams paradigm takes no advantage of the fact that modeling language means modeling deep structure, intention, and thought behind the words, not just modeling a sequence of arbitrary linguistic units.

The new millennium brought a powerful combination of ample online textual data, more inexpensive computing power, and a unique idea for building statistical phrase-based machine translation systems. (Koehn, Och, & Marcu, 2003) factorized the translation probabilities of matching phrases in the source and target sentences, which found its way to other text generation applications (Bannard & Callison-Burch, 2005) (Fader, Zettlemoyer, & Etzioni, 2013). These rule-based and statistical models have the strength of interpretability and predictability but lack flexibility and expressivity. Despite their inadequacy, there are still many tasks where these models are the only reasonable option.

Neural networks and their variations have demonstrated promising performance in various applications over the past ten years. A neural probabilistic LM was first proposed to exploit the advantages of neural networks for text generation tasks in (Bengio, Ducharme, & Vincent, 2000). This model, a feedforward neural network with a linear projection layer and a non-linear hidden layer, can be seen as an extension of the n-gram paradigm that takes advantage of a neural networks' ability to generalize, and they represent sequences of varying lengths into low-dimensional space.

The SENNA system was developed to add a convolutional architecture to the neural LM that shares representations across language modeling tasks, although it does incorporate a language model it is not a text generation system (Collobert & Weston, 2008). The neural net language model was also improved by adding recurrence to the hidden layers, which allowed it to beat smoothed n-gram models in perplexity (Mikolov, Deoras, Kombrink, Burget, & Cernocky, 2011).

Adopting improved variants of RNN, such as long short-term memory (LSTM), to generate complex sequences with long-range structure by predicting one data point at a time (Graves, 2013), and gated recurrent networks (GRU) (Chung, Gulcehre, Cho, & Bengio, 2014) to capture long-term dependencies in the text, had acceptable results. However, due to exposure bias, at training time each prediction is conditioned on the previously observed words from the ground truth and at the testing time the model will be fed with its own predictions, leading to quickly accumulating errors during inference. The result is not satisfactory (Bengio, Vinyals, Jaitly, & Shazeer, 2015a). Later neural text generation models were developed with reinforcement learning (RL) (Ranzato, Chopra, Auli, & Zaremba, 2015), generative adversarial nets (GANs) frameworks, and and end-to-end reparameterization techniques (Kusner & Hern´andez-Lobato, 2016).

In this paper, we thoroughly investigate the most recent models in neural text generation, their obstacles, and different solutions to overcome those challenges. In particular we talk about the problem of neural text degeneration and different types of strategies to address this issue.

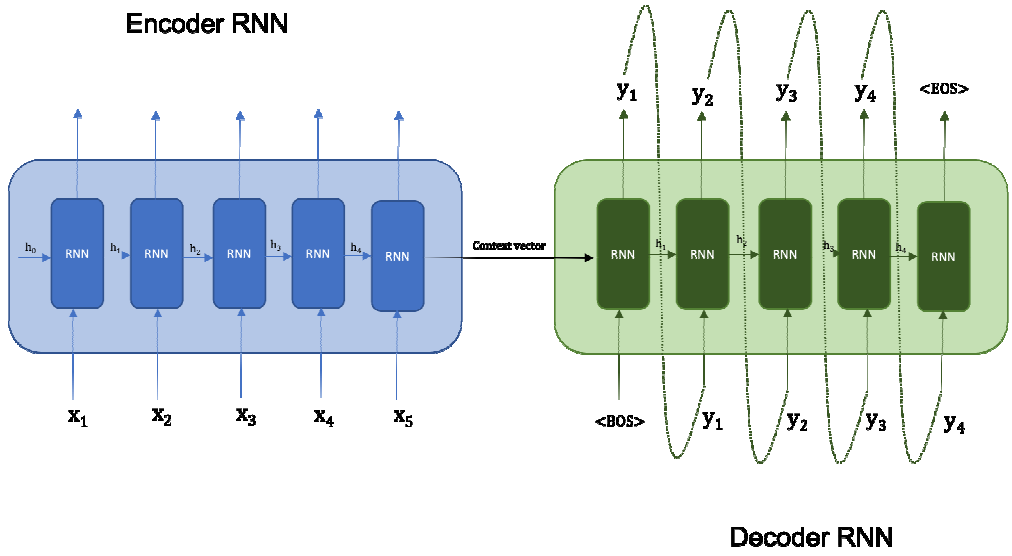


FIGURE 1: Encoder-Decoder architecture. The model reads the input sequence $X = (x_1, x_2,...,x_t)$ and produces output sequence $Y = (y_1,y_2,...,y_t)$. The model stops prediction after reaching an end of sentence (EOS) symbol.

3. NEURAL TEXT GENERATION

The abundance of large textual corpora and high-performance machines resulted in a shift toward learning representations of this large data of typically human-written texts using deep neural network models. Regarding deep learning, particularly, representation learning is the outcome of the function that a model learns, where the learning is recorded in the model's parameters as the function transforms input to output during training.

Traditional deep neural networks typically take the input textual data and encode it into a fixed-length vector using techniques such as Bag Of Words (BOW) or word2vec, none of which preserve the order of words. In most natural language processing tasks, the order of words has essential information to extract; thus, to solve this problem, RNNs were introduced.

The RNNs were the first to start and continued with LSTM, and GRUs for learning language representations and later sequence to sequence learning (Sutskever, Vinyals, & Le, 2014), which is the precursor to the encoder-decoder architecture (FIGURE 1). Encoder-decoder architecture consists of two RNNs; one encodes a sequence of symbols into an encoder/context vector, a fixed length vector representation, and the other decodes the input representation into another sequence of symbols one element at a time (Cho, Merrienboer, Gulcehre, Bougares, Schwenk, & Bengio, 2014). During the training phase, the context vector is fed to the decoder in addition to the true output from the previous time step to produce the predicted output at the current time step. The loss function, cross-entropy loss, is calculated on the predicted output from each time step and the errors are backpropagated to update model parameters. ELMo (Embeddings from Language Models), a pre-trained language model, exemplifies the usage of a bi-directional LSTM architecture to capture contextual information in word representations (Peters, Neumann, Iyyer, Gardner, Clark, Lee, & Zettlemoyer, 2018).

$$loss_{likelihood} = -\sum_{t=1}^{steps} \log p(y_t|X, y_{1,..,t-1}) \tag{1}$$

The encoder-decoder model's first application was machine translation, but shortly showed improvement in the performance of other text generation applications. The shortcoming of encoder-decoder architecture to capture dependencies in long sequences inspired the

introduction of attention (Bahdanau, Cho, & Bengio, 2015) and pointer networks (Vinyals, Fortunato, & Jaitly, 2015).

The Transformer architecture employs both an encoder and a decoder, both of which utilize the self-attention mechanism, also known as intra-attention. This mechanism leverages attention to establish connections between different positions within a single sequence, facilitating the learning of relationships between sequence elements. This approach offers two notable advantages: firstly, it's highly efficient as it enables parallel processing of words, and it doesn't require recurrent steps for understanding word positions, thanks to the injection of positional encoding into each embedding. Secondly, it enhances contextual understanding by simultaneously capturing context from both directions. During training, Transformers aim to maximize the log-likelihood of the training data by optimizing the model through sequence cross-entropy loss.

The standard Transformer model has a fixed context window, limiting its ability to capture long-range dependencies in sequences. Transformer-XL is an extension of the original Transformer model to address this limitation by introducing a positional encoding scheme and segment-level recurrence, allowing it to extend the context window and capture information beyond the fixed size. These innovations improve the model's performance in tasks requiring a broader understanding of global context or handling lengthy documents (Dai, Yang, Yang, Carbonell, Le, & Salakhutdinov, 2019).

Bidirectional Encoder Representation from Transformers (BERT) is trained using a masked language modeling objective, which is different from the conventional language modeling objective that predicts the next word in a sequence based on its history. Instead, masked language modeling predicts a word by considering its context from both the left and right sides. This unique characteristic of BERT makes it less straightforward to use as a traditional language model to evaluate the probability of a text sequence or for sampling purposes. (Kenton & Toutanova, 2019). Researchers show that BERT can be viewed as a Markov random field language model. By adopting this formulation, they develop a straightforward algorithm for generating content from BERT using Gibbs sampling, without the need for extra parameters or additional training (Wang & Cho, 2019).

XLNet is a generalized autoregressive pre-training method that takes advantage of the greatest aspects of two successful pre-training methods, autoencoding (e.g., BERT), and autoregressive language modeling, while avoiding their drawbacks (Yang, Dai, Yang, Carbonell, Salakhutdinov, & Le, 2019). XLNet outperforms BERT on twenty tasks by a large margin.

Bidirectional and Auto-Regressive Transformers (BART) is a versatile denoising autoencoder, implemented using a sequence-to-sequence model, which can be effectively utilized for a diverse set of end tasks. BART's training process involves two main steps: (1) introducing noise to the text using a flexible noising function, and (2) training the model to reconstruct the original text from the noisy input (Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Stoyanov, & Zettlemoyer, 2020). mBART (multilingual BART) is an extension of the BART model that is specifically designed for multilingual text generation tasks. (Liu, Gu, Goyal, Li, Edunov, Ghazvininejad, Lewis, & Zettlemoyer, 2020). By leveraging its pre-trained knowledge of a wide range of languages, mBART can perform well even in low-resource language settings, where there might be limited training data available for certain languages.

A stack of models, a variant of the transformer decoder, is the base for a Generative Pre-Training (GPT) to generate remarkably fluent sentences, and even paragraphs, for a given topic or a prompt (Radford, Narasimhan, Salimans, Sutskever, et al., 2018). GPT-2 is a direct scale-up of GPT with 48-layer transformers. To develop contextual text representations, GPT-2 is pre-trained on substantially larger quantities of text corpora by predicting words based on context (Radford, Wu, Child, Luan, Amodei, & Sutskever, 2019). And the OpenAI GPT-3 model with 175 billion parameters, 10x more than any previous non-sparse language model, is now competitive across many tasks with custom-made models while requiring little to no dataset-specific training data

(Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, Neelakantan, Shyam, Sastry, Askell, et al., 2020).

ChatGPT is a language model with the ability to dialogue based on GPT-3.5, a GPT3 variation that underwent training using both text and code and is capable of following the user's instructions operating via reinforcement learning (Ouyang, Wu, Jiang, Almeida, Wainwright, Mishkin, Zhang, Agarwal, Slama, Ray, et al., 2022). ChatGPT can be seen as the most human-like bot so far. Still, like GPT, it has limitations such as text degeneration and hallucination.

Several other sizable language models have been developed, including the Pathways Language Model (PaLM) (Chowdhery, Narang, Devlin, Bosma, Mishra, Roberts, Barham, Chung, Sutton, Gehrmann, et al., 2022), the Large Language Model for AI Meta (LLaMA) (Touvron, Lavril, Izacard, Martinet, Lachaux, Lacroix, Rozi`ere, Goyal, Hambro, Azhar, et al., 2023a), and the Language Models for Dialogue Applications (LaMDA) (Thoppilan, De Freitas, Hall, Shazeer, Kulshreshtha, Cheng, Jin, Bos, Baker, Du, et al., 2022). These models exhibit varying parameter counts, spanning from 40 billion parameters for PaLM1 and LLaMA-1 to 70 billion parameters for LLaMA-2 (Touvron, Martin, Stone, Albert, Almahairi, Babaei, Bashlykov, Batra, Bhargava, Bhosale, et al., 2023b), with an impressive 540 billion parameters for PaLM-1. While these models demonstrate proficiency in neural text generation and excel in specific tasks, such as LaMDA's focus on dialogues, they still grapple with the challenge of text degeneration.

The increase in the number of parameters alone does not completely address the problem of text degeneration in large language models. Despite the considerable capacity of these models, they can still produce repetitive and overly safe responses, lacking the desired level of creativity and diversity.

4. NEURAL TEXT DEGENERATION

Modern sequence-to-sequence models such as transformers show mastery in attaining low perplexity on various NLP tasks. However, they operate far from flawlessly when employed as text generators. Neural text degeneration is the condition of producing output text that is bland, incoherent, or gets stuck in repetitive loops (FIGURE 2). There are many explanations and solutions for this problem; we cover the most important ones in this part.

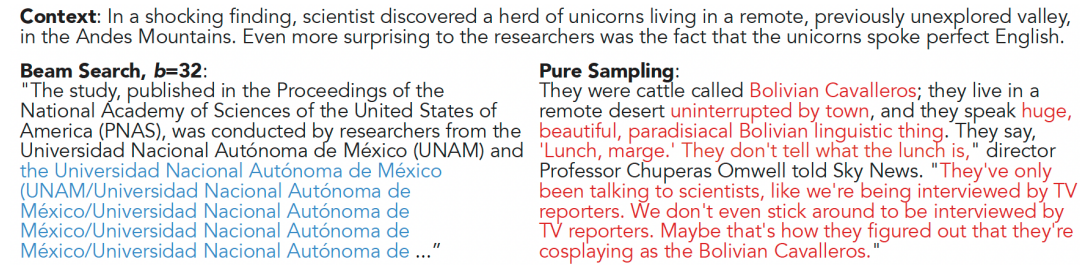


FIGURE 2: Despite having a substantial amount of human context and employing the robust GPT-2 Large language model, using Beam Search with a size of 32 results in undesirable repetitive content (indicated in blue). Conversely, opting for another decoding approach, pure sampling produces incoherent nonsensical output (highlighted in red) (Holtzman et al., 2019).

4.1 Possible Reasons for Degenerate Text

Numerous explanations for why the neural text is degenerate have been proposed, yet the true cause is still unknown. One possibility is that the language model in a transformer architecture tends to choose repeated words. A feature of human likelihood maximization (LM) is that the use of obvious statements is minimized, since humans try to omit the words they know their audience should already know from context (Holtzman et al., 2019). Another possible reason is when the model is trained on fixed textual corpora with limited words, it cannot be expected to know the whole language with many different words it never encountered in training (Choi, 2018). The next

possible reason is related to exposure bias, during training language models see diverse ground-truth target sequences, but during inference they rely on their own generated output, leading to potential biases and repetition (Bengio, Vinyals, Jaitly, & Shazeer, 2015b). Some research also takes the use of the likelihood objectives as the main factor for text degeneration (Welleck, Kulikov, Roller, Dinan, Cho, & Weston, 2019a).

4.2 Solutions to Alleviate Degenerate Text

The solutions in this part are divided into two categories. The first category blames the decoding techniques for this problem, and the second category goes to the training part of the model and replaces it with a new mechanism to address the issue. In this section, we first go through all the approaches in the decoding mechanism, and then we elaborate on the change in the training process.

4.2.1 Different Decoding Strategies

The decoding block in the Encoder-Decoder architecture takes the input sequence X representation and tries to generate the target $\hat{Y}$ that maximizes the similarity between the intended output and predicted sequence. The $\hat{Y}$ is a vector with probabilities for each generated word. The model selects the next word according to the generated probabilities.

Greedy Search is to choose the data with the highest index for the output distribution at each time step, then feed this as the input for next time step. This approach works surprisingly well in some applications, plus it is easy to understand and fast to compute.

$$\hat{y} = model.predict\ (input) \tag{2}$$

$$\hat{y} = argmax\ (\hat{y}) \tag{3}$$

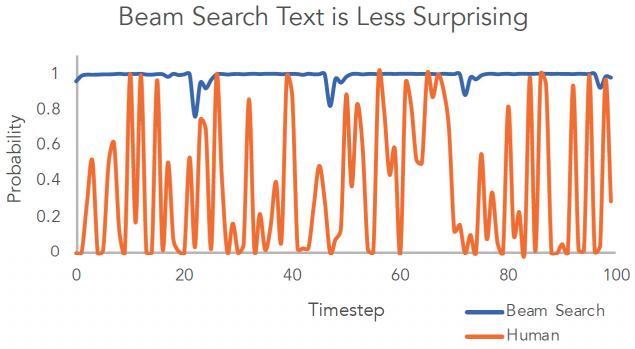
Beam search enhances greedy search in two ways. Unlike greedy search, which only considers the best word at each step, beam search widens its search to evaluate the best *k* words 1. Additionally, it takes into consideration the words that precede and follow the current time step. In beam search, the selection of the *k* best sequences (referred to as the beam size or beam width) factors in the probability of combining all the words that have come before the current word. Smaller beam sizes tend to yield better results compared to larger ones. It's important to note that beam search may degrade in performance when dealing with larger search spaces and primarily benefits translation quality when used with narrower beams (Koehn & Knowles, 2017).

There are variants of beam search to address the length bias and performance deterioration with larger beam width in neural machine translation (Bahdanau et al., 2015; Wu, Schuster, Chen, Le, Norouzi, Macherey, Krikun, Cao, Gao, Macherey, et al., 2016; He, He, Wu, & Wang, 2016).

Algorithm 1 Pseudo-code for beam search algorithm with beam size *k*

Require: $\mathcal{H} = \{[< sos >]\}$
1: **for** t = 1, . . ., steps **do**
2: **for each** $H \in$ H **do**
3: **for each** $u \in$ V **do**
4: $p = \Pr(u \mid X, H)$
5: add $H_{new} = \{[< sos >, h_1, ..., h_{steps-1}, u]\}$ to $\mathcal{H}$
6: compute and memorize the score, $s(H_{new}) = s(H) + \log p$
7: **end for**
8: **end for**
9: if *H* reached < *eos* >, add *H* to H_{final}
10: keep the *k* best *H* in H according to the score
11: **end for**
12: return $H = argmax_{H \in H_{final}} s(H)$

Decoding methods that prioritize maximization, such as the greedy approach and beam search, consistently result in highly repetitive text, regardless of the model employed. Ideally, a proficient model should assign higher probabilities to words resembling human language. However, as illustrated in FIGURE 3, human-selected words exhibit variable probabilities and aren't consistently chosen based solely on the highest probability, in contrast to these decoding techniques.



Beam Search

...to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and...

Human

...which grant increased life span and three years warranty. The Antec HCG series consists of five models with capacities spanning from 400W to 900W. Here we should note that we have already tested the HCG-620 in a previous review and were quite satisfied With its performance. In today's review we will rigorously test the Antec HCG-520, which as its model number implies, has 520W capacity and contrary to Antec's strong beliefs in multi-rail PSUs is equipped...

FIGURE 3: Probability of words chosen by beam search and humans, given the same context (Holtzman et al., 2019).

Random sampling is a straightforward approach to choosing words stochastically, to prevent repetition caused by the model's safe play to keep generating the same word for most of the context. However, as one might expect, this approach is too random and lacks coherence.

Temperature sampling, an upgrade to random sampling, uses a sharper version of a probability distribution, increasing the probability of the most likely words and decreasing the probability of the least probable words by lowering the temperature τ (as shown in Equation 4). (Ackley, Hinton, & Sejnowski, 1985) Instead of using a simple softmax we can use temperature-induced softmax to produce probabilities of each logit $u_i$at inference time (Equation 5).

$$\tilde{p} = f_\tau(p)_i = \frac{p_i^{\frac{1}{\tau}}}{\sum_j p_j^{\frac{1}{\tau}}} \tag{4}$$

$$p(x_i|x_{1:i-1}) = \frac{exp(u_i/t)}{\sum_j exp(u_j/t)} \tag{5}$$

Top-k sampling is another approach to decrease the chance of the least probable words, which only takes the sample from the top *k* probable tokens. The produced word is more human than other methods so far, which has helped this to become a popular sampling procedure (Fan, Lewis, & Dauphin, 2018). The number of *k* top tokens is chosen at the beginning of the inference process and is fixed regardless of the changes in the following word probability distribution; this may cause choosing improbable words.

Top-p sampling aims to find the smallest possible sets of words whose sum of probability is equal to or bigger than the value *p* (Equation 6). The probability distribution is re-scaled such that the probability of the words that are not in top tokens, $V^{(p)}$, is set to 0 and the rest sum to 1 (Equation 7). In this manner, the number of words in the set might fluctuate according to the probability distribution of the next word (Holtzman et al., 2019).

$$p' = \sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p \tag{6}$$

$$P\left(x|x_{1:i-1}\right) = \begin{cases} P(x|x_{1:i-1})/p' & \text{if } x \in V^{(p} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Mirostat is a sampling algorithm that uses Zipf's law to dynamically adjust k (of topk) to control per-sequence perplexity. The idea is similar to top-p sampling: both try to truncate the unreliable tail in the probability distribution. Mirostat observes the cross entropy of generated text for every word and uses it as feedback throughout the generating process, controlling the overall cross entropy (Basu, Ramachandran, Keskar, & Varshney, 2021).

Typical sampling is based on the idea that human samples are from words with high information content and low probability (Meister, Pimentel, Wiher, & Cotterell, 2022). It means we tend to not use the words we expect the audience to know or expect to see/hear. The idea of typical sampling is to choose the words that minimize the difference between the expected information content (Shannon, 1948) (conditional entropy given context) and true information content (Equation 8).

$$\sum_{y \in V^{(\tau)}} = |H(q(.|y_{<t})) + \log q(y|y_{<t})| \tag{8}$$

The set of words $V^{(\tau)} \subseteq V$ are the words with sum mass probability constrained by a threshold τ (Equation 9).

$$\sum_{y \in V^{(\tau)}} = q(.|y_{<t}) \geq \tau \tag{9}$$

For applications such as neural machine translation, which has a peaked probability distribution and a few words have the majority of probability mass, the conditional entropy is low, and hence the typical sampling is not a good decoding mechanism. However, in applications such as storytelling or paraphrasing with more spread-out distribution, this method has the potential to choose a more human-like next word.

Contrastive decoding is a search objective inspired by the fact that many failures of language models such as repetitive, bland, and irrelevant tokens are more prevalent in smaller LMs than in larger LMs. (Li, Holtzman, Fried, Liang, Eisner, Hashimoto, Zettlemoyer, & Lewis, 2022) Contrastive decoding returns the difference between the probability distribution from a larger off-the-shelf model (which authors call it expert) and a smaller model (which authors call it amateur). For a prompt sequence of $x_{1,\ldots,n}$ as an input the decoder aims to generate the continuations $x_{n+1,\ldots,n+m}$ such that it maximizes $L_{CD}=\sum_{i=n+1}^{n+m} CD_{score}(v; x_{1,\ldots,i-1})$.

The score (Equation 10) is a token-level objective limited to $v \in V_{head}(x_{<i})$ which is the set of plausible tokens whose probabilities are more than α (hyperparameter in [0,1.0]) times the maximum probability (Equation 11).

$$CD_{score}(v; x_{<i}) = \log(p_{exp}(v; x_{<i})) - \log p_{ama}(v; x_{<i}) \tag{10}$$

$$V_{head}(x_{<i}) = \{v \in V : p_{exp}(v|x_{<i}) \geq \alpha \max_{w} p_{exp}(w|x_{<i})\} \tag{11}$$

Contrastive search is similar to contrastive decoding but only requires one language model. The idea is to generate the next token from the set of most probable candidates $V^{(k)}$ and the generated token should be sufficiently discriminatory in light of the previous context (Equation 12). (Su, Lan, Wang, Yogatama, Kong, & Collier, 2022)

$$x_i = \arg\max_{v \in V^{(k)}} \Big\{ (1-\alpha) \times p_\theta(v|x_{<i}) - \alpha(\max\{s(h_v, h_{x_j}) : 1 \leq j \leq i-1\}) \Big\} \tag{12}$$

$v \in V^{(k)}$ is a token from the set of top-k predictions from the model's probability distribution with respect to the previous context. Degenerate penalty is the cosine similarity s (.,.) between representations of candidate word h_v and all tokens in the context $h_{x<i}$.

4.2.2 Unlikelihood Training

The standard likelihood objective assigns excessive probabilities to sequences containing repeated and frequent words. Unlikelihood training proposes an objective that penalizes the tokens that contribute to this pattern (Welleck et al., 2019a).

Unlikelihood training simply tries to decrease the mode's probability of negative candidates, previous context tokens C^t, by defining the unlikelihood loss (Equation 13) in such a way that the loss decreases as the probability of such candidates decreases. There is often some correct repetition in the ground-truth text. therefore, to avoid penalizing a true repetition, the ground-truth next-token is not a candidate: $C^t = \{y_1, y_2 \ldots, y_{t-1}\} \setminus \{y_t\}$. Choosing previously generated tokens as candidates is not only simple but also fast since the tokens are already present in the training sequence.

$$loss_{unlikelihood} = -\sum_{c \in C^t} \log(1 - p(c|y_{1,..,t-1}, X)) \tag{13}$$

The token-level unlikelihood loss objective is maximum likelihood training augmented with unlikelihood loss (Equation 14).

$$loss_{UL-token} = -\sum_{c \in \mathcal{C}^t} \log(1 - p(c|y_{1,..,t-1}, X)) - \log p(y_t|y_{1,..,t-1}, X) \tag{14}$$

The concept of unlikelihood training implies that when the model receives input tokens that might lead to repeated content from the prior context, it effectively minimizes the chances of incorrect repetitions. This process doesn't disrupt the occurrence of natural repeats in the language. Additionally, it causes words that were previously more common to become less frequent as the model diminishes the probability of tokens it has encountered in the context of previous tokens.

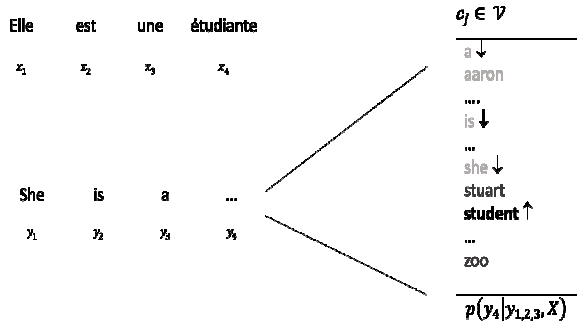


FIGURE 4: Example: Token-level unlikelihood training reduces the repeated tokens' probability.

The token-level unlikelihood training is limited to the training dataset; however, the issue of too frequent words can also happen with generated sequences. Thus, the sequence level unlikelihood training consists of decoding a sequence from the model, and then computing the unlikelihood loss using the decoded sequence instead of the ground truth (Equation 15).

$$loss_{UL-seq} = -\sum_{c \in \mathcal{C}^t} \log(1 - p(c|y'_{1,..,t-1}, X)) \tag{15}$$

A token is penalized if it is in any part of a repeating n-gram (Equation 16) or a random token in the decoded sequence.

$$C^t = \{y'_t\} \text{ if } \{y'_{t-i}, ..., y'_t, ..., y'_{t+j}\} \in y'_{<t-i} \text{ for any } (j-i) = n, i \leq n \leq j \tag{16}$$

The $loss_{UL-token}$ uses the same amount of supervision as $loss_{likelihood}$. The previous context candidates are already present in the training sequence; therefore, the model gets better results without much cost. Experiments show that $loss_{UL-seq}$ improves when measured with $loss_{UL-token}$ compared with $loss_{likelihood}$, with a slight increase in training time.

Unlikelihood training has been applied in generative dialogue models, where it lowers the probability of inconsistent dialogue. (Li, Roller, Kulikov, Welleck, Boureau, Cho, & Weston, 2020) This makes the model produce more human-like dialogue with fewer repetitive words, use more rare words from the vocabulary, and copy the context as a target. Furthermore, if the model is applied to a labeled dataset of coherent and incoherent utterances, it improves contradiction, logically or factually inaccurate or contradicting statements (Zhang, Dinan, Urbanek, Szlam, Kiela, & Weston, 2018; Welleck, Weston, Szlam, & Cho, 2019b).

4.2.3 Contrastive Training

Contrastive training is based on the idea that model degeneration is caused by the anisotropic distribution of token representation. The cosine similarity between the representation vectors for

different tokens in a sentence is very high, up to 0.95. The goal of the approach is to encourage the model to learn discriminative isotropic token representation. A contractive loss $loss_{CL}$ is introduced to the training of the model (Equation 17). (Su et al., 2022) to push away representations of different tokens.

$$loss_{CL} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \max\{0, \rho - s(h_{x_j}, h_{x_j}) + s(x_j, h_{x_j})\} \tag{17}$$

SimCTG (a simple contrastive framework for neural text generation) uses contrastive training in order to train the language model and give it an isotropic and discriminative representation space. SimCTC is companied with contrastive search to encourage diversity. SimCTG incorporates an additional term in the training formulation (Equation 18) and a modified inference time strategy during decoding. The combination induces diversity (reducing degeneration) in the generated text while maintaining relevance to the input.

When a pre-defined margin $\rho \in [-1,1]$ is 0, $loss_{SimCTG}$ degenerates to $loss_{likelihood}$. (Su et al., 2022).

$$loss_{SimCTG} = loss_{likelihood} + loss_{CL} \tag{18}$$

Approach	Algorithm	Mechanism
Decoding(deterministic)	Greedy	Generates the most probable word given the context at each time step.
	Beam search	At each level keeps k the most probable words and continue the breadth-first search until the end of the sequence.
	Contrastive decoding	Creates a set of the most probable tokens from the expert model given context. Then calculate the probability of those with the amateur model. At last, chooses the tokens with the maximized difference between expert log-prob and amateur log-prob
	Contrastive search	Create a set of most probable token from LM. Then measures the probability of these tokens and reduces it by the cosine similarity of a previously generated token.
Decoding(stochastic)	Random sampling	Randomly chooses a token or tokens from the model's generated distribution.
	Temperature	Randomly chooses a token from the shaped model's generated distribution with a temperate τ
	Top-k	Randomly chooses a token from a truncated model. Always samples from k most probable words.
	Top-p	Samples from the accumulative probability mass of 0<=p<=1.
	Mirostat	Similar to top-k sampling but the k is dynamically adjusted using Zipf's law
	Typical	Samples from a subset of words whose log probability is in the τ range of expected information content of the model.
Training	Unlikelihood	Adds $loss_{likelihood}$ to training process to penalize the model for generating undesirable tokens.
	Contrastive	Adds $loss_{CL}$ to the training process which calculates the cosine similarity between different token representations to maximize this distance.

TABLE 1: Overview of text generation approaches to alleviate degenerate text.

5. CONCLUSIONS AND FUTURE DIRECTIONS

Neural text generation is central to many NLP applications such as storytelling, summarization and translation. In recent years, research in neural text generation has resulted in remarkable architectures such as transformer-based models. Despite the reported low perplexity of such models, they suffer from degeneration when used in certain applications of NLP. In this survey, we provide an overview of text generation and the accompanying problem of degeneration. We thoroughly investigate the solutions to alleviate the problem of degeneration. This literature review aims to help researchers, students and academicians to understand the problem and see the possible solutions.

Future work can establish the reasons for the state-the-art model giving the most probability on words that are not human choices in text generation, despite the fact that these models were trained on huge amount of human-written text. One other good area for future research is the perplexity and informativeness trade-off for a generated text and why low perplexity is not always the best choice.

6. REFERENCES

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, *9*(1), 147–169.

Appelt, D. E. (1985). Planning english referring expressions. *Artificial intelligence*, *26*(1), 1–33.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Banerjee, S., & Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Bannard, C., & Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 597–604.

Basu, S., Ramachandran, G., Keskar, N., & Varshney, L. (2021). Mirostat: A neural text decoding algorithm that directly controls perplexity. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015a). Scheduled sampling for sequence prediction with recurrent neural networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 28, p. 1171–1179. Curran Associates, Inc.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015b). Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, *28*.

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, *13*.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Lai, J. C., & Mercer, R. L. (1992). An estimate of an upper bound for the entropy of english. *Computational Linguistics*, *18*(1), 31–40.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Celikyilmaz, A., Clark, E., & Gao, J. (2020). Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Chen, S. F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, *13*(4), 359–394.

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Choi, Y. (2018). The missing representation in neural language models. In *3rd Workshop on Representation Learning for NLP (RepL4NLP)*.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Cohen, P. R. (1979). *On knowing what to say: Planning speech acts.* Ph.D. thesis, ProQuest Information & Learning.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., & Salakhutdinov, R. (2019). TransformerXL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy. Association for Computational Linguistics.

Fader, A., Zettlemoyer, L., & Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1608–1618.

Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898.

Fatima, N., Imran, A. S., Kastrati, Z., Daudpota, S. M., & Soomro, A. (2022). A systematic literature review on text generation using deep neural network models. *IEEE Access*, *10*, 53490–53503.

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, *abs/1308.0850*.

Grice, H. P. (1975). Logic and conversation. In *Speech acts*, pp. 41–58. Brill.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR.

Hashimoto, T. B., Zhang, H., & Liang, P. (2019). Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1689–1701.

He, W., He, Z., Wu, H., & Wang, H. (2016). Improved neural machine translation with smt features. In *Thirtieth AAAI conference on artificial intelligence*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Kenton, J. D. M.-W. C., & Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171– 4186.

Koehn, P., & Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pp. 28–39.

Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation. Tech. rep., University of Southern California Marina Del Rey Information Sciences Inst.

Kusner, M. J., & Hern´andez-Lobato, J. M. (2016). Gans for sequences of discrete elements with the gumbel-softmax distribution.. *CoRR*, *abs/1611.04051*.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online. Association for Computational Linguistics.

Li, L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., & Lewis, M. (2022). Contrastive decoding: Open-ended text generation as optimization. *arXiv e-prints*, *30*.

Li, M., Roller, S., Kulikov, I., Welleck, S., Boureau, Y.-L., Cho, K., & Weston, J. (2020). Don't say that! making inconsistent dialogue unlikely with unlikelihood training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4715–4728.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., & Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, *8*, 726–742.

Mann, W. C. (1983). An overview of the penman text generation system. In *AAAI*, pp. 261–265.

Mann, W. C., & Moore, J. A. (1981). Computer generation of multiparagraph english text. *American Journal of Computational Linguistics*, *7*(1), 17–29.

Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919.

McDonald, D. D. (1980). *Natural language production as a process of decision-making under constraints*. Ph.D. thesis, Massachusetts Institute of Technology.

McKeown, K. (1992). *Text generation*. Cambridge University Press.

Meister, C., Pimentel, T., Wiher, G., & Cotterell, R. (2022). Typical decoding for natural language generation. *arXiv preprint arXiv:2202.00666*, *30*.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., & Cernocky`, J. (2011). Empirical eval-˘ uation and combination of advanced language modeling techniques. In *Proceedings of Interspeech*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, *32*.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

Paris, C. (2015). *User modelling in text generation*. Bloomsbury Publishing.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (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)*, pp. 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. *33*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.

Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, *28*.

Riesbeck, C. K., Schank, R. C., Goldman, N. M., & Rieger III, C. J. (1975). Inference and paraphrase by computer. *Journal of the ACM (JACM)*, *22*(3), 309–328.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here?. *Proceedings of the IEEE*, *88*(8), 1270–1278.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*(11), 2673–2681.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423.

Sheng, E., Chang, K.-W., Natarajan, P., & Peng, N. (2019). The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3407–3412, Hong Kong, China. Association for Computational Linguistics.

Simmons, R., & Slocum, J. (1972). Generating english discourse from semantic networks. *Communications of the ACM*, *15*(10), 891–905.

Su, Y., Lan, T., Wang, Y., Yogatama, D., Kong, L., & Collier, N. (2022). A contrastive framework for neural text generation. In Oh, A. H., Agarwal, A., Belgrave, D., & Cho, K. (Eds.), *Advances in Neural Information Processing Systems*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Swartout, W. R. (1981). Explaining and justifying expert consulting programs. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, p. 815–823, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. (2022). Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozi`ere, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L ., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, *28*.

Wang, A., & Cho, K. (2019). Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*.

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., & Weston, J. (2019a). Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Welleck, S., Weston, J., Szlam, A., & Cho, K. (2019b). Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3731–3741.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, *88*.

Xie, Z. (2017). Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*.

Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., & Weston, J. (2018). Personalizing dialogue agents: I have a dog, do you have pets too?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2204–2213, Melbourne, Australia. Association for Computational Linguistics.