Characterizing and Resolving Degeneracies in Neural Autoregressive Text Generation

by

Ilia Kulikov

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January, 2022

Dr. Kyunghyun Cho

Dr. Jason Weston

Acknowledgements

I am lucky to be advised by amazing people, Kyunghyun Cho and Jason Weston. Kyunghyun, I truly appreciate the guidance and knowledge I received from you. Apart from very meaningful feedback on research, you inspire me to develop myself into a decent human being. Jason, your support and discussions have changed the way I think about research problems, thanks for this. I am inspired by Jason's dedication to his research goals and how he leads his team to it.

I thank my thesis committee members He He, Andrew Wilson, and Andre Martins for their feedback. The NYU lab members supported me in many ways during this journey. I am proud to be the part of CILVR and ML^2 groups. I am glad to have met Sean Welleck and thank him for numerous chats, joint projects, and discussions about research and beyond. I am grateful for the opportunity to have worked with Maksim Eremeev, Richard Pang, Jaedeok Kim, Jason Lee, and Cinjon Resnick during my PhD. Thanks to Aahlad Puli, Aishwarya Kamath, Alex Wang, Alfredo Canziani, Angelica Chen, David Brandfonbrener, Denis Yarats, Elman Mansimov, Ethan Perez, Ilya Kostrikov, Isaac Henrion, Jaan Altosaar, Jake Zhao, Jason Phang, Katharina Kann, Katrina Evtimova, Kiante Brantley, Mark Goldstein, Martin Arjovsky, Nikita Nangia, Phu Mon Htut, Raphael Shu, Rob Fergus, Roberta Raileanu, Sam Bowman, Sreyas Mohan, Will Whitney for discussions we had in the lab. I thank Hong Tam for all his help in the office space, Hong is the best office manager I have met.

I have gained a lot of research experience during two internships at FAIR. Thanks to the entire Repartee team for their help including Stephen Roller, Alex Miller, Emily Dinan, and everyone else. I had a unique experience of a remote internship at Google. I thank Felix Stahlberg, Shankar Kumar, Orhan Firat, and Ciprian Schelba for their support and making the best possible environment to tackle the research I am interested in.

The first time I started to think about pursuing a PhD was in 2015 when I was doing my Masters degree at RWTH Aachen. I am thankful to Albert Zeyer, Ralf Schlueter, and Hermann Ney for the opportunities I have received at i6. I thank Ilya Kostrikov and Vitaly Kurin for chats and talks we have had so far since Aachen times until nowadays.

Most importantly, this journey would not be feasible without my wife Anastasia, who supported me all the time. Thanks to all my old friends for staying in touch even when I became not easily reachable on the other side of the globe. Final sentence of this paragraph is written in Russian in order to make it readable for recipients. Спасибо моим родителям за их труд и время, которое они потратили на мое воспитание.

Abstract

Autoregressive neural networks have shown great success as part of the sequence to sequence framework solving a diverse set of sequence generation tasks. These tasks include machine translation, dialogue modeling, question answering, text summarization, and sequence completion. In spite of the visible success, many challenges remain to be solved and are reported across these tasks. These challenges are usually discussed as visible deviations in the predicted sequence compared to the given reference. It is, however, not always possible to do the comparison, because interactive tasks, such as dialogue modeling, do not come together with reference sequences in the middle of the conversation at the test time. We refer to such deviations as *degeneracies* which result in degenerate sequences. In this thesis, we work on reducing widely reported degeneracies within specific tasks or in text generation in general. To do so, we often first need to formulate the degeneracy in a measurable way and hypothesize what is the major cause behind it.

We investigate the issue of oversmoothing, where the model assigns high probability to overly short sequences. We address this degeneracy from the learning aspect by proposing a novel regularization which minimizes the newly proposed oversmoothing rate directly. We show the effectiveness of the proposed method in the context of neural machine translation. Still concentrating on the learning aspect, we next address the problem of repetition in the context of sequence completion, where the generated sequences have unreasonably many repetitive substrings compared to the ones we see in the data. We propose a novel unlikelihood training procedure which allows to penalize undesired continuations, such as repetitive substrings. Unlikelihood training significantly reduces the number of repetitions and improves the naturalness of the generated continuations. One issue with the repetition degeneracy is that it can also lead to non-termination. We study if the original model is able to terminate the repetitive loop itself even if we do not enforce the maximum generated length during decoding. We connect this problem of non-termination with the consistency of the distribution induced by the chosen decoding algorithm. After proving that an incomplete decoding algorithm, such as beam search, may induce the inconsistent distribution when paired with a consistent model, we propose an alternative parametrization which guarantees the decoding-induced distribution to be consistent. After that, we switch to a more complicated scenario of conversation modeling, where the model has to generate a response in a multi-turn setting. We investigate the issue of unengaging or dull responses by highlighting the importance of the decoding algorithm. We observe a low diversity of beam search candidates compared to iterative beam search which explores a wider search subspace via efficient pruning. We find that the selection criterion is as important as the decoding strategy. Along the way, we stress the importance of careful human evaluation in the presence of annotator bias and calibrate the observed scores using Bayesian inference. While we address different kinds of degeneracy, the list we tackle is not exhaustive. For instance, neural machine translation is known to produce hallucinated translations or copy large parts of the input sentence. Furthermore, degeneracies exist past autoregressive modeling in both non-autoregressive and semi-autoregressive settings. We believe our contributions will be helpful for future research solving new problems.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Neural autoregressive text generation is the basis of many NLP tasks

We start this thesis by discussing general terms which we use repeatedly throughout this work. First, we discuss the place of *text* in sequence modeling. Second, we explain the notion of *autoregressive* generation in the context of discrete sequences. After that, we present where *neural* networks come in the process of model parametrization. Finally, we discuss similarities and differences between NLP tasks, which we consider in the context of the "sequence to sequence" [142] approach.

1.1.1 Symbolic representation of a natural language using a sequence of tokens

Widely used natural languages provide a way to write the meaning of a thought in the form of a sequence. Here we casually refer to the result of this process as a piece of text. We think of a text as a sequence of tokens or small units composed in a predefined order which depends on the language. All possible tokens of a given language form a vocabulary. The size of the vocabulary varies significantly depending on what is the smallest unit or token we consider, e.g., a character or a word.

For a moment, let us consider a sentence written in English. Such a sentence consists of words and punctuation marks. Every word can be seen as a sequence of syllables or characters. Furthermore, every character is presented digitally as a sequence of bits. In the case of bits, the vocabulary size equals two (zero and one) regardless of the language. As we go higher in the hierarchy, the vocabulary size may grow to hundreds of thousands of words.

The symbolic representation of a text allows us to convert any piece of text into a vector of indices, where every index maps to a token from the vocabulary. We discuss this process of tokenization as part of data collection in Section 2.2.1. We refer readers to the extensive study of different tokenization schemes in Mielke et al. [91] to learn more about it.

1.1.2 Autoregressive modeling

We assume a probabilistic view of sequence modeling in this thesis, because it helps to take the uncertainty into account. For instance, the same input sentence written in one language may have multiple translations which are correct. In addition, the noise is likely to be presented in the data, resulting in uncertainty as well. In this work, we consider the autoregressive decomposition of a multivariate random variable which represents a sequence of tokens $\mathbf{y}$. We assume there exists an unknown target distribution $p^*(\mathbf{y})$. Our goal then is to estimate a model $p(\mathbf{y};\theta)$ such that it closely follows the unknown distribution. Given a sequence of tokens $\mathbf{y} = (y_1, \ldots, y_T)$, autoregressive decomposition allows us to rewrite the joint distribution $p(y_1, \ldots, y_T; \theta)$ as a product of conditional distributions:

$$p(y_1, \ldots, y_T; \theta) = \prod_{t=1}^{T} p(y_t | y_{<t}; \theta),$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ is a prefix of all tokens up to time $t-1$. Here we assume the left-to-right monotonic dependency, but in general this can be any order traversing this set of tokens. Chan

et al. [18] and Welleck et al. [151] investigate different ways to learn the ordering as part of the modeling process.

The "neural" part of the title of this thesis means that we use a neural network defined with parameters θ to parametrize each conditional distribution $p(y_t|y_{<t};\theta)$. In Section 2.3 we give more details and provide references to specific types of neural networks which we use in this work.

1.1.3 Seq2seq approach applied to different NLP tasks

We consider a diverse set of NLP tasks, such as machine translation, dialogue modeling, and sequence completion. These tasks can be characterized by the properties of the input and target sequences used to estimate the resulting model. In sequence completion, the input is a prefix of some text sequence, and the target is a continuation or suffix of the same sequence. In machine translation, the target sequence represents the meaning of the input sequence translated to a different language. In open-ended dialogue modeling (chit-chat), the target sequence forms a reasonable response to the input utterance or the dialogue history including multiple utterances.

Table 1.1 shows examples of such sequences for every task we discussed above. The amount of grounding information between the target sequence and the source sequence depends on the task. For instance, the target sequence from the dialogue modeling in Table 1.1 has positive sentiment and a clarifying question. Alternative sequences which include positive sentiment with other auxiliary information can be seen as a suitable response too. In contrast, the machine translation example shows a stronger connection between source and target sentences. We expect this, because translating a sentence from one language to the other requires to keep as much information from the source sentence as possible. Nevertheless, machine translation involves ambiguity in the space of possible translations too, but it is still lower compared to other tasks.

Task	Source	Target
Machine translation	Washington Square and Greenwich Village have been hubs of cultural life in New York City since the early 19th century.	Вашингтон сквер и Гринвич виллейдж сосредо-тачивают культурную жизнь Нью-Йорка с начала 19-ого века.
Dialogue modeling	Do you want to meet near the WSP after school?	Sounds good! Are we going to any specific place there?
Sequence completion	Washington Square and Greenwich Village	have been hubs of cultural life in New York City since the early 19th century.

Table 1.1: Examples of input and target sequences for tasks we consider in this thesis.

1.2 Statistical text generation before seq2seq

While we exclusively rely on sequence to sequence modeling approach in this thesis, a lot of work has been done on other methods for generating a text in a probabilistic way. Machine translation was one of the main tasks attracting a lot of researchers since 1980s. Early systems, such as IBM-1,2, performed word-based translation following the noisy channel approach. The posterior $p(\mathbf{y}|\mathbf{x})$ can be rewritten using the Bayes rule:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}) \cdot p(\mathbf{x}|\mathbf{y})}{p(\mathbf{x})},$$

where $\mathbf{y}$ is the translated sentence and $\mathbf{x}$ is the input sentence.

In the probabilistic setting, we want to choose the translation $\mathbf{y}$ which maximizes the posterior above:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \frac{p(\mathbf{y}) \cdot p(\mathbf{x}|\mathbf{y})}{p(\mathbf{x})} = \arg\max_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}) \cdot p(\mathbf{x}|\mathbf{y}),$$

where $\mathcal{Y}$ is the set of all possible sentences in the target language, $p(\mathbf{y})$ is the language model, $p(\mathbf{x}|\mathbf{y})$ is the backward translation model. A particular advantage of this modeling design is the ability to use the language model on the target side. IBM translation models specify the way to parametrize the backward translation model $p(\mathbf{x}|\mathbf{y})$ using the idea of alignments between words in input and target sequences. We encourage readers to read Collins [25] to learn more details about these models.

Phrase-based statistical machine translation [63] is an advancement of the earlier word-based models. In contrast to the word-based alignments, they allowed many-to-many mappings between phrases written in different languages [103]. Phrase-based systems included extra features in addition to language and translation models and combined it altogether in the form of a log-linear model:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{i=1}^{N} w_i \cdot \log f_i(\mathbf{y}, \mathbf{x}),$$

where w_i is a weight of feature i out of N feature models. Estimation of weights w_i was an active research question itself [102]. Instead of feature engineering and composing many different models together, sequence to sequence approach with autoregressive decomposition allows us to compute the posterior $p(\mathbf{y}|\mathbf{x})$ directly.

1.3 Degeneracy in text generation

As the title of this thesis says, our main research effort is concentrated on characterizing and resolving degeneracies. We thus must explain what degeneracy means in the context of text generation. We say that the generated sequence is degenerate if its content or form deviates from the one we expect based on the available observations. We do not attempt to categorize every visible deviation ever existed in text generation. Instead, we study widely reported issues which lack a clear understanding of the cause behind it. Our goal is to understand which part of the learning pipeline (Section 2.1) is the most responsible for the degeneracy to appear. Such understanding allows us to propose a solution and to check how well it overcomes the issue. Next, we highlight the degeneracies we addressed in this thesis and discuss the effectiveness of the proposed solution. This discussion is a brief overview of the following chapters of the thesis.

1.3.1 The issue of oversmoothing

In Chapter 3 we thoroughly describe the problem when the autoregressive model assigns an unreasonably high probability to an overly short sequence compared to the ground-truth one. We scrutinize the subset of sequences which are (1) shorter than the ground-truth sequence and (2) form prefix sequences of it. We study this issue in the context of machine translation and confirm a high degree of oversmoothing across a diverse set of task configurations. While this problem was widely reported, it was not addressed directly due to the lack of a concrete measure to work with. In earlier work [97, 110, 134], this problem was tackled from different aspects including tokenization, model parametrization, and decoding.

We define the oversmoothing rate and suggest to directly minimize it during training to overcome the degeneracy directly. To do so, we construct a convex upper bound on the oversmoothing rate and augment the negative log-likelihood loss with it during fine-tuning. We thus address this degeneracy using an alternative learning objective. Experimental results show high effectiveness of the proposed approach in reducing the oversmoothing rate while the language modeling performance of the decoder remains intact. Minimizing the oversmoothing loss leads to a significant decrease of both probability and rank of $\langle\text{eos}\rangle$ token at undesired positions. Finally, the reduced oversmoothing rate correlates with improved length modeling and translation quality when decoding is performed using larger beam sizes. We refer reader to Chapter 3 to learn more details about our findings. This work was done jointly with Maksim Eremeev and Kyunghyun Cho. We released it as a preprint on Arxiv [66].

1.3.2 The issue of repetitive content in a generated sequence

Despite the importance of ⟨eos⟩ token in modeling sequence lengths, it is a common practice to train large language models using streams of text without the notion of the end of sequence. Such models, e.g., GPT-2 [116] and GPT-3 [17], are later used as the initialization to estimate a model to solve downstream tasks [125, 163]. These approaches use a variant of stochastic sampling as part of generation process, because the deterministic decoding (Section 2.5.1) is known to produce highly repetitive, degenerate sequences [52]. In Chapter 4, we address the repetition degeneracy in the context of sequence completion task (Section 2.2.2) when the model is paired with deterministic decoding, such as beam search.

We suggest to augment the training objective with an auxiliary part called unlikelihood loss. Our suggestion is based on the fact that the negative log-likelihood loss alone has no explicit signal, which prevents repetition to have a high probability under the model distribution. We train several language models and show a significant decline in the number of repetitions produced by the models trained with the proposed objective. We conduct a human evaluation to see which model is able to produce sequences which resemble more natural continuations. Results reveal that our models are able to produce more natural continuations compared to strong opponent models which use repetition blocking heuristics during the generation process. Please refer to Chapter 4 to find more details about this project. This work was done jointly with Sean Welleck, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. It is published in the proceedings of ICLR 2020 [150].

1.3.3 The problem of non-terminating generation

After substantial improvements in the problem of repetitive sequences, we seek for a deeper understanding of the nature behind such repetitive loops. In particular, is it theoretically possible for the model paired with a decoding algorithm to never terminate the generation process?

In Chapter 5, we study the discrepancy between the model distribution over sequences and the one induced by the decoding strategy coupled with that model. To do so, we extend the notion of consistency [19] of a recurrent language model that includes the context distribution. We prove that widely used models, such as LSTMs and transformers, induce consistent distributions. Furthermore, we show that incomplete decoding algorithms, such as beam search, may induce inconsistent distributions when paired with a consistent recurrent language model. We address this degeneracy using an alternative parametrization. It ensures that the probability of $\langle \text{eos} \rangle$ token increases monotonically between generation steps. We find that the proposed method significantly reduces the empirical proxy of the number of non-terminated sequences. However, the proposed approach must be used with caution, because it slightly reduces the language modeling performance in terms of perplexity with both smaller and larger models. This work was done jointly with Sean Welleck, Richard Pang, Jaedeok Kim, and Kyunghyun Cho. It is published in the proceedings of EMNLP 2020 [148].

1.3.4 Low diversity problem in multi-turn dialogue modeling

We investigate the issue of a dull response in the open-ended dialogue modeling task or the task of predicting the next utterance (Section 2.2.4). One of the main aspect of a good conversational agent is how engaging its responses are. The degeneracy we tackle here is named as "I don't know" problem in some literature [75, 77]. We connect this issue to the deficiency of the search algorithm, such as beam search, as it is unable to uncover more engaging responses in the search space due to its approximation.

We show that the widely used beam search produces a hypothesis space approximation (Section 2.5.1) with low diversity. To do so, we compare it with an alternative decoding algorithm called iterative beam search which returns candidates with higher n-gram diversity. Despite the set of candidates with higher diversity, the resulting human evaluation does not show significant improvement in the engagingness of the resulting model measured during human evaluation.

We hypothesize that the reason is behind the decision rule which selects the candidate with the highest probability, and a dull response in the candidate set tends to have a higher probability. In addition to the importance of the search strategy and the final decision rule, we show the evidence of annotator bias which results in high variance of the human evaluation scores. We propose a Bayesian calibration which assumes the latent system's score and infer it using the observed data coming from biased human annotators. This work was done jointly with Alex Miller, Kyunghyun Cho, and Jason Weston. It is published in the proceedings of INLG 2019 [69].

2 | Background

This chapter aims to familiarize the reader with the background of autoregressive text generation and provides references which can fill the gaps to understand the following chapters. This thesis includes discussions of several natural language processing tasks, such as Machine Translation, Sequence Completion, and Dialogue Modeling. Therefore, the background chapter includes examples which apply to either each of those or to some specific task in that scope.

2.1 Learning pipeline of autoregressive text generation

We start by describing a generic pipeline applicable to the NLP tasks we consider in this thesis. We first explain why do we require such a pipeline and what is the common high-level goal across these tasks? We need to obtain a generator returning an output sequence given the input sequence. The way we choose the model architecture and parametrization, the choice of learning parameters, and decoding settings form what we call the learning pipeline. In other words, by the pipeline we mean a sequence of stages where the output of one stage can be used as is in the following stage. For ease of describing the background, we assume that there are no feedback loops in the pipelines we consider, but they exist in the real world systems.

Data collection is the first stage of the pipeline, resulting in a data distribution or dataset. The dataset can be seen as the empirical distribution of the unknown target distribution. The dataset comprises pairs of labeled observations which are used to train the model. The goal of training is to estimate a model as close as possible to the unknown distribution approximated via the dataset used during training. After training, the model is used to make a prediction given the unseen input with an expectation of high quality prediction regarding a task metric. Evaluation stage helps to answer multiple questions. Given a single model, it helps to judge the approximate model performance we may expect after deploying the system. In addition, it helps to perform model selection when we have several models trained differently and wish to choose the best one. We can see the pipeline described above in tasks beyond text generation, such as general multiclass classification. Autoregressive text generation, however, introduces distinct properties to every stage of the pipeline, which we discuss in more detail below.

2.2 Data collection

We use the term data collection to identify the stage when relevant pairs of input and output samples are collected. The amount of annotation effort differs depending on the task, where machine translation requires hiring professional translators, but sequence completion does not need that. Despite of these differences among tasks, in the end we observe structured sequences of natural language text where the task defines both structure and constraints.

2.2.1 Tokenization: representing natural text mathematically

The essential first step of data preprocessing is tokenization. Given sequence of bytes $\mathbf{b}$, tokenization scheme or tokenizer T is a mapping $\mathbf{b} \rightarrow \mathbf{s}$ where $\mathbf{s}$ is a sequence of indices mapping to discrete symbols from the finite vocabulary set V:

$$\begin{aligned}\mathbf{b} &= \text{A sentence written in English.}\\ \mathbf{s} &= (2, 5, 6, 8, 10, 1),\end{aligned}$$

where the punctuation is separated from neighboring words first, and then the tokenizer splits the sentence by space and maps every word to its corresponding index from vocabulary. Creation of vocabulary is optional and can be done as part of tokenization on newly scraped data.

One of hyper-parameters of a tokenizer is granularity, or what is the smallest piece it considers. This includes sentences, phrases, words, syllabus, subword units, characters, and even bytes. Intuitively, as smaller units occur more frequently, this results in smaller sizes of vocabulary. Choice of tokenization hyper-parameter and how it affects task performance is an active area of research across all tasks we consider in this thesis [15, 55, 113, 147, 156]. Mielke et al. [91] study the history and motivation behind commonly used tokenization practices in the context of NLP tasks. The *de facto* standard choice of the tokenization scheme is a variant of Byte-Pair Encoding (BPE) [128]. It compresses the size of the vocabulary by joining together the most frequent pairs of subword units appearing in the data. One of the benefits of such subword tokenization is the better ability to handle out of vocabulary (oov) words.

We add special tokens during tokenization to meet specific properties of a given model or to make the computation more efficient. For instance, ⟨pad⟩ token is a placeholder token used to combine variable length sequences in one minibatch of the same length, which makes training on GPUs much more efficient. This token is never predicted or used as input for the model during learning. The ⟨eos⟩ token denotes the end of sequence and plays an important role at modeling the length distribution. We discuss the role of ⟨eos⟩ in the context of oversmoothing and non-termination degeneracies in more detail in Chapters 3 and 5.

2.2.2 Sequence completion data

The sequence completion task can be seen as the interpretation of many different NLP tasks if the sequence we consider has specific properties. Lets consider dialogue modeling: if the sequence is a pair of the dialogue history and the next response, then we can see it as an example of a sequence completion task. Nevertheless, we consider sequence completion in the most generic sense without hard constraints of the text domain or structure. This allows us to investigate the issues of completing a sequence in general, which are likely to be transported to more specific tasks, such as machine translation.

Sequence completion or open-ended text generation is the only task in this thesis which does not necessarily require an extra human in the loop for annotation during data collection. It mainly requires some collection of natural text sequences originally written by humans in natural language. Because of this apparent simplicity in terms of data preprocessing, extremely large scale models have been trained on massive amounts of data crawled from web [116]. The ease of data collection process brings in the non-trivial amount of noise and other biased data which may hurt the model performance in different ways [11]. Careful data filtering and initial collection was recently shown to provide significant improvements to completions quality [73].

2.2.3 Machine translation data

As we mentioned earlier, the machine translation task usually involves collecting professional translations of a large set of sentences. Different benchmark tasks specify the domain of text where the data is being collected, such as sport, politics, or education. In addition to content-level domains, there exist differences in the language setting. For instance, shared tasks from IWSLT conference [4] study the translation of spoken language which exhibits more challenges compared to the written one. In addition, the text modality may be switched to audio, resulting in the speech translation task.

High/low resource MT One of the important descriptive axes in MT is how large or low resource a given language pair is. Large resource language pair is one which is known to have massive amounts of training data available to use, such as English and German. In contrast, low resource language pairs have data scarcity and requires more effort to achieve the translation quality of the former.

Multi-lingual setting In general, multi-lingual machine translation models require parallel sentences written in multiple languages. The idea behind multi-lingual design is to obtain a single model which is able to cover diverse set of language pairs [37]. For example, Aharoni, Johnson, and Firat [2] introduced the model covering 59 languages in 116 directions. Collecting parallel corpora for this large number of languages is hardly tractable. This opens room for research about combining sparse multi-lingual parallel corpus with larger bilingual corpora [165].

2.2.4 Dialogue modeling data

Dialogue modeling or conversation modeling has a variety of subclasses, and each subclass may differ in a way of how the data collection process looks like. In this thesis, we only consider the grounded, open-ended conversation modeling aka "chit-chat". There exists both automatically collected tasks and ones which were specifically gathered by hiring people who conducted the conversation by following a set of constraints. Automatically collected datasets include the Ubuntu dialogue corpus and Movie-DiC [9, 84]. More complicated, externally annotated datasets include Persona Chat, Wizard of Wikipedia, and Dialogue NLI [31, 149, 161]. This thesis uses the latter type where the grounding information consists of personalized attributes or properties of the agent. An alternative to open-ended conversation modeling is task-oriented conversation modeling. The main goal behind it is to conduct a conversation which results in a successful completion of the task, such as hotel booking or deal negotiation. Task-oriented tasks are out of scope for this thesis.

2.3 Model parametrization

Model parametrization defines the way we compute a score, such as the probability, for a given pair of input and output sequences. In this thesis, we consider a probabilistic setting such that we can compute the conditional probability $p(\mathbf{y}|\mathbf{x})$ exactly. We rely on locally normalized modeling, where the probability of a given token at time t, conditioned on the target sequence prefix and the input sequence, is computed using the softmax function:

$$p_\theta(y_t = v | y_{<t}, \mathbf{x}) = \frac{\exp(f(y_{<t}, \mathbf{x}; \theta)[v])}{\sum_{v' \in V} \exp(f(y_{<t}, \mathbf{x}; \theta)[v'])},$$

where f returns a vector of logits $l \in \mathbb{R}^{|V|}$. In our work, f is constructed using a neural network, such as recurrent or feed-forward one.

This allows us to compute the probability of the entire sequence using the autoregressive decomposition:

$$p_\theta(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p_\theta(y_t | y_{<t}, \mathbf{x}).$$

Alternatively to the locally normalized model, the globally normalized modeling computes the probability of a sequence at once:

$$p_\theta(\mathbf{y}, \mathbf{x}) = \frac{\exp(f(\mathbf{y}, \mathbf{x}; \theta))}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp(f(\mathbf{y}', \mathbf{x}; \theta))},$$

where f computes a single score, and the $\mathcal{Y}$ is the set of all possible target sequences for the given input sequence in the context of the given task. Computing the denominator exactly is usually intractable, thus some approximations are used. Bakhtin et al. [8] and Goyal, Dyer, and Berg-Kirkpatrick [42] study globally-normalized autoregressive sequence modeling in the context of sequence completion and machine translation.

Encoder, decoder and end-to-end learning Here we briefly present ideas behind the encoder-decoder type of the model. Sequence to sequence modeling assumes direct prediction of the output sequence given the observed input one. In this context, the encoder encodes the input sequence and produces the encoder representation. The encoder does not need to be autoregressive as it does not generate a sequence, but extracts features which are relevant for the generating process. That is, the decoder takes the encoded representation as input and learns to use it at every step of the generation.

The major benefit of this modeling approach is the ability to be trained end-to-end. That is, given the training objective of predicting the ground-truth sequence, both the decoder and the encoder parameters are updated during gradient-based optimization. The separate encoding part is not strictly necessary for all tasks. For instance, sequence completion and language modeling in general do not use any encoder, but only use the decoder to generate a sequence. In Chapters 4 and 5 we use decoder-only architectures.

Parametrization with neural networks As pointed earlier, we rely on neural networks to design f, and we later train these models such that it satisfies our training criteria. We use recurrent based LSTM neural networks [49] in Chapters 5 and 6 and transformer based neural networks [143] in Chapters 3 to 5. As it appears from the name, the recurrent neural network computes the representation of a sequence by recurrently updating its internal state step by step:

$$h_t = f(x_t, h_{t-1}; \theta),$$

where x_t is the current input, and the h_{t-1} is the immediate hidden state predecessor. Recurrent neural networks (RNN) can handle sequences of arbitrary length. It is, however, not trivial to effectively utilize the sequence content as its length grows while the model capacity remains the same [107]. We refer readers to Greff et al. [45] to learn more details about the LSTM variants as well as alternative RNN designs, such as Gated Recurrent Unit (GRU) [22].

The transformer neural network [143] does not involve recurrence as the RNN does, but it relies on the self-attention mechanism to learn dependencies between token-level representations. Self-attention alone can't introduce the position information at a particular time step, thus some variant of special positional tokens with corresponding embeddings is used to meet this goal.

2.4 Training the model

Training the model usually means solving an optimization problem. It aims to find the most optimal configuration of parameters θ used by the model. In this thesis, we rely on gradient-based optimization to estimate parameters θ with respect to the given training objective J. We assume that the objective is differentiable and the gradient (or the subgradient) can be computed using the backpropagation algorithm [123].

Maximum likelihood estimation The usual choice for the training criterion is the maximum likelihood estimator. This means we aim to pick the parameters θ_{mle} that assign the highest probability to the training set. This can be formulated equivalently as minimizing the negative log-likelihood loss. The log-likelihood is a function of the parameters θ. It says how likely we can observe the data given θ:

$$J(D;\theta) = -\sum_{n=1}^{|D|} \log p_\theta(\mathbf{y}_n|\mathbf{x}_n) = -\sum_{n=1}^{|D|}\sum_{t=1}^{|\mathbf{y}_n|} \log p_\theta(y_{t,n}|y_{<t,n}, \mathbf{x}_n),$$

where D is the training dataset. This definition assumes that the training pairs were sampled independently from the same unknown distribution.

We use stochastic gradient descent (SGD) [119] to update θ iteratively using a reasonably sized minibatch of samples:

$$\theta_{t+1} = \theta_t - \mu_t \cdot \nabla_\theta J(D_t;\theta),$$

where μ_t is the step size at time t and D_t is a minibatch of samples from D. Gradient descent may have an undesirable convergence rate, and alternative optimizers, such as Adam [59] or LAMB [158], are used to overcome such issues.

Teacher forcing and exposure bias Lets consider a pair $(\mathbf{y}^*, \mathbf{x}^*) \in D$, the log-likelihood of these target tokens equals $\sum_{t=1}^{|\mathbf{y}^*|} \log p_\theta(y^*_t | y^*_{<t}, \mathbf{x}^*)$. At time step t, the model does not necessarily predicts the ground-truth token y^*_t. Then what should we use as the prefix history at time step $t+1$? The teacher forcing approach suggests to feed only the tokens from the ground-truth prefix during training. This introduces a discrepancy between training and decoding, because the model is likely to produce a prefix which it has never seen during training. Scheduled sampling [12, 92] is one example of how to interpolate between ground-truth history and model predictions during training. We discuss this discrepancy in detail in the context of repetition degeneracy in Chapter 4 and propose sequence-level objective which utilizes decoding during training. Edunov et al. [32] analyze classical sequence-level objectives with sequence to sequence models.

Regularization and stopping criteria Picking parameters θ with the lowest negative log-likelihood loss may result in overfitting to the training dataset. Regularization techniques, such as dropout [137], are used to regulate the training, so overfitting can be partially mitigated. In addition to that, we early-stop [114] the training once the stopping criterion is met. The stopping criterion value is usually chosen in an ad-hoc fashion. For instance, it can be the gap between training and validation losses. Once the validation loss becomes higher than that computed during training, we early-stop and return the latest best parameters in terms of the validation loss.

2.5 Searching for a prediction given an unseen input

Once the model is trained, we use it to generate a sequence such that it has high quality with respect to the task and evaluate it against a given reference sequence if it is available. For example, in machine translation, we seek for a sequence which resembles the correct translation of the input sentence. In dialogue modeling, we expect the dialogue response which will engage a dialogue partner to continue the conversation. There are far fewer constraints on the desired output in open-ended sequence completion. In this case, the predicted sequence is expected to resemble a continuation from the underlying unknown target distribution.

There are many ways to describe the process of finding or generating a prediction. In this chapter, we describe it on the high level as given the input sequence $\mathbf{x}$ we search for the best candidate $\mathbf{h}$ out of finite hypothesis space H using some scoring function $S(\mathbf{h}, \mathbf{x})$:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{h} \in H} S(\mathbf{h}, \mathbf{x}), \tag{2.1}$$

which we name the maximum score decision rule. In other words, the score function S re-ranks sequences from H according to itself. We use the model to construct H and S and doing so recovers commonly used decoding strategies and decision rules across NLP tasks. Terms "search algorithm", "decoding strategy" and "decision rule" are sometimes used interchangeably, so it is important to bear in mind the difference between the way to approximate H and the way we choose the best candidate from it using the scoring function S.

2.5.1 Approximating hypothesis space H

The trivial and virtually intractable way of representing H is to assume the entire space of all possible sentences up to some predefined length. In addition, the majority of hypotheses in H will be completely irrelevant to the input sequence $\mathbf{x}$. Alternatively, the ideal way to define H is to use high-likely candidates with respect to the real target distribution, which is unavailable to start with. Fortunately, we may rely on the model which is trained to match the empirical data distribution as closely as possible. Here we divide commonly used decoding strategies approximating H into deterministic and stochastic algorithm groups.

Deterministic decoding Beam search and greedy search (a special case of beam search with beam size 1) are the most widely used decoding strategies. Given a model distribution p_θ, beam search approximately solves the following optimization problem:

$$h_{\text{beam}} = \hat{\mathbf{y}}_{\text{beam}} \approx \arg\max_{\mathbf{y} \in \mathcal{Y}} p_\theta(\mathbf{y}|\mathbf{x}),$$

where $\mathcal{Y}$ is the set of all sequences with non-zero probability under the model distribution. Beam search can be seen as a truncated variant of breadth-first search where the breadth at every step is truncated to the given beam size. Beam size controls how large is the search space and the resulting candidate set H which is usually set to be equal to the beam size. While it may seem that we want to use as large beam size as possible, in reality sequences with the highest probability may not be that representative in terms of the performance metrics [62, 138]. We discuss one of such examples in the context of neural machine translation in Chapter 3. A more detailed definition of beam search in the context of sequence completion and conversation modeling is given in Chapter 5 and Chapter 6, respectively.

Stochastic decoding In contrast to deterministic decoding approximately maximizing the probability of a candidate, the stochastic variant aims to sample a sequence from the model distribution or some truncated variant of it. For example, the ancestral sampling procedure gives an unbiased sample from an autoregressive model by sampling each token step-by-step:

$$h_{\text{anc}} = \hat{\mathbf{y}}_{\text{anc}} \sim p_\theta(\mathbf{y}|\mathbf{x}),$$

$$\hat{y}_{\text{anc},t} \sim p_\theta(y_t|\hat{y}_{\text{anc},<t}, \mathbf{x}),$$

where $\hat{y}_{\text{anc},<t}$ is the so far sampled prefix of tokens. This algorithm constructs the sequence candidate by sampling a token step by step following the autoregressive decomposition discussed earlier. By sampling a large number of sequences, we can form a group of representative samples of the given model. For example, sampling-based minimum Bayes risk decoding uses such an approach to construct H as well as to construct S [33, 39].

Ancestral sampling may produce sequences that are too random or irrelevant with respect to the given input [52]. Lowering the temperature of a token-level softmax distribution (thus altering the sequence-level model distribution) is one way of preventing such rare outcomes to be sampled. Alternatively to the temperature tuning, nucleus sampling [52] and top-k sampling [35] truncates the tail of the token-level distribution in a specific way. We discuss these algorithms in detail later in Chapter 5.

2.5.2 Scoring functions S

Here we highlight some relevant scoring functions which are used at the decoding stage across different NLP tasks. We use maximum a posteriori (MAP) scoring in this thesis, while minimum Bayes risk (MBR) and external reranking functions have received attention in the context of machine translation recently [39, 72].

Maximum a posteriori (MAP) scoring Maximum a posteriori naming comes from the statistical estimator named the same way. In our context, we do not directly deal with the prior distribution and the likelihood to compute the posterior, as in the Bayesian view. Here the posterior is the model $p_\theta(\mathbf{y}|\mathbf{x})$, i.e., what is the probable sequence $\mathbf{y}$ given the observation $\mathbf{x}$? The scoring function then is written as:

$$S_{\text{MAP}}(\mathbf{h}, \mathbf{x}) = p_\theta(\mathbf{y} = \mathbf{h}|\mathbf{x}).$$

In the literature, we can often find the term "MAP decoding" which attributes to beam search and MAP scoring combined in the decision rule [33, 138].

Minimum Bayes risk (MBR) scoring Minimum Bayes risk scoring computes the negative Bayes risk B, where negative sign is introduced so that the scoring function can be used with max scoring rule we defined earlier in Equation (2.1). The underlying idea is to compute the expected risk with respect to the unknown target distribution p^*. The unknown p^* is then approximated using the model distribution p_θ. Finally, the expectation is approximated using the Monte-Carlo estimator, because computing the expectation exactly is hardly feasible given the model size and the sequence space size. The MBR scoring function is then written as:

$$\begin{aligned} S_{\text{MBR}}(\mathbf{h}, \mathbf{x}) = -B(\mathbf{h}, \mathbf{x}) = -\mathbb{E}_{\mathbf{y}' \sim p^*} U(\mathbf{h}, \mathbf{y}') \\ \approx -\mathbb{E}_{\mathbf{y}' \sim p_\theta} U(\mathbf{h}, \mathbf{y}') \approx -\frac{1}{|\Omega|} \sum_{n=1}^{|\Omega|} U(\mathbf{h}, \mathbf{y}_n), \end{aligned}$$

where $U(\mathbf{h}, \mathbf{y}')$ is a function which computes some distance on the sequence-level between the hypothesis $\mathbf{h}$ and the sample from the model $\mathbf{y}'$.

2.6 Evaluation

As we discussed earlier, the evaluation stage is necessary to quantify the performance of a given model or to choose the final model among a larger set. Let's consider the machine translation task. Ideally, we want to hire professional translators and ask them to judge the translations given by our models. How would they do that? For instance, they translate the input sentence on their own and compare it with the model output. Even in this case there can be ambiguity: different translators are likely to come up with slightly different translations given the same input sentence. Freitag et al. [38] investigate this uncertainty in detail. More interactive tasks can be even harder to evaluate. In Chapter 6, we discuss the importance of careful human evaluation in multi-turn dialogue modeling and show the evidence of annotator bias.

Let's assume we can obtain human judgement scores. It would be pretty expensive to hire translators to evaluate every single model we produce as part of a research project or other development. To overcome this, the research community comes up with automatic evaluation tools which are expected to correlate with human judgement. The *de facto* standard machine translation metric is BLEU [106]. It is based on the n-gram overlap between the model's translation and the generated one. Neural based metrics, such as BertScore, BLEURT, and COMET, have become more popular recently [118, 127, 162]. These neural metrics were shown to correlate better with human judgement compared to overlap-based metrics.

3 Characterizing and addressing the issue of oversmoothing

3.1 Introduction

Neural autoregressive sequence modeling is a widely used scheme for conditional text generation. It is applied to many NLP tasks, including machine translation, language modeling, and conversation modeling [17, 21, 121, 142]. Despite the substantial success, major issues still exist, and it is still an active area of research. Here we highlight two major issues which have been discussed extensively.

The first issue is the model assigning too high a probability to a sequence which is unreasonably shorter than a ground-truth sequence. Stahlberg and Byrne [138] report evidence of an extreme case where the model frequently assigns the highest probability to an empty sequence given a source sequence in machine translation. In addition, Koehn and Knowles [62] demonstrate that the length of generated translation gets shorter with better decoding (i.e., beam search with a larger beam.)

In the second issue, which is more often observed in open-ended sequence generation tasks, such as sequence completion, generated sequences often contain unreasonably many repetitions [52, 150]. This phenomenon was partly explained in a recent year by Welleck et al. [148], as approximate decoding resulting in an infinitely long, zero-probability sequence.

In this work, we tackle the first issue where the model prefers overly short sequences compared to longer, often more correct ones. We assume that any prefix substring of a ground-truth sequence is an unreasonably short sequence and call such a prefix as a premature sequence. This definition allows us to calculate how often an unreasonably short sequence receives a higher probability than the original, full sequence does. This value quantifies the degree to which the probability mass is oversmoothed toward shorter sequences. We call this quantity an *oversmoothing rate.* We empirically verify that publicly available, well-trained translation models exhibit high oversmoothing rates.

We propose to minimize the oversmoothing rate during training together with the negative log-likelihood objective. Since the oversmoothing rate is difficult to minimize directly due to its construction as the average of indicator functions, we design its convex relaxation, to which we refer as an *oversmoothing loss.* This loss is easier to use with gradient-based learning.

We apply the proposed regularization to neural machine translation using IWSLT'17 and WMT tasks and observe promising findings. We effectively reduce the oversmoothing rate by minimizing the proposed oversmoothing loss across all tasks we consider. We see the narrowing gap between the length distribution of generated sequences and that of the reference sequences, even when we increase the beam size, with a lower oversmoothing rate. Finally, by choosing the strength of the proposed regularization appropriately, we improve the translation quality when decoding with large beam sizes. We could not, however, observe a similar improvement with a small beam size.

3.2 Background: Neural autoregressive sequence modeling

We study how a neural sequence model assigns too high probability to unreasonably short sequences due to its design and training objective. We do so in the context of machine translation in which the goal is to model a conditional distribution over a target language given a source sentence. More specifically, we consider a standard approach of autoregressive neural sequence modeling for this task of neural machine translation, where the conditional probability of a target sentence given a source sentence is written down as:[1]

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t|y_{<t}, \mathbf{x}; \theta), \tag{3.1}$$

where $y_{<t}$ is a sequence of tokens up to (and not including) step t. θ refers to the parameters of an underlying neural network that computes the conditional probability. Each of the source and target sentences ends with a special ⟨eos⟩ token indicating the end of the sequence. As was demonstrated by Newman et al. [98], this ⟨eos⟩ token is used by an autoregressive neural network to model the length of a sequence.

Given this parametrization, we assume a standard practice of maximum likelihood learning which estimates the parameters θ that maximizes the following objective function:

$$L(\theta) = \frac{1}{|D|} \sum_{n=1}^{N} \log p(\mathbf{y}^n|\mathbf{x}^n; \theta) + \mathcal{R}(\theta).$$

$\mathcal{R}$ is a regularization term that prevents overfitting, such as weight decay.

Once training is done, we use this autoregressive model as a translation system by approximately solving the following optimization problem:

$$\hat{\mathbf{y}}_{\text{map}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}; \theta).$$

We often resort to greedy decoding or beam search, both of which belong to a family of incomplete decoding algorithms [148].

[1]In the rest of the chapter, we often omit X for brevity.

3.3 Oversmoothing: the issue of premature sequences

In this section, we carefully describe the issue of premature translation or premature sequence in autoregressive modeling, which has more often been referred to casually as the issue of oversmoothing in earlier studies [see, e.g., 134]. To do so, we first define formally what we mean by a 'premature sequence'. A premature sequence is a length-t prefix of an original sequence, where t is smaller than the length of the original sequence. In other words, length-t prefix is defined as:

Definition 3.3.1 (Length-t prefix)**.** *Given an original sequence* $\mathbf{y} = (y_1, y_2, \ldots, y_T = \langle\text{eos}\rangle)$*, the length-t prefix is* $\mathbf{y}_{\leq t} = (y_1, y_2, \ldots, y_{t-1}, \langle\text{eos}\rangle)$*, where* $1 \leq t < T$*.*

With this definition, we make a reasonable assumption that most of such premature sequences are not valid sequences on their own. In the case of natural language processing, for instance, these premature sequences correspond to sentences that suddenly terminate in the middle. Only a few of these premature sequences may be a coherent, well-formed text.

A good autoregressive language model should then assign a lower probability to such an ill-formed premature sequence than that assigned to a well-formed original sequence. That is, it must satisfy:

$$\underbrace{\prod_{t'=1}^{T} p(y_{t'}|y_{<t'})}_{=p(\mathbf{y})} > \underbrace{p(\langle\text{eos}\rangle\,|y_{<t}) \prod_{t'=1}^{t-1} p(y_{t'}|y_{<t'})}_{=p(\mathbf{y}_{\leq t})} \tag{3.2}$$

which is equivalent to

$$\prod_{t'=t}^{T} p(y_{t'}|y_{<t'}) > p(\langle\text{eos}\rangle\,|y_{<t}),$$

because of the autoregressive formulation.

In order for this inequality to hold, the probability assigned to the $\langle\text{eos}\rangle$ must be extremely small, as the left-hand side of the inequality is the product of many probabilities. In other words, the dynamic range of the $\langle\text{eos}\rangle$ token probability must be significantly greater than that of any other token probability, in order for the autoregressive language model to properly capture the ill-formed nature of premature sequences.

It is, however, a usual practice to treat the $\langle\text{eos}\rangle$ token just like any other token in the vocabulary, which is evident from Equation (3.1). This leads to the difficulty in having a dramatically larger dynamic range for the $\langle\text{eos}\rangle$ probability than for other token probabilities. In other words, this limited dynamic range due to the lack of special treatment of $\langle\text{eos}\rangle$ is what previous studies [134] have referred to as "oversmoothing", and this leads to the degeneracy in length modeling.

Under this observation, we can now quantify the degree of oversmoothing[2] by examining how often the inequality in Eq. (3.2) is violated:

Definition 3.3.2 (Oversmoothing rate)**.** *The oversmoothing rate of a sequence is defined as*

$$r_{\text{os}}(\mathrm{y}) = \frac{1}{|\mathrm{y}| - 1} \sum_{t=1}^{|\mathrm{y}|-1} \mathbb{1}\Big(\prod_{t'=t}^{|\mathrm{y}|} p(y_{t'} | y_{<t'}) < p(\langle eos \rangle \, | y_{<t}) \Big), \tag{3.3}$$

where $\mathbb{1}$ *is an indicator function returning* 1 *if true and otherwise* 0*.*

With this definition, we can now quantify the degree of oversmoothing and thereby quantify any improvement in terms of the issue of oversmoothing by any future proposal, including our own in this chapter.

Because premature sequences may be well-formed, it is not desirable for the oversmoothing rate to reach 0. We, however, demonstrate later empirically that this oversmoothing rate is too high for every system we considered in this work.

[2]To be strict, this should be called the degree of 'smoothing', but we stick to oversmoothing to be in line with how this phenomenon has been referred to in previous studies [134].

3.3.1 Minimizing the oversmoothing rate

The oversmoothing rate above is defined as the average of indicator functions, making it challenging to directly minimize. We instead propose to minimize an upper bound on the original oversmoothing rate, that is differentiable almost everywhere and admits gradient-based optimization:

Definition 3.3.3 (Oversmoothing loss)**.** *Given a sequence* $\mathbf{y}$*, the oversmoothing loss is defined as*

$$l_{\text{os}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} \max\left(0, \log p(\langle eos\rangle \,|y_{<t}) - \sum_{t'=t}^{|\mathbf{y}|} \log p(y_{t'}|y_{<t'}) + m\right),$$

which is an upper bound of $r_{\text{os}}(y)$ *with* $m \geq 1$.

We use this oversmoothing loss as a regularization term and augment the original objective function with it. We use $\alpha \in [0,1)$ to balance the relative strengths of these two terms:

$$l(\mathbf{y}) = (1-\alpha) \cdot l_{\text{nll}}(\mathbf{y}) + \alpha \cdot l_{\text{os}}(\mathbf{y}),$$

where $l_{\text{nll}}(\mathbf{y}) = -\sum_{t=1}^{|\mathbf{y}|} \log p(y_t|y_{<t})$.

When the inequality in Eq. (3.2) is satisfied at step t with the log-probability difference between the l.h.s. and r.h.s. at least as large as m, the oversmoothing loss disappears, implying that the step t does not contribute to the issue of oversmoothing. When this loss is activated at step t, we have two terms, excluding the constant margin m, the log-probability of *incorrect* ⟨eos⟩ given the context $y_{<t}$ and the negative log-probability of the *correct* suffix given the same context.

Minimizing the first term explicitly prevents a premature sequence $\mathbf{y}_{\leq t}$ from being a valid sequence by lowering the probability y_t being ⟨eos⟩ even further compared to the other tokens in the vocabulary. The second term on the other hand prevents the premature sequence by ensuring that the full sequence $\mathbf{y} = (y_{<|y|}, \langle\text{eos}\rangle)$ is more likely than the premature sequence $\mathbf{y}_{\leq t} = (y_{<t}, \langle\text{eos}\rangle)$. In short, the proposed oversmoothing loss addresses both of these scenarios which lead to oversmoothing. Finally, only when both of these factors are suppressed enough, the loss vanishes.

The second scenario above, i.e., increasing the probability of a suffix at each position t, has the effect of greatly emphasizing the latter part of the sequence during training. This can lead to a degenerate case in which the earlier part of a sequence cannot be modeled by an autoregressive sequence modeling, if the strength of the proposed oversmoothing loss is too large. We thus use this loss together with the original negative log-likelihood loss ($\alpha > 0$) only after pretraining a model with the negative log-likelihood loss only ($\alpha = 0$).

3.4 Related work

The issue of generating sequences that are shorter than the ground-truth one has been studied from various aspects including model parametrization, data collection, and decoding. Here we highlight some of these projects in the context of our work.

On the aspect of model parametrization, Peters and Martins [110] suggest using sparse transformation of the next-token distribution rather than the usual way of using softmax. Such a model is then able to assign zero probability to short sequences more readily and thereby reduce the oversmoothing rate. Their approach, however, does not explicitly encourage ⟨eos⟩ tokens to be assigned zero probability, unlike ours where ⟨eos⟩ is treated specially. Shi, Xiao, and Knight [134] embed the ⟨eos⟩ token with a distinct vector at each position within the sequence. This was shown to help the probability of empty sequence, although they do not report its impact on translation quality at all.

On data collection, Nguyen, Murray, and Chiang [100] analyze data collection and show that data augmentation techniques altering sequence length may address the issue of oversmoothing and improve translation quality. Their work is however limited to low-resource tasks. With respect to decoding, Murray and Chiang [97] designed a decoding algorithm that learns to correct the underestimated length. Alternative decoding algorithms, such as minimum Bayes risk decoding [33, 96], have been shown to alleviate the length mismatch to a certain extent when compared to beam search.

These earlier approaches do not attempt at formally characterizing the cause behind the issue of oversmoothing. This is unlike our work, where we start by formalizing the issue of oversmoothing and propose a way to alleviate this issue by directly addressing this cause.

3.5 Experimental Setup

We follow a standard practice to train our neural machine translation models, following [105], using the FairSeq framework [104]. We use BPE tokenization via either fastBPE [128] or SentencePiece [65], depending on the dataset. Although it is not required for us to use state-of-the-art models to study the issue of oversmoothing, we use models that achieve reasonable translation quality. The code implementing FairSeq task with the oversmoothing rate metric, oversmoothing loss, and experimental results is available on Github.[3]

3.5.1 Tasks and Models

We experiment with both smaller datasets using language pairs from IWSLT'17 and larger datasets using language pairs from WMT'19 and WMT'16. In the latter case, we use publicly available pretrained checkpoints in FairSeq. We execute five training runs with different random initialization for every system. These language pairs and checkpoints cover different combinations of languages and model sizes. This allows us to study the oversmoothing rate under a variety of different settings.

[3] https://github.com/uralik/oversmoothing_rate

IWSLT'17 {De,Fr,Zh}→En: We adapt the data preprocessing procedure from FairSeq IWSLT recipe and use SentencePiece tokenization. The training sets consist of 209K, 236K, and 235K sentence pairs for De→En, Fr→En, and Zh→En, respectively. We use the TED talks 2010 development set for validation, and the TED talks 2010-2015 test set for testing. The development and test sets, respectively, consist of approximately 800 and 8,000 sentence pairs for all tasks.

We use the same architecture named `transformer_iwslt_de_en` in FairSeq for each language pair. It consists of 6 encoder and decoder layers with 4 self-attention heads followed by feed-forward transformations. Both encoder and decoder use embeddings of size 512 while the input and output embeddings are not shared. Both the encoder and decoder use learned positional embedding. We early-stopping training based on the validation set. Evaluation is done on the test set.

WMT'16 En→De: We prepare the data following the recipe from FairSeq Github.[4] The training set has 4.5M sentence pairs. Following Ott et al. [105], we use newstest13 as the development set and newstest14 as the test set, they contain 3K sentence pairs each. We fine-tune the pretrained checkpoint which was originally released by Ott et al. [105] and is available from FairSeq. The recipe uses a transformer architecture based on Vaswani et al. [143]. Different from all other models considered in this work, this architecture shares vocabulary embeddings between the encoder and the decoder.

WMT'19 Ru→En, De↔En We closely follow Ng et al. [99] in preparing data, except for filtering based on language identification. We use the subset of WMT'19 training set consisting of news commentary v12 and common crawl resulting in slightly more than 1M and 2M training sentence pairs for Ru→En and De↔En pairs, respectively. We fine-tuned single model checkpoints from Ng et al. [99].[5] We early-stop training on the official WMT'19 development set. For evaluation,

[4]https://git.io/JDqB2
[5]https://git.io/JDqBo

we use the official WMT'19 test set.

3.5.2 Training

We use Adam optimizer [58] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. We use the inverse square root learning scheduler with 4,000 warm-up steps. We use the initial learning rate of 5×10^{-4}, dropout rate of 0.3 [137] , and weight decay with its rate set to 10^{-4}. We use label smoothing with 0.1 of probability smoothed uniformly during pretraining with NLL loss and turn it off after starting to use the oversmoothing loss. We vary the oversmoothing loss weight α from 0.0 to 0.95 with a step size of 0.05. We use a fixed margin $m = 10^{-4}$ whenever we use the oversmoothing loss.

Early stopping We use early stopping for model selection based on the value of the objective function computed on the development set. We evaluate the model on the development set every 2K updates for IWSLT (~2K tokens per update) and WMT (~9K tokens per update) systems. We stop training when the objective has not improved over more 5 consecutive validation runs. We fine-tune models around 5K updates for IWSLT'17 DE-EN and ZH-EN, and 7K updates for IWSLT'17 FR-EN. As for WMT'19, it takes approximately 45K updates for DE-EN and EN-DE language pairs to early-stop, and 76K updates for RU-EN model, and 12K updates for WMT'16.

3.5.3 Decoding

To test translation quality, we translate a test set with beam search decoding, as implemented in FairSeq. We vary beam sizes to study their effect in-depth. We set the lower- and upper-bound of a generated translation to be, respectively, 0 and $1.2 \cdot l_x + 10$, where l_x is the length of the source x. We do not use either length normalization nor length penalty, in order to study the impact of oversmoothing on decoding faithfully. We compute and report BLEU scores using `sacreBLEU` on detokenized predictions.

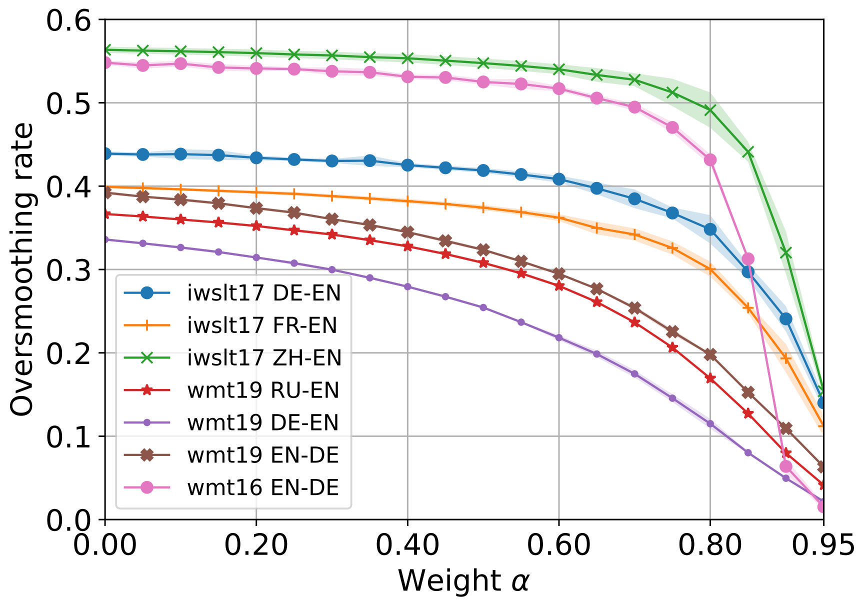
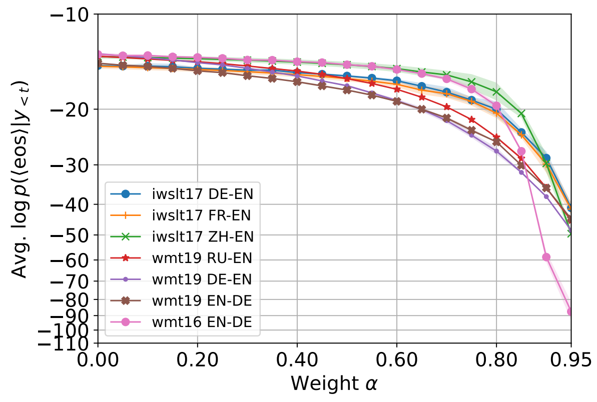


Figure 3.1: Average oversmoothing rate is going down as we increase contribution of the oversmoothing loss during fine-tuning. Filled regions denote the standard deviation across training runs according to Section 3.5.

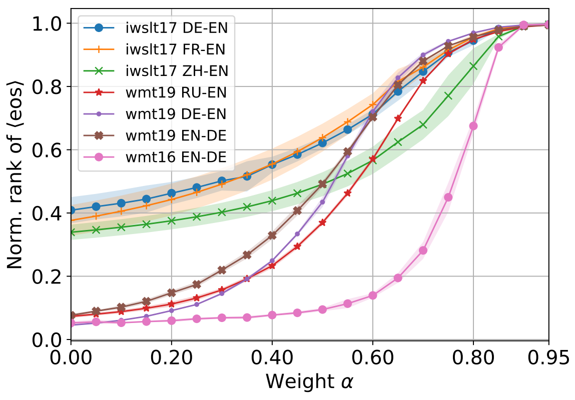
3.6 Experiments

As we pointed out earlier, publicly available translation systems exhibit a high degree of oversmoothing. See the left-most part of Figure 3.1, where $\alpha = 0$. In particular, this rate ranges from **34%** (WMT'19 DE→EN) up to **56%** (IWSLT'17 ZH→EN).

According to Section 3.3.1, the oversmoothing rate should decrease as we increase the relative strength of the oversmoothing loss. To verify this, we fine-tune these models while varying the coefficient α. In Figure 3.1 we demonstrate the oversmoothing rate reduces all the way down to **3%** (WMT'19 DE→EN) and **17%** (IWSLT'17 ZH→EN) as we increase the strength of the regularizer. The oversmoothing rate monotonically decreases for every system we consider, as we increase α up to 0.95.



(a)



(b)

Figure 3.2: (a) Log-probabilities of ⟨eos⟩ token within length-t prefixes averaged across all positions per translation and then averaged across all translations. (b) Normalized rank of ⟨eos⟩ token within length-t prefixes averaged across all positions t per translation and then averaged across all translations. 1 means the lowest rank within the vocabulary. Filled regions denote the standard deviation across training runs according to Section 3.5.

3.6.1 Regularization and ⟨eos⟩ token

Minimizing the proposed oversmoothing loss minimizes the log-probability of ⟨eos⟩ token at the end of every length-t prefix unless it is already low enough. We analyze how the strength of regularization affects the average log-probability of ⟨eos⟩ token measured at the end of each premature translation. As presented in Figure 3.2 (a), the log-probability of ⟨eos⟩ at the end of premature sequences decreases monotonically as the oversmoothing rate decreases (i.e., as the strength of the oversmoothing loss increase, as evident from Figure 3.1).

Although the log-probability of ⟨eos⟩ is an important factor in oversmoothing, Welleck et al. [148] claim that it is the rank of ⟨eos⟩ token that matters when using an incomplete approximate decoding strategy, such as beam search, for generation. We thus look at the average normalized rank of ⟨eos⟩ token at the end of every length-t prefix in Figure 3.2 (b). The rank drops rapidly and almost monotonically as we add more regularization. The effect of regularization is more visible with the rank than with the log-probability, especially when α is small.

Although the proposed regularization reduces the probability of ⟨eos⟩ token where it is not

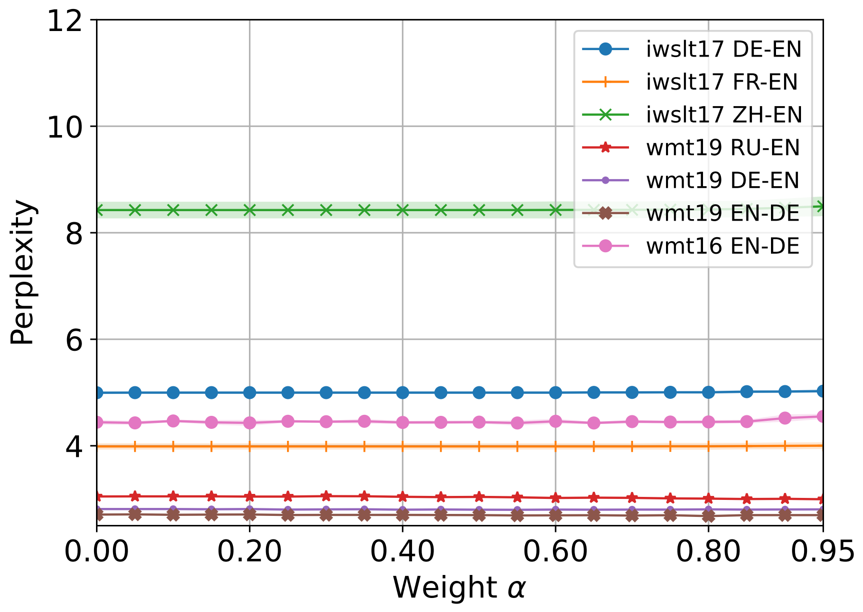
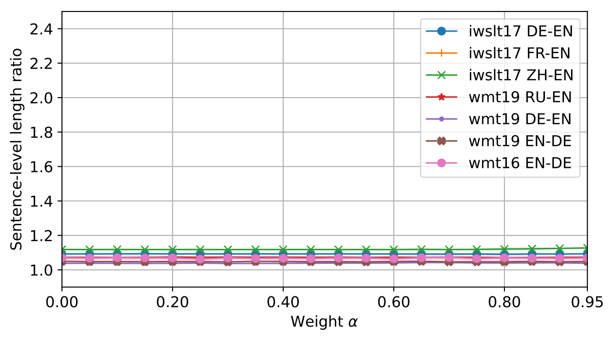


Figure 3.3: Perplexity measured on reference translations remains stable as we increase the strength of the regularization. Filled regions denote the standard deviation across training runs according to Section 3.5.

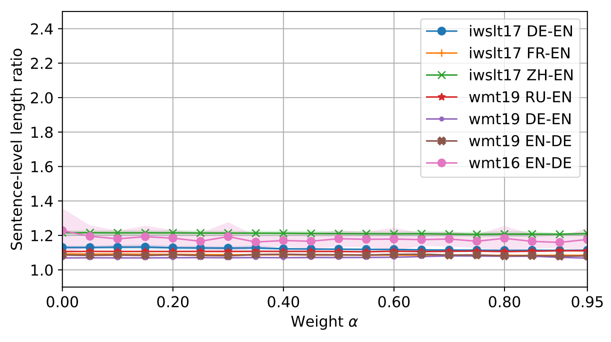
supposed to be, we observe that the performance of the system as a language model does not degrade much regardless of the chosen value of α. This is evident from the flat lines in Figure 3.3 where we plot the perplexity of each model while varying α. This demonstrates that there are many different ways to minimize the negative log-likelihood, and some of those solutions exhibit a higher level of oversmoothing than the others. The proposed oversmoothing loss is an effective way to bias the solution toward a lower level of oversmoothing.

3.6.2 Oversmoothing rate and decoding

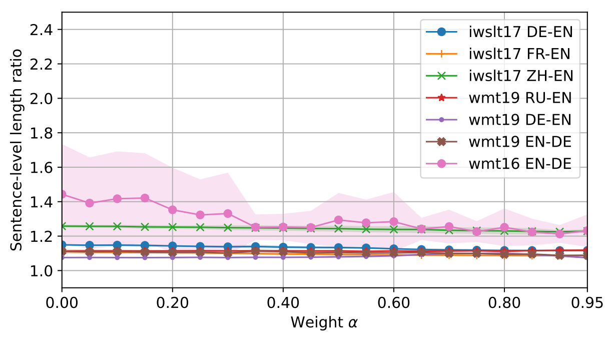
Earlier Koehn and Knowles [62] noticed this issue of oversmoothing by observing that the length of generated sequences dramatically dropped as the beam width increased. We confirm the decreasing length of generated translation as the beam size increases in Figure 3.4 when $\alpha = 0$. We study the change of this length as we add more regularization and calculate the sequence-level length ratio in Figure 3.4.



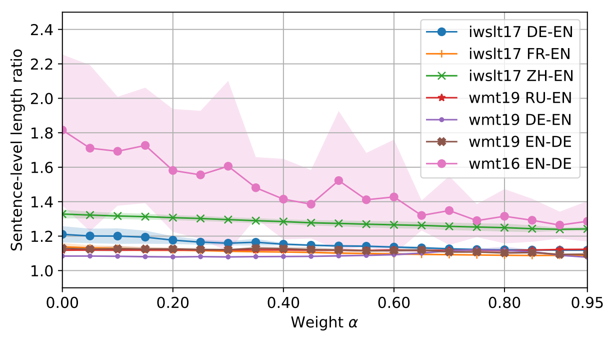
(a) beam 5



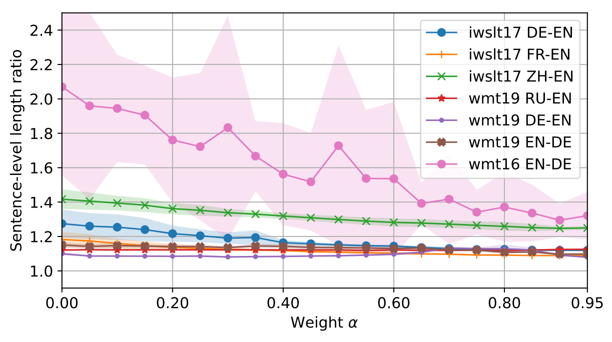
(b) beam 100



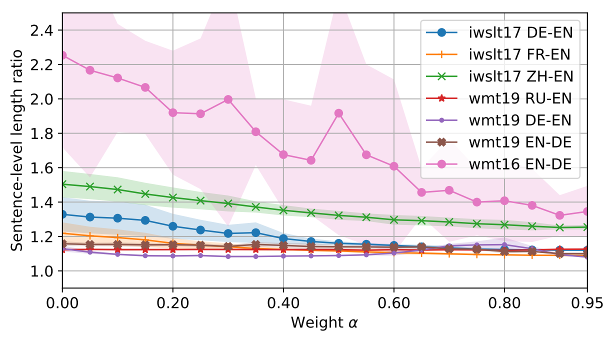
(c) beam 250



(d) beam 500



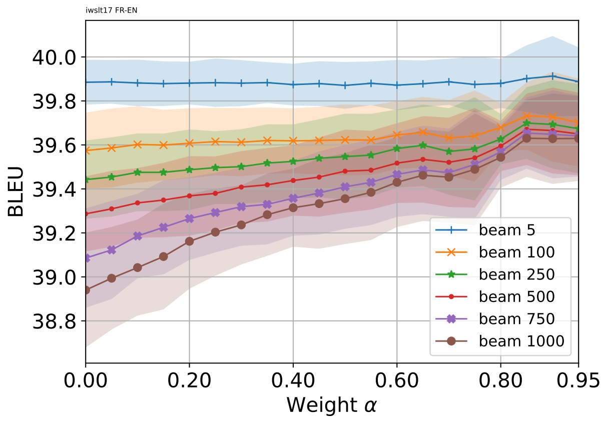
(e) beam 750



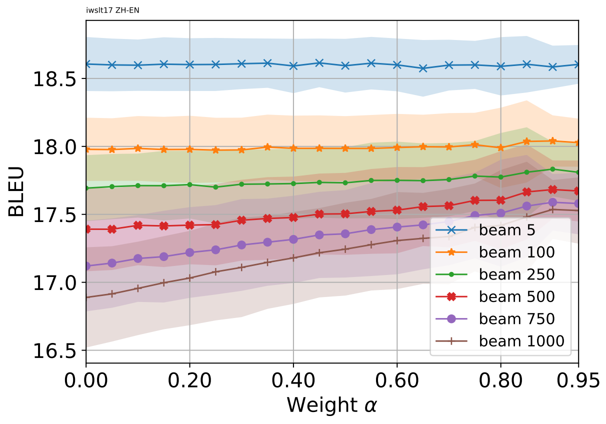
(f) beam 1000

Figure 3.4: Sentence-level length ratio is $\frac{1}{|D_{\text{test}}|}\sum_{i=1}^{|D_{\text{test}}|}|\mathbf{y}_i^{\text{ref}}|/|\mathbf{y}_i^{\text{beam}}|$, where $\mathbf{y}_i^{\text{beam}}$ is generated translation using beam search for i-th input sentence from the test set D_{test}, and $\mathbf{y}_i^{\text{ref}}$ is the corresponding reference translation. Filled regions denote the standard deviation across training runs according to Section 3.5.

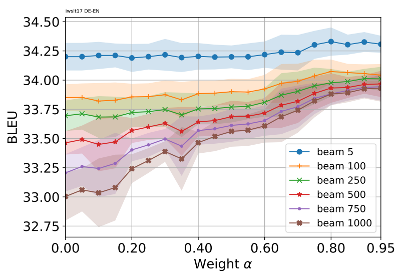
When fine-tuned with the proposed oversmoothing loss, the length ratio degrades significantly less, as we increase the beam size during decoding, than without. For instance, with $\alpha \geq 0.8$ the length ratio remains more or less constant with respect to the size of the beam. Despite the observed robustness, decoding with a smaller beam size produces translations with



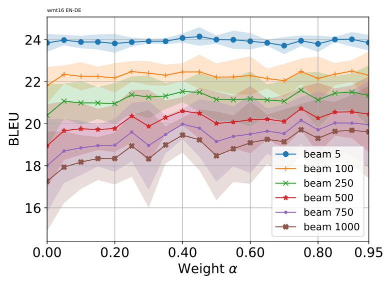
(a) IWSLT'17 FR→EN



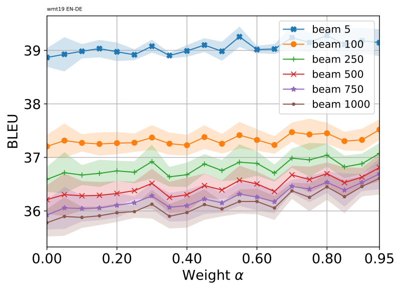
(b) IWSLT'17 ZH→DE



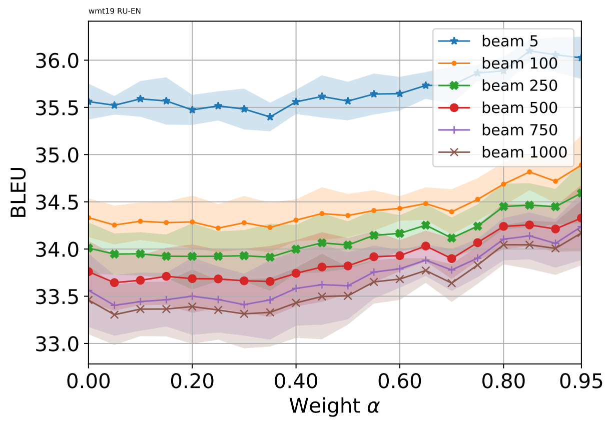
(c) IWSLT'17 DE→EN



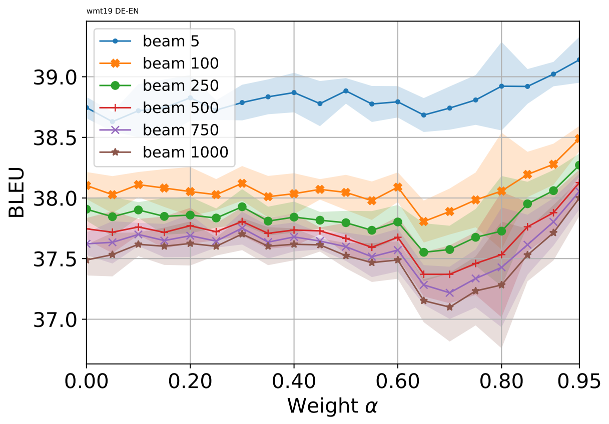
(d) WMT'16 EN→DE



(e) WMT'19 EN→DE



(f) WMT'19 RU→EN



(g) WMT'19 DE→EN

Figure 3.5: BLEU score is measured on corresponding test sets. Decoding is done using beam search with beam sizes given in the legend. Section 3.5 provides more details on test sets and decoding hyper-parameters. Filled regions denote the standard deviation across training runs according to Section 3.5.

lengths which match reference lengths better regardless of the strength of regularization.

Translation quality The quality of the produced translation is directly related to its length, because this length needs to closely match the length of the reference translation. However, the length information is not sufficient to make a conclusion about the translation quality. We quantify the quality of the translation by calculating the corpus-level BLEU score. The scores in Figure 3.5 indicate that the reduced degradation of length modeling does correlate with the improvements in translation quality, although the degree of such correlation varies across language pairs and beam widths. We highlight two major aspects of the effect of regularization on the translation quality. First, the impact of regularization is only visible when the beam size is substantially larger than what is commonly used in practice. Second, the degradation of translation quality with a larger beam size lessens as oversmoothing does as well, but it does not eliminate the degradation fully. These observations imply that the effectiveness of approximate decoding in neural machine translation remains unsolved, despite our successful attempt at addressing the issue of oversmoothing.

3.7 Conclusion

In this work, we tackled a well-reported issue of oversmoothing in neural autoregressive sequence modeling, which has evaded rigorous characterization until now despite of its ubiquity. We characterized it by defining the oversmoothing rate. It computes how often the probability of the ground-truth sequence is lower than the probability of any of its prefixes. We confirmed that the oversmoothing rate is too high among well-trained neural machine translation systems and proposed a way to directly minimize it during training. We designed a differentiable upper bound of the oversmoothing rate called the oversmoothing loss. We experimented with a diverse set of neural machine translation systems to study the effect of the proposed regularization.

The experiments revealed several findings and takeaways. First, the oversmoothing loss is effective: we were able to monotonically decrease the oversmoothing rate by increasing the strength of the loss. Second, we found that this regularization scheme significantly expands the dynamic range of the log-probability of ⟨eos⟩ token and has even greater impact on its rank, without compromising on sequence modeling. Third, the proposed approach dramatically alters the behaviour of decoding when a large beam width was used. More specifically, it prevents the issue of degrading length ratio and improves translation quality. These effects were not as apparent with a small beam size though. The proposed notion of oversmoothing explains some of the degeneracies reported earlier, and the proposed mitigation protocol alleviates these degeneracies. We, however, find that the proposed approach could not explain a more interesting riddle, that is, the lack of improvement in translation quality despite lower oversmoothing when beam search with a smaller beam was used. This unreasonable effectiveness of beam search continues to remain a mystery and needs to be investigated further in the future.

4 Neural text generation with unlikelihood training

4.1 Introduction

Neural text generation is a vital tool in a wide range of natural language applications. However, the standard approach – training a sequence to sequence model, e.g., Transformer [143], to maximize log-likelihood and approximately decoding the most likely sequence – is known to be flawed. Generated text in open-ended applications such as language modeling or dialogue has been observed to be dull, with high frequency tokens used too often and interesting content words used too rarely [30, 52]. Moreover, the models repeat themselves at the token, phrase, and sentence levels, and statistics comparing a set of human-generated utterances and model-generated responses indicate a discrepancy between the human and model word distributions. This does not appear to be rectified by training on more data [116]. Recent fixes involve modifying the decoding strategy using sampling or more sophisticated beam search variants. However, these decoding strategies do not address the core issue: the model's underlying sequence probabilities are clearly not correct.

Several reasons for exactly why neural text is degenerate have been posited, with the cause currently unknown. Possible candidates include the problem being (i) a by-product of the model architecture, e.g., the Transformer architecture preferring repeats [52, 144], (ii) an intrinsic property of human language [52] rather than a modeling deficiency, or that (iii) a training objective relying on fixed corpora cannot take into account the real goal of using the language [23]. Our work shows that, while the above may be factors, a primary factor is the use of the likelihood objective itself, as we demonstrate that degeneration is alleviated if we replace the likelihood objective with our proposal.

While low perplexity in the limit should lead to predicting the correct next target word, there are two major flaws of the likelihood objective: (i) it pays relatively little attention to the argmax or the top of the ranked list of next token probabilities, instead optimizing the likelihood of the entire distribution; (ii) it is not focused on optimizing sequence generation, only on producing the next token. The first issue means that greedy or beam search decoding, which rely on the top of the list to generate, are not optimized – there is a discrepancy between maximizing the log-probability of a ground-truth token and ensuring the rank of the ground-truth token to be one. The second issue means that during sequence generation, any imperfection in next token prediction leads to error accumulation that is not addressed by likelihood training.

In this work, we introduce *unlikelihood training*, an approach that addresses the two aforementioned issues. It combines two types of updates: a likelihood update on the true target tokens so that they are assigned high probability, and an *unlikelihood* update on tokens that are otherwise assigned too high a probability. We can collect these *unlikely* token candidates either during next-token prediction or from generated sequences, allowing us to train at both the token and sequence levels. Both token and sequence level unlikelihood training are shown to improve metrics that measure dullness and repetition of the model, while maintaining performance in other metrics such as perplexity or token accuracy compared to the maximum likelihood baseline. Finally, we assess our models using human evaluations. We find that our generations have vastly improved quality compared to likelihood trained models when both models use beam search decoding. Moreover, our approach when using beam search also significantly improves over likelihood trained models using either beam blocking or nucleus sampling, thus outperforming the current state-of-the-art.

4.2 Related Work

Neural Text Degeneration Recently, several papers have observed various forms of *neural text degeneration*, especially in open-ended generation tasks. In dialogue, it has been shown that there is a mismatch between model and human word distributions, where generative models are more likely to output frequent words, but less likely to produce rare words compared to humans. For example, this was observed across all generative models submitted to the ConvAI2 NeurIPS 2018 competition [30]. In language modeling, the work of [52] highlighted problems with the word frequency distribution and level of repetition in model generations compared to human text. These issues are not remedied by simply increasing the amount of the training data; e.g., large-scale GPT-2 language models [116] display the same issues.

Improved Decoding Algorithms Several methods have been proposed to rectify these issues. The primary ones involve changing the decoding method to a sophisticated beam search variant or to stochastic decoding, e.g., sampling. Different variants of beam search have been explored [51, 69, 145] which can decrease a model's level of repetition by selecting candidates that are unlike previously chosen ones. Separately, hard or soft beam blocking has been investigated [61, 108], whereby previously generated *n*-grams are blocked from subsequent generation. This approach is often used in dialogue generation, fixing some token or phrase level repetitions but removing repetitions that would naturally occur in human text.

The second major approach is that of sampling from the model at generation time. Top *k*-sampling [36] and nucleus sampling [52] are two methods that sample sequences based on a function of the predicted next token probability distribution given by the model. Both approaches vastly improve the repetition issue, as the randomization often reduces the number of duplicate tokens in a decoded sequence, even if highly scored paths under the model (represented by beam search candidates) contain repetitions. However, as the underlying model is unchanged, it often prefers semantically similar phrasing, depending on the temperature parameter of the sampling [52]. Furthermore, this solution is less relevant in less open-ended tasks such as machine translation, where beam search variants are the preferred method. Ideally we would like a model that can work with both beam and sampling decoding methods.

Improved Learning Algorithms The proposed learning criteria are closely related to structured output prediction methods in which the goal is to increase the scores assigned by a model to true examples while decreasing those assigned to negative examples often generated by the model itself. Some representative algorithms include structured perceptron [24], energy-based models [71] and more recently reflective likelihood [29]. A particular variant in this family of algorithms, called negative training, was recently used by He and Glass [47] to prevent generic and malicious responses in dialogue models. Similarly, these structured prediction algorithms with neural language models have been applied to machine translation in recent years by Shen et al. [132] and Edunov et al. [32].

4.3 Neural Text Generation

Language Modeling In language modeling, our goal is to model a probability distribution $p_*(\mathbf{x})$ over variable-length text sequences $\mathbf{x} = (x_1, \ldots, x_{|\mathbf{x}|})$ composed of tokens from a vocabulary, $x_t \in \mathcal{V}$. We wish to find a model $p_\theta(\mathbf{x})$ which resembles $p_*(\mathbf{x})$, meaning that samples $\hat{x} \sim p_\theta$ are similar to samples from p_*, and $p_\theta(\mathbf{x}) \approx p_*(\mathbf{x})$ for all $\mathbf{x}$. When $p_\theta(\mathbf{x})$ is parameterized by a neural network, we call p_θ a neural language model. We assume that p_θ takes the form $p_\theta(\mathbf{x}) = \prod_{t=1}^{|\mathbf{x}|} p_\theta(x_t|x_{<t})$.

The *de facto* approach to training such a model is to find parameters θ that maximize the log-likelihood of a finite set of samples $\mathcal{D}$ from p_* by minimizing:

$$\mathcal{L}_{\text{MLE}}(p_\theta, \mathcal{D}) = -\sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{|\mathbf{x}^{(i)}|} \log p_\theta(x_t^{(i)}|x_{<t}^{(i)}). \tag{4.1}$$

Sequence Completion A closely related problem consists of sampling a sub-sequence, or **prefix**, $\mathbf{x}_{1:k} \sim p_*$, then using p_θ to conditionally decode a **continuation**, $\hat{\mathbf{x}}_{k+1:N} \sim p_\theta(\cdot|\mathbf{x}_{1:k})$. We now want the resulting **completion** $(x_1, \ldots, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_N)$ to resemble a sample from p_*.

We use sequence completion as a setting to study the behavior of neural language models due to its generality. For instance, sequence completion encompasses story generation [36], contextual text completion [116], language modeling (for $k = 0$), and dialogue modeling [161] where $\mathbf{x}_{1:k}$ is a dialogue history and a continuation is a next utterance.

Given p_θ and a prefix $\mathbf{x}_{1:k}$, finding the optimal continuation is not tractable, so in practice approximate deterministic or stochastic decoding strategies are used to generate continuations.

Deterministic Decoding Widely used deterministic decoding algorithms are greedy search and beam search. The former can be seen as a special case of the latter. Greedy search selects the highest probability token at each time step: $x_t = \arg\max p_\theta(x_t|x_{<t})$. Beam search maintains a fixed-size set of partially-decoded sequences, called hypotheses. At each time step, beam search forms new hypotheses by appending each token in the vocabulary to each existing hypothesis, scoring the resulting sequences As we describe in Section 4.4, these deterministic decoding strategies, which depend highly on underlying model probabilities, expose issues with conventionally trained neural language models.

Stochastic Decoding An alternative is to sample from a model-dependent distribution at each step, $x_t \sim q(x_t|x_{<t}, p_\theta)$. In order to prevent sampling low probability tokens, a typical approach

is to restrict sampling to a subset of the vocabulary $U \subset \mathcal{V}$ at each step:

$$q(x_t|x_{<t}, p_\theta) = \begin{cases} p_\theta(x_t|x_{<t})/Z & x_t \in U \\ 0 & \text{otherwise}, \end{cases}$$

where $Z = \sum_{x \in U} p_\theta(x|x_{<t})$. The *top-k* sampler restricts sampling to the k most-probable tokens; i.e. U is the size k subset of $\mathcal{V}$ which maximizes $\sum_{x \in U} p_\theta(x|x_{<t})$ [36]. The *nucleus* sampler instead restricts sampling to the smallest set of tokens with total mass above a threshold $p \in [0, 1]$; i.e. U is the smallest subset with $\sum_{x \in U} p_\theta(x|x_{<t}) \geq p$ [52].

4.4 Neural Text Degeneration

In this section we discuss two degenerate properties that frequently occur in conventional neural language models trained with the maximum likelihood objective (Equation (4.1)).

Repetition First, model-generated continuations exhibit **sequence-level** repetition, especially with deterministic decoding. The problem is seen by observing samples in Table 4.1, which shows completions from the state-of-the-art GPT-2 language model [116]. Greedy decoding as well as top-k and nucleus sampling exhibit degenerate repetition (with a certain hyper-parameter setting), although greedy decoding shows the worst degradation. Using a Transformer language model trained with maximum likelihood (Section 4.6), we find that the average percentage of repeated n-grams in model continuations with greedy decoding (43%) far exceeds that of humans (0.5%), computed over prefixes drawn from a validation corpus.

Unlike previous work which only focused on degenerate sequence-level repeats [52], we additionally observe that neural language models exhibit substantially more repetition in **next-token**

Prefix	*Lyrically the song has excerpts of different languages including French , Spanish ,*
GPT2-greedy	German , Italian , Portuguese , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , ...
GPT2-top3	German , Italian , Portuguese , Spanish , Portuguese , Portuguese, Italian and Spanish. It is also available in English, French and Spanish. In addition to its lyrics, the album features a number of original songs, ...
GPT2-nucleus-0.3	German , Italian , Portuguese , Spanish , Portuguese , Italian , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , ...
Prefix	*The first pair of pereiopods is armed with a large , asymm*
GPT2-greedy	etrical, and long-range laser cannon. The second pair is armed with a large , asymmetrical, and long-range laser cannon. The third pair is armed with a large , asymmetrical, and long-range laser cannon. The fourth pair is armed with a large ...
GPT2-top3	etrical, and highly mobile head, and the second pair has a small , asymmetrical, and highly mobile head. The second pair has a large and highly mobile head, and the third pair is armed with a large and highly mobile head. The first pair ...
GPT2-nucleus-0.3	etrical head and a large body. The first pair of pereiopods is armed with a large , asymmetrical head and a large body. The first pair of pereiopods is armed with a large , asymmetrical head and a large body. The first pair of pereiopods is armed ...

Table 4.1: Top: Degenerate repetition in completions from a state-of-the-art large-scale language model [116]. The examples contain single-word repetitions, phrase-level repetitions, and structural repetitions where some tokens within a repeating phrase vary. Recently proposed stochastic samplers (top-k, nucleus) exhibit degeneration based on hyper-parameter settings.

prediction compared to human text:

$$\Pr\left(\hat{x}_{k+1} = \arg\max p_\theta(x|\mathbf{x}_{1:k}) \in \mathbf{x}_{1:k}\right) > \Pr\left(x_{k+1} \in \mathbf{x}_{1:k}\right). \tag{4.2}$$

For instance, the Transformer language model (Section 4.6) predicted next-tokens that appeared in the preceding 128 words 62% of the time, versus 49% in ground-truth text. This is especially concerning since the maximum-likelihood objective focuses on optimizing next-token conditional distributions.

Token Distribution Mismatch Second, both greedy continuations and next-token predictions from conventional neural text generators have different token distributions from human text. As demonstrated by Holtzman et al. [52], such models with greedy or beam search tend to produce high frequency tokens too often and low frequency tokens too rarely, where frequency is defined by the human token distribution. With the Transformer language model (Section 4.6), the set of next-token greedy predictions on a held-out validation set had roughly 40% fewer unique tokens than the ground-truth tokens (11.6k vs. 18.9k), and overproduced frequent tokens as seen in Figure 4.1. Such behavior has been linked to generations being judged as dull by humans because rare words can add engaging specificity [126, 152].

4.5 The Unlikelihood training objective

We now describe *unlikelihood training* for neural language models, then in Section 4.6 demonstrate empirically that our proposal substantially improves neural text degeneration (Section 4.4).

4.5.1 Unlikelihood Training

The key idea behind unlikelihood training is decreasing the model's probability of certain tokens, called *negative candidates*. Given a sequence $(x_1, \dots, x_T)$ and a set of negative candidate tokens $C^t = \{c_1, \dots, c_m\}$, where each $c_j \in \mathcal{V}$, we define the **unlikelihood loss** for step t as:

$$\mathcal{L}^t_{\text{UL}}(p_\theta(\cdot|x_{<t}), C^t) = -\sum_{c \in C^t} \log(1 - p_\theta(c|x_{<t})).$$

The loss decreases as $p_\theta(c|x_{<t})$ decreases. We incorporate the unlikelihood loss into a **token-level unlikelihood objective** which augments each time-step of maximum likelihood training:

$$\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), C^t) = -\alpha \cdot \underbrace{\sum_{c \in C^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p_\theta(x_t|x_{<t})}_{\text{likelihood}}. \tag{4.3}$$

As candidates, we use previous context tokens:

$$C^t_{\text{prev-context}} = \{x_1, \ldots, x_{t-1}\} \setminus \{x_t\}. \tag{4.4}$$

Intuitively, minimizing the unlikelihood loss with this candidate set makes (i) incorrect repeating tokens less likely, as the previous context contains potential repeats, and (ii) frequent tokens less likely, as these tokens appear often in the previous context. These candidates are efficient to compute, without requiring additional supervision.

Gradient analysis Let x_t^* be the true next-token (index $i^* \in \mathcal{V}$) at step t, and let x_{neg} be a negative candidate (index i_{neg}). Let $p = p(x_t|x_{<t}) \in \mathbb{R}^{\mathcal{V}}$ be the output of softmax(a) where $a \in \mathbb{R}^{\mathcal{V}}$. The (negative) token-level loss with a single candidate is,

$$\mathcal{L}_t = \log p(x_t^*|x_{<t}) + \alpha \cdot \log(1 - p(x_{\text{neg}}|x_{<t})),$$

and its gradient with respect to a logit a_i is:

$$\frac{\partial \mathcal{L}}{\partial p_i}\frac{\partial p_i}{\partial a_i} = (\mathbb{I}[i = i^*] - p_i) - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}} \left(\mathbb{I}[i = i_{\text{neg}}] - p_i\right).$$

We consider the gradient when i is the true next-token, a negative-candidate, and any other token.

True Next-Token ($i = i^*$)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_*}\frac{\partial p_*}{\partial a_{i^*}} &= (1 - p_*) - \alpha\frac{p_{\text{neg}}}{1 - p_{\text{neg}}}(0 - p_*)\\ &= 1 - p_*(1 - \alpha\frac{p_{\text{neg}}}{1 - p_{\text{neg}}}).\end{aligned}$$

Negative Candidate ($i = i_{\text{NEG}}$)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_{\text{neg}}}\frac{\partial p_{\text{neg}}}{\partial a_{\text{neg}}} &= (0 - p_{\text{neg}}) - \alpha\frac{p_{\text{neg}}}{1 - p_{\text{neg}}}(1 - p_{\text{neg}})\\ &= -p_{\text{neg}}(1 + \alpha).\end{aligned}$$

Other Token ($i \notin \{i^*, i_{\text{NEG}}\}$)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tilde{p}_i}\frac{\partial \tilde{p}_i}{\partial a_i} &= (0 - \tilde{p}_i) - \alpha\frac{p_{\text{neg}}}{1 - p_{\text{neg}}}(0 - \tilde{p}_i)\\ &= -\tilde{p}_i(1 - \alpha\frac{p_{\text{neg}}}{1 - p_{\text{neg}}}).\end{aligned}$$

Combining the three cases above, we get:

$$\nabla \mathcal{L}_a = x^* - m \odot p, \tag{4.5}$$

where $x^* \in \{0, 1\}^V$ is 1 at index i^* and 0 otherwise, and $m \in \mathbb{R}^V$ is:

$$m_i = \begin{cases}(1 - \alpha\frac{p_{\text{neg}}}{1-p_{\text{neg}}}) & i \neq i_{\text{neg}}\\ (1 + \alpha) & i = i_{\text{neg}}. \quad \square\end{cases} \tag{4.6}$$

In general the objective considers multiple candidates:

$$\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), C^t) = -\alpha \cdot \underbrace{\sum_{c \in C^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p_\theta(x_t|x_{<t})}_{\text{likelihood}}.$$

We regroup the token-level objective to be a weighted sum of per-candidate objectives:

$$-\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), C^t) = \frac{1}{|C^t|} \sum_{c \in C^t} \big(\log p_\theta(x_t|x_{<t}) + \alpha_c \cdot \log(1 - p_\theta(c|x_{<t})) \big)$$

where $\alpha_c = \alpha \cdot |C^t|$.

Now the gradient can be generalized to multiple candidates, in which case the gradient takes the same form as Equation (4.6), but with α_c in place of α.

This unlikelihood gradient (Equation (4.5)) differs from the likelihood gradient, $(x^* - p)$, due to the term m which varies based on the hyper-parameter α and the model's negative candidate probability, p_{neg}. At the ground-truth token index i^*, the unlikelihood gradient is positive, increasing the ground-truth token's probability with a magnitude that grows with p_{neg}. Conversely, at the negative candidate index i_{neg} the gradient is negative. At all other token indices $i \notin \{i^*, i_{\text{neg}}\}$, the gradient moves from negative to positive as p_{neg} increases. For instance, with $\alpha = 1.0$ the gradient increases the probability of each token x_i when the model assigns high probability to the negative candidate ($p_{\text{neg}} > 0.5$).

4.5.2 Sequence-Level Unlikelihood Training

While the token-level unlikelihood objective efficiently augments maximum likelihood training with token-level penalties, it is limited to prefixes drawn from the training distribution. The resulting *distribution mismatch* between training sequences and generated sequences is a known issue with maximum-likelihood training, motivating objectives that operate on model-generated sequences [27, 117, 122, 159].

We thus propose a **sequence-level unlikelihood objective** which uses unlikelihood on decoded continuations. That is, given a prefix $(x_1, \ldots, x_k) \sim p_*$, we decode a continuation $(x_{k+1}, \ldots, x_{k+N}) \sim p_\theta(\cdot|x_1, \ldots, x_k)$, construct per-step negative candidate sets $(C^{k+1}, \ldots, C^{k+N})$, and define each per-step sequence-level loss for $t \in \{k+1, \ldots, k+N\}$ as:

$$\mathcal{L}^t_{\text{ULS}}(p_\theta(\cdot|x_{<t}), C^t) = -\sum_{c \in C^t} \log(1 - p_\theta(c|x_{<t})). \tag{4.7}$$

Intuitively, the negative candidates can identify problematic tokens for the loss to penalize. We choose to penalize repeating n-grams in the continuation:

$$C^t_{\text{repeat-n}} = \{x_t\} \text{ if } (x_{t-i}, \ldots, x_t, \ldots, x_{t+j}) \in x_{<t-i} \text{ for any } (j-i) = n, i \leq n \leq j, \tag{4.8}$$

which says that x_t is the (single) negative candidate for step t if it is part of a repeating n-gram. An alternative we tried is to choose a penalization probability p_{penalize}, and use x_t as the single negative candidate for time t when $z_t \sim \text{Bernoulli}(p_{\text{penalize}})$ is 1, and no negative candidate for time t otherwise; this approach was effective but under-performed the $C_{\text{repeat-n}}$ candidates as we discuss further in Section 4.6.4.

In our experiments we apply this sequence loss in two ways: (i) using it to fine-tune a standard MLE baseline; and (ii) using it to fine-tune an unlikelihood model trained at the token level, $\mathcal{L}_{\text{UL-token}}$. We refer to the former as $\mathcal{L}_{\text{UL-seq}}$ and the latter as $\mathcal{L}_{\text{UL-token+seq}}$. In both cases, fine-tuning is done by equally mixing sequence-level unlikelihood updates (Equation (4.7)) and the token-level loss from which it was initially trained (either likelihood updates (Equation (4.1)) or token-level unlikelihood updates (Equation (4.3))).

Efficiency Any objective that requires explicitly decoding a sequence is constrained by sample efficiency when decoding is slow; if sample efficiency is low, the total decoding time is too large for practical use. In our experiments we show that when used for fine-tuning, the sequence-level unlikelihood objective substantially reduced degeneration in **under 1,500 updates**, rendering it practical for modern large-scale neural models, even with high decoding costs.

4.6 Experiments

We follow a standard language modeling setup from Baevski and Auli [5] and evaluate our method on the task of sequence completion, detailed below.[1]

Model Architecture Recent large-scale language models are based on the Transformer architecture, a multi-layer feed-forward network with self-attention [143]. We use a 16-layer Transformer with 8 attention heads, embedding dimension 1024, and fully-connected dimension 4096; the architecture is based on Baevski and Auli [5] but with standard embedding and softmax layers. Our proposed method is architecture agnostic; we choose this one as a representative of recent large-scale language models, e.g., Radford et al. [116].

Dataset We use the Wikitext-103 dataset [90], a large-scale collection of Wikipedia articles containing over 100 million words and 260 thousand unique tokens. As a document-level dataset, Wikitext-103 is an open-source representative of recent datasets used for large-scale language modeling [5, 116]. We perform experiments at the word level.

Training We train on fixed-length contiguous sequences, in our case of length 1,536, which was selected based on GPU memory constraints. For the token-level losses ($\mathcal{L}_{\text{MLE}}$, $\mathcal{L}_{\text{UL-token}}$), we train each model on 8 GPUs for a maximum of 150k updates, evaluating on the validation set and saving the model state every 10k updates. For the experiments below, we select the saved model state with the best validation perplexity.

[1]Our code is available at `https://github.com/facebookresearch/unlikelihood_training`; implemented with Fairseq [104].

		seq-rep-4
Prefix	*... Lyrically the song has excerpts of different languages including French , Spanish*	
$\mathcal{L}_{\text{MLE}}$	, Italian , Spanish , Italian , Spanish , Italian , Spanish , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese	0.744
$\mathcal{L}_{\text{UL-token+seq}}$	, and German . In the first verse , the protagonist sings about being a " girl who 's been in love with someone else " , while the second verse describes the relationship between the protagonist and her lover . In the third verse , the protagonist sings	0.063
Prefix	*... starboard engines and was going to crash . " We 're going in ,"*	
$\mathcal{L}_{\text{MLE}}$	he said . " We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to	0.787
$\mathcal{L}_{\text{UL-token+seq}}$	Hood said . " I 'm going to make sure we 're going to get back to the water . " The order to abandon ship was given by Admiral Beatty , who ordered the remaining two battlecruisers to turn away . At 18 : 25 , Hood turned his	0.000
Prefix	*... career - high 27 points on 8 - for - 11 shooting with three rebounds*	
$\mathcal{L}_{\text{MLE}}$	and two assists . On January 3 , 2012 , he was named to the 2012 – 13 All - Atlantic 10 first team . On February 3 , 2012 , he was named to the Atlantic 10 first team . On February 5 , 2012 , he was named	0.277
$\mathcal{L}_{\text{UL-token+seq}}$	and a career - high 7 assists against the Minnesota Timberwolves . On February 3 , 2012 , he was named to the 2012 All - NBA First Team . On March 7 , 2012 , he was named one of five finalists for the Naismith Award , which is	0.064

Table 4.2: Example greedy completions showing representative examples of the MLE model's degenerate single-token repetition (top), phrase-level repetition (middle), and 'structural' repetition (bottom), as well as the proposed method's ability to fix these degenerate behaviors.

Sequence-level fine-tuning begins with the model state selected based on the validation perplexity. Models are fine-tuned for 1,500 total updates. With probability 0.5 an update uses $\mathcal{L}_{\text{ULS}}$ and otherwise uses the token-level loss with which the model was trained. For a $\mathcal{L}_{\text{ULS}}$ update, we split each training sequence and greedily decode continuations (details below). The experiments use a prefix length $k = 50$ and continuation length $N = 100$ for fine-tuning.

Completions We evaluate a model on sequence completion by using the model to decode continuations of prefixes derived from the validation (or test) set. Specifically, the validation (or test) set is first partitioned into sequences of 1,536 tokens, as in training. Then we split each sequence into a batch of prefixes of length k (discarding extra tokens), and decode a continuation of length N for each prefix. The experiments below use $k = 50$ and $N = 100$ for evaluation. For deterministic decoding we use greedy search and beam search with beam size 10, and for stochastic decoding we use top-k sampling with $k \in \{3, 50\}$ and nucleus sampling with $p \in \{0.3, 0.9\}$.

4.6.1 Evaluation Metrics

Repetition As a token-level metric for repetition, we use the fraction of next-token (top-1) predictions that occur in the previous ℓ tokens (**rep/ℓ**); given a set $\mathcal{D}$ of length-T sequences,

$$\text{rep}/\ell = \frac{1}{|\mathcal{D}|T} \sum_{\mathbf{x}\in\mathcal{D}} \sum_{t=1}^{T} \mathbb{I}\left[\arg\max p_\theta(x|\mathbf{x}_{<t}) \in \mathbf{x}_{t-\ell-1:t-1}\right]. \tag{4.9}$$

A predicted token is called a "single-token repeat" when $\mathbb{I}[\cdot]$ is 1. Some of these single-token repeats also occur in the human-generated sequences, we thus report a variant which only counts single-token repeats that are additionally not equal to the ground-truth next-token (**wrep/ℓ**).

We use the portion of duplicate n-grams (**seq-rep-n**) in a generated sequence to measure sequence-level repetition. That is, for a continuation $\mathbf{x}_{k+1:k+N}$ we compute,

$$\text{seq-rep-n} = 1.0 - \frac{|\text{unique n-grams}(\mathbf{x}_{k+1:k+N})|}{|\text{n-grams}|}, \tag{4.10}$$

and average over continuations. **seq-rep-n** is zero when the continuation has no repeating n-grams, and increases towards 1.0 as the model repeats. We compute **seq-rep-n** on the continuation.

Token Distribution We quantify a model's predicted token distribution using the number of unique tokens. As a token-level metric (**uniq**), we use the number of unique next-token predictions on a validation or test set $\mathcal{D}$, i.e. $|\{\arg\max p(x_t|x_{<t}) \mid x_{<t} \in \mathcal{D}\}|$. As a sequence-level metric (**uniq-seq**) we use the number of unique tokens in continuations of validation or test prefixes (Section 4.6).

Model	search	seq-rep-4	uniq-seq	ppl	acc	rep	wrep	uniq
$\mathcal{L}_{\text{MLE}}$	greedy	.429	10.6k	**24.59**	**.401**	.619	.346	11.6k
	beam	.495	9.4k					
$\mathcal{L}_{\text{UL-token}}$	greedy	**.274**	**12.6k**	25.62	.396	**.569**	**.305**	**12.5k**
	beam	**.327**	**11.2k**					
$\mathcal{L}_{\text{UL-seq}}$	greedy	.130	12.7k	**24.28**	**.406**	.603	.329	12.4k
	beam	.018	16.8k					
$\mathcal{L}_{\text{UL-token+seq}}$	greedy	**.051**	**14.8k**	25.37	.401	**.551**	**.287**	**13.4k**
	beam	**.013**	**17.6k**					
Human	-	.005	18.9k	-	-	.479	-	18.9k

Table 4.3: Results for token-level objectives (upper) and sequence-level fine-tuning (lower) according to sequence-level (left) and token-level (right) metrics using the validation subset of wikitext-103.

Language Modeling Quality We use perplexity (**ppl**), and next-token prediction accuracy (**acc**), defined as

$$\frac{1}{N}|\{\arg\max p(x_t|x_{<t}) = x_t^* \mid x_{<t} \in \mathcal{D}\}|,$$

with N prefixes $x_{<t}$ and true next tokens x_t^*.

4.6.2 Results

Token-level and sequence-level results on the test and valid sets are in Table 4.4 and Table 4.3, respectively.

Baseline The baseline model trained with maximum likelihood ($\mathcal{L}_{\text{MLE}}$) achieved 25.64 test perplexity, comparable to a current state-of-the-art system [5] (24.92). However, the greedy baseline's seq-level repeats (seq-rep-4 .442) and single-token repeats (rep .627) far exceed those in human text (.006, .487 respectively). The baseline continuations have far fewer unique tokens than human text (uniq-seq 11.8k vs 19.8k), with a high rate of frequent tokens (Figure 4.1).

Model	search	seq-rep-4	uniq-seq	ppl	acc	rep	wrep	uniq
$\mathcal{L}_{\text{MLE}}$	greedy	.442	10.8k	**25.64**	**.395**	.627	.352	11.8k
	beam	.523	9.5k					
$\mathcal{L}_{\text{UL-token}}$	greedy	**.283**	**13.2k**	26.91	.390	**.577**	**.311**	**12.7k**
	beam	**.336**	**11.7k**					
$\mathcal{L}_{\text{UL-seq}}$	greedy	.137	13.1k	**25.42**	**.399**	.609	.335	12.8k
	beam	.019	18.3k					
$\mathcal{L}_{\text{UL-token+seq}}$	greedy	**.058**	**15.4k**	26.72	.395	**.559**	**.293**	**13.8k**
	beam	**.013**	**19.1k**					
Human	-	.006	19.8k	-	-	.487	-	19.8k

Table 4.4: Results for token-level objectives (upper) and sequence-level fine-tuning (lower) according to sequence-level (left) and token-level (right) metrics using the test subset of Wikitext-103.

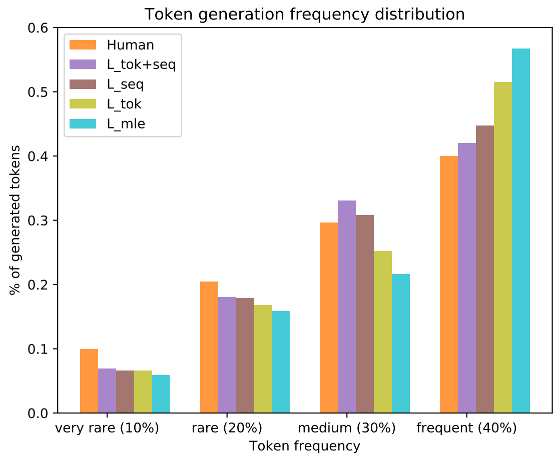
Token-Level Objective The proposed token-level unlikelihood objective ($\mathcal{L}_{\text{UL-token}}$) reduced next-token wrong repetition (wrep .311 vs. .352) while increasing the number of unique next-tokens (uniq 12.7k vs. 11.8k) compared to the baseline ($\mathcal{L}_{\text{MLE}}$). Perplexity and accuracy were similar.

Importantly, the token-level unlikelihood objective yielded substantial improvements in *generations on the sequence-level.* With greedy search, token-level unlikelihood training improved the 4-gram repetition in continuations by 36% (seq-rep-4 .283 vs. .442) while generating roughly 22% more unique tokens than the baseline (uniq-seq 13.2k vs. 10.8k), and a more favorable rate of infrequent tokens (Figure 4.1).

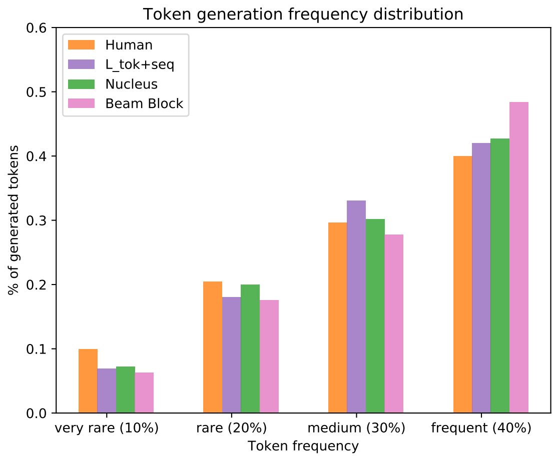
With beam search, unlikelihood training showed similar improvements over the baseline.

Sequence-Level Objective The sequence level fine-tuning ($\mathcal{L}_{\text{UL-token+seq}}$) yielded further improvements, with a **97% reduction** in 4-gram repetitions (seq-rep-4 .013 vs. .442) from the baseline level (greedy $\mathcal{L}_{\text{MLE}}$), and **77% more** unique tokens (uniq-seq 19.1k vs. 10.8k) with beam search.

Compared to the token-level unlikelihood model ($\mathcal{L}_{\text{UL-token}}$) which was the starting point of fine-tuning, the fine-tuned model's repetition substantially improved (seq-rep-4 .058 vs. .283),



(a) Different combinations of unlikelihood



(b) Unlikelihood vs. stochastic decoding

Figure 4.1: Sequence-level token distribution using the test subset of Wikitext-103. Nucleus sampling ($p = 0.9$) and beam blocking ($n = 4$) are used with the maximum likelihood baseline ($\mathcal{L}_{\text{MLE}}$).

unique tokens increased (uniq-seq 15.4k vs. 13.2k), and token-level metrics such as perplexity improved (ppl 26.72 vs. 26.91), despite using *only 1,500 updates*. The token distribution improved, with infrequent tokens produced more often than the baseline, and frequent tokens approaching the human level (Figure 4.1). Finally, after sequence-level fine-tuning, beam search out-performed greedy search.

To visualize how these improvements in metrics translate to generation quality, Table 4.2 shows greedy completions that characterize the baseline's degeneration and $\mathcal{L}_{\text{UL-token+seq}}$'s improved behavior.

GPT-2 Fine-Tuning In the preceding experiment, sequence-level fine-tuning alone ($\mathcal{L}_{\text{UL-seq}}$) showed substantial improvements over the baseline using a small number of updates. This indicates that the proposed sequence-level fine-tuning can be a cheap, effective way to improve existing pre-trained language models. We demonstrate this by fine-tuning a pre-trained GPT-2 [116] language model with sequence-level unlikelihood, using a comparable experimental setup to Section 4.6. We evaluated the GPT-2 medium pre-trained model ('GPT-2') and two separate fine-tuning variants on Wikitext-103. The first variant ('GPT-2$_{\text{MLE}}$') was fine-tuned using maximum likelihood; we select the model state with the lowest validation perplexity. The second model ('GPT-$2_{\text{UL-seq}}$') was fine-tuned using the sequence-level unlikelihood objective (Section 4.5.2). For both evaluation and sequence-level tuning, we used a prefix length of 50 BPE tokens and a continuation length of 100 BPE tokens. In order to train on a single GPU, we used a batch-size of 1024 tokens for MLE updates, and 300 prefix tokens for unlikelihood updates. Due to the smaller batch size and single-GPU setting, we used 10,000 updates during sequence-level fine-tuning, comparable to the 1,500 updates in the main experiment (Section 4.6) in terms of the total number of tokens. Fine-tuning with unlikelihood yielded similar improvements in sequence-level repetition (seq-rep-4 .042 vs. .506) to those observed in Table 4.3, while maintaining language modeling quality according to perplexity and accuracy (see Table 4.5).

Model	**search**	**seq-rep-4**	**ppl**	**acc**	**rep**	**wrep**	**uniq**
GPT-2	greedy	.506	20.75	.430	**.589**	.306	**13.3k**
GPT-2_{MLE}	greedy	.460	**15.82**	**.464**	.612	**.305**	11.8k
GPT-$2_{\text{UL-seq}}$	greedy	**.042**	18.49	.444	.613	.317	11.3k
Human	-	.005	-	-	.407	-	17.7k

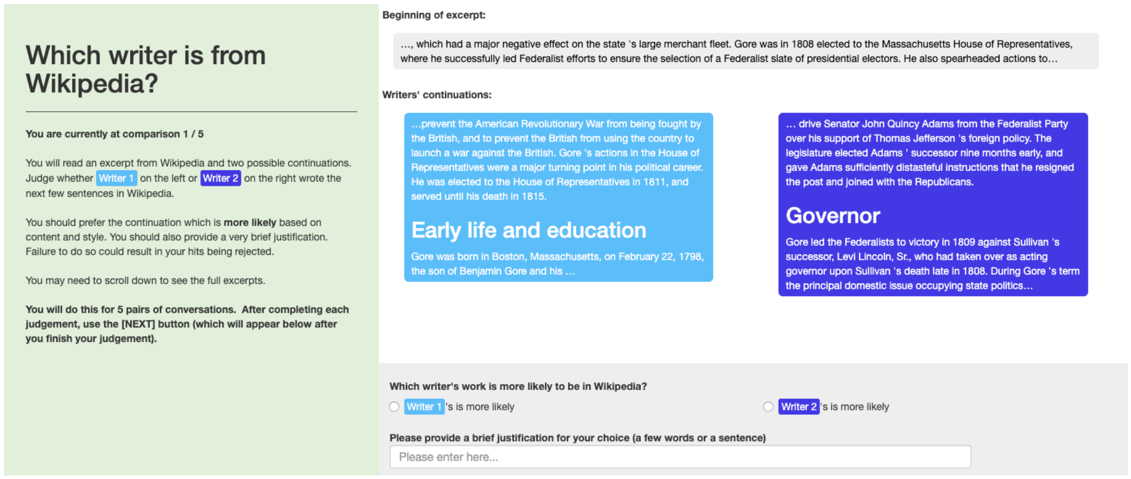
Table 4.5: GPT-2 results according to sequence-level and token-level metrics using the validation subset of wikitext-103. seq-rep-4 is computed on the word level; ppl, acc, rep, wrep are computed on the BPE level.

Stochastic Decoding Although we have focused on deterministic decoding, we also confirm that a model trained with the proposed unlikelihood objectives may still be used with stochastic decoders. Table 4.7 provides automatic metrics for top-k and nucleus sampling (called top-p) on the Wikitext-103 test set. These can be compared with the main results of the chapter in Table 4.4. In general, sampling methods yield worse next-token predictions than deterministic approaches (0.302 vs. 0.394 acc for top-k-50 vs. greedy MLE, where acc for stochastic decoding measures the probability that the decoding strategy chooses the ground truth word given a ground truth context). As the choice of sampling hyperparameter gets closer to greedy (i.e. lower values of

k and p) next token accuracy improves, eventually approaching the greedy MLE results. The unlikelihood-trained sampling models have similar next token accuracy (acc) to their likelihood-trained counterparts, but exhibit fewer repetitions. For lower values of p and k the improvements of unlikelihood training are larger, e.g., 0.277 reduced to 0.0041 for 4-gram sequence repetitions (seq-rep-4) using top-p-0.3. At higher levels of p and k, for all methods the continuations contain more unique tokens than that of humans, meaning those values may be too high.

4.6.3 Human Evaluation

We perform a crowdworker evaluation to judge the quality of the generations of our proposed models compared to each other, the baseline, two other generation methods, and the reference. We employ a pairwise setup: an evaluator is presented with a prefix and shown continuations from two different models and asked to select which continuation they found more natural.



Which writer is from Wikipedia?

You are currently at comparison 1 / 5

You will read an excerpt from Wikipedia and two possible continuations. Judge whether Writer 1 on the left or Writer 2 on the right wrote the next few sentences in Wikipedia.

You should prefer the continuation which is **more likely** based on content and style. You should also provide a very brief justification. Failure to do so could result in your hits being rejected.

You may need to scroll down to see the full excerpts.

You will do this for 5 pairs of conversations. After completing each judgement, use the [NEXT] button (which will appear below after you finish your judgement).

Beginning of excerpt:

..., which had a major negative effect on the state 's large merchant fleet. Gore was in 1808 elected to the Massachusetts House of Representatives, where he successfully led Federalist efforts to ensure the selection of a Federalist slate of presidential electors. He also spearheaded actions to...

Writers' continuations:

...prevent the American Revolutionary War from being fought by the British, and to prevent the British from using the country to launch a war against the British. Gore 's actions in the House of Representatives were a major turning point in his political career. He was elected to the House of Representatives in 1811, and served until his death in 1815.

Early life and education

Gore was born in Boston, Massachusetts, on February 22, 1798, the son of Benjamin Gore and his ...

... drive Senator John Quincy Adams from the Federalist Party over his support of Thomas Jefferson 's foreign policy. The legislature elected Adams ' successor nine months early, and gave Adams sufficiently distasteful instructions that he resigned the post and joined with the Republicans.

Governor

Gore led the Federalists to victory in 1809 against Sullivan 's successor, Levi Lincoln, Sr., who had taken over as acting governor upon Sullivan 's death late in 1808. During Gore 's term the principal domestic issue occupying state politics...

Which writer's work is more likely to be in Wikipedia?

○ Writer 1 's is more likely ○ Writer 2 's is more likely

Please provide a brief justification for your choice (a few words or a sentence)

Please enter here...

Figure 4.2: Screen shot of the user interface used in the human evaluation.

Quality control Following Li, Weston, and Roller [79], we filter workers using quality control questions and limit the number of annotations that they may complete. We require workers to correctly answer two quality control questions for their evaluations to be included. Both quality controls compare the true completion against a greedy baseline model. Following [79], we informed workers that they must provide reasoning for their choices. We filtered workers who did not provide reasoning for at least 80% of their choices. 63% of workers fail at least one of our three quality control mechanisms (2 quality control metrics, and failing to give reasons). 61% fail at least one quality control question; 16% of workers fail both; 4% of workers fail to give reasoning for their choices.

			Crowdworkers		Experts	
Winner		Loser	Win rate	W–L	Win rate	W–L
$\mathcal{L}_{\text{UL-token}}$		$\mathcal{L}_{\text{MLE}}$ baseline	57%	17–13		
$\mathcal{L}_{\text{UL-seq}}$		$\mathcal{L}_{\text{MLE}}$ baseline	*71%	41–17		
$\mathcal{L}_{\text{UL-token+seq}}$	*beats*	$\mathcal{L}_{\text{MLE}}$ baseline	*82%	41–9		
$\mathcal{L}_{\text{UL-token+seq}}$		$\mathcal{L}_{\text{UL-token}}$	*75%	56–19		
$\mathcal{L}_{\text{UL-token+seq}}$		$\mathcal{L}_{\text{UL-seq}}$	59%	38–27		
$\mathcal{L}_{\text{UL-token+seq}}$	*beats*	Nucleus	59%	47–33	*83%	30–6
$\mathcal{L}_{\text{UL-token+seq}}$		Beam blocking	60%	50–34	*74%	25–9
Reference		$\mathcal{L}_{\text{MLE}}$ baseline	*85%	17–3		
Reference		Nucleus	*69%	38–17		
Reference		Beam blocking	*68%	48–23		
Reference	*beats*	$\mathcal{L}_{\text{UL-token}}$	*73%	44–16		
Reference		$\mathcal{L}_{\text{UL-seq}}$	50%	30–30		
Reference		$\mathcal{L}_{\text{UL-token+seq}}$	*64%	46–26		

Table 4.6: Human evaluation results. * denotes statistical significance (2-sided binomial test, $p < .05$). W/L denotes raw number of wins and loses by each comparison.

Human evaluation results Prompts are from the Wikitext-103 test set. All models used beam search (beam size 10) for generation, except for those that use stochastic decoding. We report the win rates for each pairwise comparison. The results are presented in Table 4.6. We find that all proposed models are preferred over the baseline, and that congruent with automatic metrics, win rates improve after adding the sequence level objective. Our best model also outperforms the baseline used with either nucleus sampling or beam blocking.

We also collected limited annotations from other NLP researchers. These *Expert* annotators

Search	Model	seq-rep-4	uniq-seq	ppl	acc	rep	wrep	uniq
top-k-3	$\mathcal{L}_{\text{MLE}}$	.0991	14.7k	25.70	.350	.597	.355	12.6k
	$\mathcal{L}_{\text{UL-token}}$	.0491	16.4k	27.02	.344	.539	.306	13.6k
	$\mathcal{L}_{\text{UL-seq}}$	.0068	17.9k	25.11	.353	.581	.341	13.6k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0087	15.2k	26.84	.347	.524	.292	14.6k
top-k-50	$\mathcal{L}_{\text{MLE}}$	.0165	21.9k	25.70	.302	.511	.303	16.1k
	$\mathcal{L}_{\text{UL-token}}$	.006	23.5k	27.02	.286	.440	.247	17.8k
	$\mathcal{L}_{\text{UL-seq}}$	.0005	25.7k	25.11	.291	.497	.291	17.3k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0009	23.7k	26.84	.289	.430	.238	18.8k
top-p-0.3	$\mathcal{L}_{\text{MLE}}$	.273	13.6k	25.70	.264	.339	.154	12.6k
	$\mathcal{L}_{\text{UL-token}}$	.101	16.5k	27.02	.247	.290	.121	13.9k
	$\mathcal{L}_{\text{UL-seq}}$	.0033	20.8k	25.11	.266	.327	.145	13.6k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0041	19.1k	26.84	.250	.284	.116	14.9k
top-p-0.9	$\mathcal{L}_{\text{MLE}}$	.0154	26.9k	25.70	.288	.462	.263	18.6k
	$\mathcal{L}_{\text{UL-token}}$	.004	30.2k	27.02	.266	.381	.202	22.3k
	$\mathcal{L}_{\text{UL-seq}}$	.0003	34.7k	25.11	.290	.450	.254	19.6k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0007	32.4k	26.84	.269	.376	.198	22.7k
Human	-	.006	19.8k	-	-	.487	-	19.8k

Table 4.7: Stochastic decoding results according to sequence-level (left) and token-level (right) metrics using the test subset of Wikitext-103.

were given the same UI as the crowdworkers, and not told about models they were evaluating, but all annotators were familiar with language models. As shown in Table 4.6, the $\mathcal{L}_{\text{UL-token+seq}}$ model significantly outperforms both nucleus sampling and beam blocking according to the experts.

4.6.4 Sequence-level random candidates

In Section 4.5.2 we described a way to penalize tokens that occurred in a n-gram repetition. One alternative is to penalize a random subset of the generated sequence. That is, given a continuation $x_{t+1}, \ldots, x_{t+K}$, we now define per-step candidates $(C^{k+1}, \ldots, C^{k+N})$ as:

$$C^t_{\text{random-seq}} = \begin{cases} \{x_t\} & \text{if } z_t = 1 \\ \emptyset & \text{if } z_t = 0, \end{cases}$$

for each $t \in \{k+1, \ldots, k+N\}$, where $z_t \sim \text{Bernoulli}(p_{\text{penal}})$, and $p_{\text{penal}} \in [0,1]$ is a fixed hyper-parameter. Intuitively, these candidates identify random tokens in the generated sequence (hence 'random-seq'), which are then penalized by the sequence-level loss (Equation (4.7)).

Results with different values of p_{penal} are shown in Table 4.8. Penalizing 10% of the generated tokens led to substantial improvements in seq-rep-4 for both greedy and beam search compared to the baseline (e.g., 41% for $\mathcal{L}_{\text{UL-seq}}$ greedy, 73% for $\mathcal{L}_{\text{UL-tok+seq}}$ greedy), though using n-gram repetition candidates yielded further improvements (Section 4.5.2, Table 4.3). Improvements in single-token metrics were similar to those from the n-gram repetition candidates (e.g., wrep .287). These results with random-seq candidates demonstrate that sequence fine-tuning can yield improvements without explicitly using the notion of repetition for candidate selection. We also find that penalizing 90% of the generated tokens yields substantial improvements in beam search, but not greedy search; investigating this is left as future work.

4.7 Conclusion

We described *unlikelihood training*, an approach to training neural language models. We observed that state-of-the-art models trained to maximize likelihood exhibit neural text degeneration, which we characterized and quantified in terms of repetition and token distribution mismatch. Our results show that the likelihood objective is not constrained enough, in the sense that two models with the same perplexity can exhibit wildly different generation performance. We empirically showed that unlikelihood training - both at the token and sequence levels - substantially reduced degeneration according to automatic metrics, and outperformed likelihood-trained

Model	p_{penal}	**search**	**seq-rep-4**	**uniq-seq**	**ppl**	**acc**	**rep**	**wrep**	**uniq**
$\mathcal{L}_{\text{MLE}}$	-	greedy	.429	10.6k	24.590	.401	.619	.346	11.6k
	-	beam	.495	9.4k					
$\mathcal{L}_{\text{UL-seq}}$	0.1	greedy	.253	9.9k	24.329	.404	.602	.330	12.3k
		beam	.274	13.1k					
$\mathcal{L}_{\text{UL-seq}}$	0.9	greedy	.434	5.3k	26.519	.399	.600	.330	12.2k
		beam	.231	13.5k					
$\mathcal{L}_{\text{UL-tok+seq}}$	0.1	greedy	.116	12.5k	25.518	.399	.551	.287	13.2k
		beam	.146	14.2k					
$\mathcal{L}_{\text{UL-tok+seq}}$	0.9	greedy	.423	6.7k	26.629	.396	.551	.288	13.2k
		beam	.080	16k					
Human	-	-	.005	18.9k	-	-	.479	-	18.9k

Table 4.8: Results for sequence-level fine-tuning using **random-seq candidates** according to sequence-level (left) and token-level (right) metrics using the **validation subset of wikitext-103**.

models with various decoding methods according to human evaluation, being superior to the current state-of-the-art approaches.

4.8 Developments since chapter release

We have published our work in the proceedings of ICLR 2020 [150]. In addition to this work, we have applied the unlikelihood training framework in the context of dialogue modeling to address a similar kind of degeneracy [81]. In particular, we have shown that unlikelihood training can be used with external negative candidates to reduce inconsistency in the form of contradiction with the context.

Our findings gained attention and further work was built on top of it. Fu et al. [40] have done a theoretical analysis of a repetition problem and propose a novel encoding approach which reduces context-level ambiguity of a next token prediction. Su et al. [139] used unlikelihood training for non-autoregressive text generation, where the repetition degeneracy exhibits itself even more due to conditional independence approximation. Bahuleyan and Asri [7] applied our proposed framework to increase diversity of the keyphrase generation model.

Other works proposed alternative variants of training objectives and decoding algorithms addressing this repetition issue. Lin, Han, and Joty [82] altered cross-entropy gradient with the 'ScaleGrad' modification. They showed that it facilitates the model to use more novel tokens during generation. Jiang et al. [56] proposed a new objective called 'TLDR' which stands for "Token Loss Dynamic Reweighting". It increases the error signal contribution for 'hard to learn' tokens. The level of hardness was approximated via the value of per token objective during training. Similarly to the nucleus sampling from Holtzman et al. [52], Martins, Marinho, and Martins [86] proposed to sample from the model parametrized with the entmax transformation [111]. In that case, the process of truncating the tail of the distribution is not needed, because the model induces a sparse distribution by design.

5 Non-termination, consistency, and incomplete decoding

5.1 Introduction

Neural sequence models trained with maximum likelihood estimation (MLE) have become a standard approach to modeling sequences in a variety of natural language applications such as machine translation [6], dialogue modeling [146], and language modeling [115]. Despite this success, MLE-trained neural sequence models have been shown to exhibit issues such as length bias [136, 138] and degenerate repetition [52]. These issues are suspected to be related to the maximum likelihood objective's local normalization, which results in a discrepancy between the learned model's distribution and the distribution induced by the decoding algorithm used to generate sequences [3, 70]. This has prompted the development of alternative decoding methods [52, 154] and training objectives [97, 150]. In this chapter, we formalize and study this discrepancy between the model and the decoding algorithm.

We begin by formally defining *recurrent neural language models*, a family that encompasses neural models used in practice, such as recurrent neural networks [22, 34, 48], and transformers [143]. Next, we formally define a decoding algorithm – a function that induces a distribution over sequences given a recurrent language model and a context distribution – which is used to obtain probable sequences from a model. In this chapter, we show that the distribution induced by a decoding algorithm can contradict this intended use; instead, the decoding algorithm may return improbable, infinite-length sequences.

Our main finding is that a sequence which receives zero probability under a recurrent language model's distribution can receive nonzero probability under the distribution induced by a decoding algorithm. This occurs when the recurrent language model always ranks the sequence termination token outside of the set of tokens considered at each decoding step, yielding an infinite-length, zero probability sequence. This holds whenever the decoding algorithm is *incomplete*, in the sense that the algorithm excludes tokens from consideration at each step of decoding, which is the case for common methods such as greedy search, beam search, top-k sampling [36], and nucleus sampling [52]. We formalize our main finding using the notion of *consistency* [19] – whether a distribution assigns probability mass only to finite sequences – and prove that a consistent recurrent language model paired with an incomplete decoding algorithm can induce an inconsistent sequence distribution.

Based on the insight that inconsistency occurs due to the behavior of the termination token under incomplete decoding, we develop two methods for addressing inconsistency. First, we propose *consistent sampling methods* which guarantee that the termination token is not excluded from selection during decoding. Second, we introduce a *self-terminating recurrent language model* which ensures that the termination token is eventually ranked above all others, guaranteeing consistency under incomplete decoding.

To empirically measure inconsistency, we decode sequences from trained recurrent language models and measure the proportion of sequences with lengths far exceeding the maximum training sequence length. Our experiments on the Wikitext2 dataset [90] suggest that inconsistency occurs in practice when using incomplete decoding methods, while the proposed consistent sampling methods and self-terminating model parametrization prevent inconsistency and maintain language modeling quality.

The theoretical analysis reveals defects of existing decoding algorithms, providing a way to develop future models, inference procedures, and learning algorithms. We present methods related to sampling and model parametrization, but there are more directions for future investigation; we close with directions related to sequence-level learning.

5.2 Background

We begin our discussion by establishing background definitions. First, we define a sequence which is the main object of our investigation.

Definition 5.2.1 (Sequence)**.** *A sequence Y is an ordered collection of items from a predefined finite vocabulary V. A sequence of finite length always ends with a special token $\langle eos \rangle \in V$ that only appears at the end of a sequence.*

Each model we consider generates a sequence conditioned on context information, such as a prefix in sentence completion. To consider this, we define a context distribution.

Definition 5.2.2 (Context distribution)**.** *A context distribution $p(C)$ is a probability distribution defined over a set $\mathcal{C}$. An element $C \in \mathcal{C}$ is called a context.*

5.2.1 Recurrent Language Models

A recurrent language model is an autoregressive model of a sequence distribution, where each conditional probability is parameterized with a neural network. Importantly, we assume that all tokens in a sequence are dependent on each other under a recurrent language model. This allows us to avoid cases in which the model degenerates to a Markovian language model, such as an n-gram model with a finite n.

Definition 5.2.3 (Recurrent language model)**.** *A recurrent language model p_θ is a neural network* *that computes the following at each time step:*

$$p_\theta(y_t = v \mid y_{<t}, C) = \frac{\exp(u_v^\top h_t + c_v)}{\sum_{v' \in V} \exp(u_{v'}^\top h_t + c_{v'})},$$

where $h_t = f_\theta(y_t, h_{t-1})$ and $h_0 = g_\theta(C)$, and u, c, θ are parameters. A recurrent language model thereby computes the probability of a sequence $Y = (y_1, \ldots, y_T)$ by

$$p_\theta(Y \mid C) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, C),$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$. This distribution satisfies $y_i \not\perp y_j \mid C, \ \forall i < j$.

Practical variants of the recurrent language model differ by the choice of transition function f_θ [22, 34, 48, 143]. The use of softmax [16] implies that every unique token in the vocabulary is considered at every location of a sequence.

Remark 5.2.1. *Under the conditional distribution of a recurrent LM, every token $v \in V$ is assigned a positive probability, implying that $0 < p_\theta(v \mid y_{<t}, C) < 1$. Any finite sequence is probable under a recurrent LM under any context, i.e., $p_\theta(Y \mid C) > 0$ for any sequence Y of finite length.*

5.2.2 Decoding Algorithms

Because it is intractable to decode the most probable sequence in general, it is necessary in practice to use an approximate decoding algorithm.

Definition 5.2.4 (Decoding algorithm)**.** *A decoding algorithm $\mathcal{F}(p_\theta, C)$ is a function that generates a sequence $\tilde{Y}$ given a recurrent language model p_θ and context C. Let $q_{\mathcal{F}}$ denote the distribution induced by the decoding algorithm $\mathcal{F}$.*

We consider two families of decoding algorithms. In our analysis we only consider algorithms that decode in a single pass, forward in time, without modifying previously selected tokens.

STOCHASTIC DECODING. The first family consists of stochastic algorithms. Among them, ancestral sampling is asymptotically unbiased and can be used for finding the most probable sequence, although with high variance.

Definition 5.2.5 (Ancestral sampling). *Ancestral sampling $\mathcal{F}_{anc}$ generates a sequence from a recurrent language model p_θ given context C by recursively sampling from $p_\theta(y_t \mid \tilde{y}_{<t}, C)$ until $\tilde{y}_t = \langle eos \rangle$: $\tilde{y}_t \sim p_\theta(y_t \mid \tilde{y}_{<t}, C)$.*

To avoid the high variance, two approximate stochastic decoding algorithms have recently been proposed and tested with recurrent language models. Top-k sampling considers only a subset of the k most probable tokens from the vocabulary at a time, while nucleus sampling considers only the minimal subset of most probable tokens whose total probability is higher than a predefined threshold.

Definition 5.2.6 (Top-k sampling [36]). *Top-k sampling $\mathcal{F}_{top\text{-}k}$ generates a sequence from a recurrent language model p_θ given context C by recursively sampling from:*

$$q(v) \propto \begin{cases} p_\theta(v \mid y_{<t}, C), & \textit{if } v \in V_k, \\ 0, & \textit{otherwise.} \end{cases}$$

where $V_k = \arg \underset{v'}{\textit{top-k}}\, p_\theta(v' \mid y_{<t}, C)$.

Definition 5.2.7 (Nucleus sampling [52]). *Nucleus sampling $\mathcal{F}_{nuc\text{-}\mu}$ generates a sequence from a recurrent language model p_θ given context C by recursively sampling from the following proposal distribution. Let $v_1, \ldots, v_{|V|}$ denote tokens in V such that $p_\theta(v_i \mid y_{<t}, C) \geq p_\theta(v_j \mid y_{<t}, C)$ for all $i < j$, and define*

$$q(v) \propto \begin{cases} p_\theta(v \mid y_{<t}, C), & \textit{if } v \in V_\mu, \\ 0, & \textit{otherwise,} \end{cases}$$

where $V_\mu = \{v_1, \cdots, v_{k_\mu}\}$ *with*

$$k_\mu = \min \left\{ k \;\middle|\; \sum_{i=1}^{k} p_\theta(v_i \mid y_{<t}, C) > \mu \right\}.$$

Deterministic decoding. The other family consists of deterministic decoding algorithms, where a token is selected deterministically according to a rule at each decoding step. The most naive algorithm, called greedy decoding, simply takes the most probable token at each step.

Definition 5.2.8 (Greedy decoding). *Greedy decoding $\mathcal{F}_{greedy}$ generates a sequence from a recurrent language model p_θ given context C by recursively selecting the most likely token from $p_\theta(y_t|\tilde{y}_{<t}, C)$ until $\tilde{y}_t = \langle eos \rangle$:*

$$\tilde{y}_t = \arg\max_{v \in V} \log p_\theta(y_t = v \mid \tilde{y}_{<t}, C).$$

In contrast to greedy decoding, beam search with width k, $\mathcal{F}_{\text{beam-k}}$, operates on the level of partial sequences or prefixes.

Definition 5.2.9 (Prefix). *A prefix ρ_t is an ordered collection of items from V. The score of a prefix is*

$$s(\rho_t) = \sum_{\tau=1}^{t} \log p_\theta(y_\tau = \rho_t[\tau] \mid \rho_t[<\tau], C),$$

where $\rho_t[\tau]$ is a token at time τ from ρ_t.

Starting from a set of empty prefixes, at each iteration a new prefix set is formed by expanding each prefix, then choosing the k highest scoring expanded prefixes.

Definition 5.2.10 (Beam search). *Beam search with width k, $\mathcal{F}_{beam-k}$, generates a sequence from a recurrent language model p_θ by maintaining a size-k prefix set P_t^{top}. Starting with $P_0^{top} = \varnothing$, at each iteration $t \in \{1, 2, \ldots\}$ beam search forms a new prefix set P_t^{top} by expanding the current set,*

$\mathrm{P}_t = \bigcup_{\rho \in \mathrm{P}_{t-1}^{top}} \{\rho \circ v \,|\, v \in V\}$ *(where $\rho \circ v$ is concatenation), then choosing the k highest scoring elements:* $\mathrm{P}_t^{top} = \underset{\rho \in \mathrm{P}_t}{\arg \textit{top-k}}\, s(\rho)$. *Any $\rho \in \mathrm{P}_t^{top}$ ending with $\langle eos \rangle$ is restricted from being expanded further, and is added to a set S. Beam search ends when S contains k sequences, and returns the highest scoring sequence in S.*

Incompleteness. Other than ancestral sampling, the decoding algorithms above are *incomplete* in that they only consider a strict subset of the full vocabulary V at each time step, aside from the trivial case of $k = |V|$.[1]

Definition 5.2.11 (Incomplete Decoding). *A decoding algorithm $\mathcal{F}$ is incomplete when for each context C and prefix $y_{<t}$, there is a strict subset $V'_t \subsetneq V$ such that*

$$\sum_{v \in V'_t} q_{\mathcal{F}}(y_t = v \,|\, y_{<t}, C) = 1.$$

5.3 Consistency of a Decoding Algorithm

Definition of consistency. A recurrent language model p_θ may assign a positive probability to an infinitely long sequence, in which case we call the model inconsistent. This notion of consistency was raised and analyzed earlier, for instance by Booth and Thompson [14] and Chen et al. [19], in terms of whether the distribution induced by p_θ is concentrated on finite sequences. We extend their definition to account for the context C.

Definition 5.3.1 (Consistency of a recurrent language model). *A recurrent language model is consistent under a context distribution $p(C)$ if $p_\theta(|Y| = \infty) = 0$. Otherwise, the recurrent language model is said to be inconsistent.*

Any sequence decoded from a consistent model for a given context is guaranteed to terminate.

[1]Nucleus sampling is incomplete when for every context C and prefix $y_{<t}$, $\min_{v \in V} p_\theta(v|y_{<t}, C) < 1 - \mu$.

Lemma 5.3.1. *If a recurrent LM p_θ is consistent, $p_\theta(|Y| = \infty \mid C) = 0$ for any probable context C.*

Proof. Suppose there exists a probable context $\tilde{C}$ such that $p_\theta(|Y| = \infty \mid \tilde{C}) > 0$. Then

$$p_\theta(|Y| = \infty) = \mathbb{E}\left[p_\theta(|Y| = \infty \mid C)\right] \geq p(\tilde{C}) p_\theta(|Y| = \infty \mid \tilde{C}) > 0,$$

which contradicts the consistency of the model p_θ. □

Next, we establish a practical condition under which a recurrent language model is consistent.

Lemma 5.3.2. *A recurrent LM p_θ is consistent if $\|h_t\|_p$ is uniformly bounded for some $p \geq 1$.*

Proof. Let $B > 0$ be an upper bound such that $\|h_t\|_p < B$ for all t. Let q be the conjugate of p satisfying $1/p + 1/q = 1$. Then we have from Hölder's inequality, for all $v \in V$ and t,

$$u_v^\top h_t \leq \|u_v^\top h_t\|_1 \leq \|h_t\|_p \|u_v\|_q < Bu^+,$$

where $u^+ = \max_{v \in V} \|u_v\|_q$. Note that

$$\begin{aligned}
\log \sum_{v \in V} e^{u_v^\top h_t + c_v} &\leq \log \left(\max_{v \in V} e^{u_v^\top h_t + c_v} \times |V| \right) \\
&\leq \max_{v \in V} \{u_v^\top h_t + c_v\} + \log |V| \\
&< Bu^+ + c^+ + \log |V|,
\end{aligned}$$

where $c^+ = \max_{v \in V} c_v$. For a given $y_{<t}$ and context C,

$$\begin{aligned}
\log p_\theta(\langle\text{eos}\rangle \mid y_{<t}, C) &= (u_{\langle\text{eos}\rangle}^\top h_t + c_{\langle\text{eos}\rangle}) - \log \sum_{v \in V} e^{u_v^\top h_t + c_v} \\
&> (-Bu^+ + c_{\langle\text{eos}\rangle}) - (Bu^+ + c^+ + \log |V|) > -\infty,
\end{aligned}$$

and it follows that $p_\theta(\langle\text{eos}\rangle \mid y_{<t}, C) > \xi > 0$ for some strictly positive constant ξ. Then

$$\begin{aligned} p_\theta(|Y| = \infty) &= \lim_{t\to\infty} p_\theta(|Y| > t) = \lim_{t\to\infty} \mathbb{E}\left[p_\theta(|Y| > t \mid C)\right] \\ &= \mathbb{E}\left[\lim_{t\to\infty} p_\theta(|Y| > t \mid C)\right] \le \mathbb{E}\left[\lim_{t\to\infty} (1-\xi)^t\right] = 0, \end{aligned}$$

and hence p_θ is consistent. □

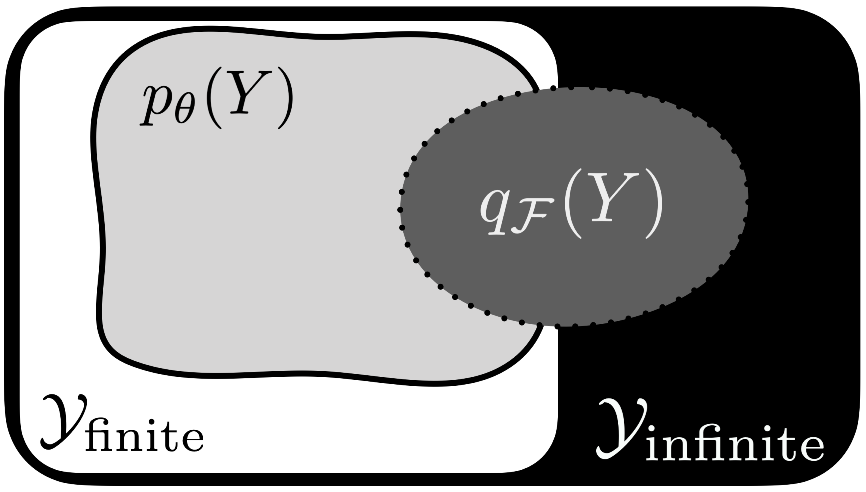
Although this condition is practical because layer normalization or bounded activation functions [22, 34, 143] result in bounded h_t, we show that even if a recurrent language model is consistent, a decoding algorithm may produce an infinite-length sequence. We formalize this discrepancy using the consistency of a decoding algorithm.

Definition 5.3.2 (Consistency of a decoding algorithm)**.** *A decoding algorithm $\mathcal{F}$ is consistent with respect to a consistent recurrent language model p_θ under a context distribution $p(C)$ if the decoding algorithm $\mathcal{F}$ preserves the consistency of the model p_θ, that is, $q_{\mathcal{F}}(|Y| = \infty) = 0$.*

When a consistent recurrent language model p_θ and a decoding algorithm $\mathcal{F}$ induce a consistent distribution $q_{\mathcal{F}}$, we say that p_θ paired with $\mathcal{F}$ is consistent. For instance, any consistent recurrent language model paired with ancestral sampling is consistent, because the induced distribution $q_{\mathcal{F}_{\text{anc}}}$ is the same as the distribution of the original model. We also have an analogue of Lemma 5.3.1.

Lemma 5.3.3. *A consistent decoding algorithm with respect to a consistent recurrent LM decodes only probable sequences. That is, if $q_{\mathcal{F}}(Y \mid C) > 0$, then $p_\theta(Y \mid C) > 0$ for any probable context C.*

Proof. Suppose there exists a decoded sequence $\tilde{Y}$ by $\mathcal{F}$ and probable context $\tilde{C}$ such that $q_{\mathcal{F}}(\tilde{Y} \mid \tilde{C}) > 0$ but $p_\theta(\tilde{Y} \mid \tilde{C}) = 0$. By Remark 2.1, the sequence $\tilde{Y}$ is of infinite length and thus $q_{\mathcal{F}}(|Y| = \infty \mid \tilde{C}) \ge q_{\mathcal{F}}(\tilde{Y} \mid \tilde{C}) > 0$, which contradicts the consistency of $q_{\mathcal{F}}$ by Lemma 3.1. □



$p_\theta(Y)$
$q_{\mathcal{F}}(Y)$
$\mathcal{Y}_{\text{finite}}$
$\mathcal{Y}_{\text{infinite}}$

Figure 5.1: A depiction of the model's sequence distribution (light grey, solid border) and the decoder's induced sequence distribution (dark grey, dotted border). The white and black rectangles depict the set of all finite and infinite sequences, respectively. We prove that under practical conditions, any incomplete decoding algorithm may be inconsistent with respect to a consistent model, as depicted.

Inconsistency of incomplete decoding. Any incomplete decoding algorithm (Definition 5.2.11) can be inconsistent regardless of the context distribution, because there is a recurrent LM that places ⟨eos⟩ outside of V_t' at every step of decoding. To show this, we construct a consistent recurrent language model whose distribution induced by an incomplete decoding algorithm is inconsistent.

Theorem 5.3.1 (Inconsistency of an incomplete decoding algorithm)**.** *There exists a consistent recurrent LM p_θ from which an incomplete decoding algorithm $\mathcal{F}$, that considers only up to $(|V|-1)$-most likely tokens according to $p_\theta(y_t \mid y_{<t}, C)$ at each step t, finds an infinite-length sequence $\tilde{Y}$ with probability 1, i.e., $q_{\mathcal{F}}(|Y| = \infty) = 1$.*

Proof. We prove this theorem by constructing a tanh recurrent network. We define the recurrent function f_θ as

$$h_t = f_\theta(y_t, h_{t-1}) = \tanh\left(\left[\begin{array}{c|c} W_h & \mathbf{0} \\ \hline \mathbf{0} & I \end{array}\right] h_{t-1} + \left[\begin{array}{c} \mathbf{0} \\ \hline e(y_t) \end{array}\right]\right),$$

where $e(y_t) \in \mathbb{R}^{|V|}$ is a one-hot representation of y_t, $W_h \in \mathbb{R}^{d \times d}$ where every entry is positive, and I is an identity matrix of size $|V| \times |V|$. $h_0 = g_\theta(C)$ is constructed to consist of positive values only. Because each element of $|h_t|$ is bounded by 1, the constructed recurrent language model p_θ is consistent by Lemma 5.3.2.

We set u_v (see Definition 5.2.3) to

$$u_v = \left[\begin{array}{c} \bar{u}_v \\ \hline e(v) \end{array}\right], \quad u_{\langle \text{eos} \rangle} = \left[\begin{array}{c} \bar{u}_{\langle \text{eos} \rangle} \\ \hline e(\langle \text{eos} \rangle) \end{array}\right],$$

where $v \neq \langle \text{eos} \rangle$, all elements of $\bar{u}_v$ are positive, all elements of $\bar{u}_{\langle \text{eos} \rangle}$ are negative, and $e(v)$ is a one-hot representation of v. c_v is set to zero.

This defines a valid recurrent language model (Definition 5.2.3), since the conditional distribution at each time t is influenced by all the previous tokens. More specifically, the logit of a token v depends on $\sum_{t'=1}^{t} \mathbb{1}(y_{t'} = v)$, where $\mathbb{1}$ is an indicator function.

This recurrent language model always outputs positive logits for non-$\langle \text{eos} \rangle$ tokens, and outputs negative logits for the $\langle \text{eos} \rangle$ token. This implies $p(\langle \text{eos} \rangle \,|\, y_{<t}, C) < p(v \,|\, y_{<t}, C)$ for all $v \in V \backslash \{\langle \text{eos} \rangle\}$. This means that $\langle \text{eos} \rangle$ is always ranked last at each time step, so an incomplete decoding algorithm that considers at most $(|V| - 1)$ most probable tokens at each time step from $p_\theta(y_t \,|\, y_{<t}, C)$ cannot decode $\langle \text{eos} \rangle$ and thus always decodes an infinitely long sequence $\hat{Y}$, i.e., $q_{\mathcal{F}}(|Y| = \infty \,|\, C) = 1$ for any context C. It yields $q_{\mathcal{F}}(|Y| = \infty) = 1$, while $p_\theta(|Y| = \infty) = 0$ due to consistency of the model p_θ. □

Greedy decoding, beam search, top-k sampling, and nucleus sampling are all inconsistent according to this theorem.

5.4 Fixing the inconsistency

In this section, we consider two ways to prevent inconsistency arising from incomplete decoding algorithms. First, we introduce consistent versions of top-k and nucleus sampling. Second, we introduce the *self-terminating recurrent language model*, which is consistent when paired with any of the decoding algorithms considered in this chapter.

5.4.1 Consistent Sampling Algorithms

The proof of Theorem 5.3.1 suggests that the inconsistency of incomplete decoding algorithms arises from the fact that $\langle \text{eos} \rangle$ may be excluded indefinitely from the set of top-ranked tokens. We propose a simple modification to top-k and nucleus sampling that forces $\langle \text{eos} \rangle$ to be included at each step of decoding. First, we give a condition for when a particular model p_θ paired with a decoding algorithm $\mathcal{F}$ is consistent.

Theorem 5.4.1. *Suppose a recurrent LM p_θ has uniformly bounded $\|h_t\|_p$ for some $p \geq 1$. If a decoding algorithm $\mathcal{F}$ satisfies $q_{\mathcal{F}}(\langle eos \rangle \mid y_{<t}, C) \geq p_\theta(\langle eos \rangle \mid y_{<t}, C)$ for every prefix $y_{<t}$ and context C, then the decoding algorithm $\mathcal{F}$ is consistent with respect to the model p_θ.*

Proof. By Lemma 5.3.2 the model p_θ is consistent and $p_\theta(\langle \text{eos} \rangle \mid y_{<t}, C) > \xi$ for some positive value ξ. Thus, $q_{\mathcal{F}}(\langle \text{eos} \rangle \mid y_{<t}, C) \geq p_\theta(\langle \text{eos} \rangle \mid y_{<t}, C) > \xi$. For $t \geq 1$,

$$q_{\mathcal{F}}(|Y| > t \mid C) = q_{\mathcal{F}}(y_1 \neq \langle \text{eos} \rangle, \cdots, y_t \neq \langle \text{eos} \rangle \mid C) \leq (1 - \xi)^t.$$

Taking the limit $t \to \infty$ and expectation over C, we have

$$q_{\mathcal{F}}(|Y| = \infty) = \mathbb{E}_C \left[\lim_{t \to \infty} q_{\mathcal{F}}(|Y| > t \mid C) \right] \leq \lim_{t \to \infty} (1 - \xi)^t = 0,$$

from which the decoding algorithm is consistent. □

We define consistent variants of top-k and nucleus sampling which satisfy this condition.

Definition 5.4.1 (Consistent top-k sampling)**.** *Consistent top-k sampling is top-k sampling with the following modified proposal distribution:*

$$q(v) \propto \begin{cases} p_\theta(v|y_{<t}, C), & \text{if } v \in V', \\ 0, & \text{otherwise}, \end{cases}$$

where $V' = \{\langle eos \rangle\} \cup \arg \underset{v'}{top\text{-}k}\, p_\theta(v' \mid y_{<t}, C)$.

Definition 5.4.2 (Consistent nucleus sampling)**.** *Consistent nucleus sampling is nucleus sampling with the following modified proposal distribution:*

$$q(v) \propto \begin{cases} p_\theta(v \mid y_{<t}, C), & \text{if } v \in V_\mu \cup \{\langle eos \rangle\}, \\ 0, & \text{otherwise}. \end{cases}$$

The induced probability of ⟨eos⟩ under these two algorithms is always equal to or larger than the model's probability. By Theorem 5.4.1, these algorithms are consistent with respect to any consistent recurrent language model.

5.4.2 Self-Terminating Recurrent LM

Although these consistent sampling algorithms can be used with any recurrent language model, their stochastic nature may not be suitable for finding a single, highly probable sequence. To avoid this limitation, we propose the *self-terminating recurrent language model* (STRLM).

Definition 5.4.3 (Self-terminating recurrent language model)**.** *A self-terminating recurrent lan-*

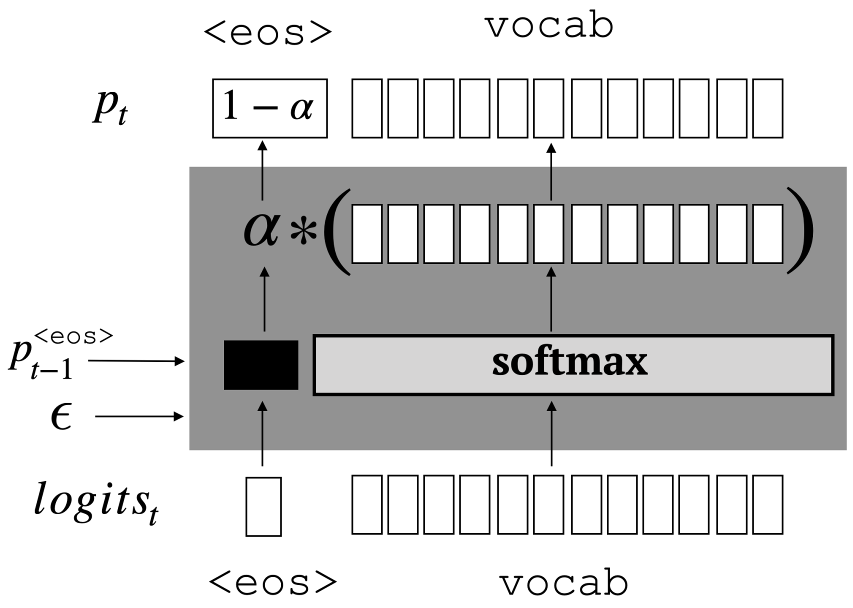


Figure 5.2: The self-terminating recurrent LM uses the layer shown in grey instead of the standard softmax layer. The layer takes logits ($u_{\cdot}^{\top} h_t$), the previous step's ⟨eos⟩ probability ($p_{t-1}^{\langle\text{eos}\rangle}$), and a hyper-parameter $\epsilon \in (0,1)$. The layer computes α using Definition 5.4.3, which determines the ⟨eos⟩ probability ($p_t^{\langle\text{eos}\rangle} \in (\epsilon, 1)$), and guarantees that $p_t^{\langle\text{eos}\rangle} > p_{t-1}^{\langle\text{eos}\rangle}$. The remaining probability mass is allocated to the non-⟨eos⟩ tokens.

guage model computes the following conditional probability at each time step:

$$p_\theta(v \mid y_{<t}, C) = \begin{cases} 1 - \alpha(h_t), & v = \langle eos \rangle\,, \\ \frac{\alpha(h_t)\exp(u_v^\top h_t + c_v)}{\sum_{v' \in V'} \exp(u_{v'}^\top h_t + c_{v'})}, & \end{cases}$$

$$\alpha(h_0) = \sigma(u_{\langle eos \rangle}^\top h_0),$$

$$\alpha(h_t) = \sigma(u_{\langle eos \rangle}^\top h_t)\,[1 - p_\theta(\langle eos \rangle\,|y_{<t-1}, C)]\,,$$

with $\sigma : \mathbb{R} \longrightarrow [0, 1-\varepsilon]$ and $\varepsilon \in (0,1)$. h_t is computed as in the original recurrent LM.

The underlying idea is that the probability of ⟨eos⟩ increases monotonically, since

$$p_t^{\langle \text{eos} \rangle} = 1 - \prod_{t'=0}^{t} \sigma(u_{\langle \text{eos} \rangle}^\top h_{t'}).$$

Consequently, the STRLM is consistent when paired with greedy decoding or beam search as we show in the following theorems.

Theorem 5.4.2. *Greedy decoding is consistent with respect to any self-terminating recurrent LM.*

Proof. Let $p_t^{\langle \text{eos} \rangle}$ denote $p_\theta(\langle \text{eos} \rangle \mid y_{<t}, C)$ and $a_t^{\langle \text{eos} \rangle}$ denote $u_{\langle \text{eos} \rangle}^\top h_t + c_{\langle \text{eos} \rangle}$. By Definition 5.4.3 we have

$$\begin{aligned} p_t^{\langle \text{eos} \rangle} &= 1 - \sigma(a_t^{\langle \text{eos} \rangle})(1 - p_{t-1}^{\langle \text{eos} \rangle}) \\ &= 1 - \prod_{t'=0}^{t} \sigma(a_{t'}^{\langle \text{eos} \rangle}) \geq 1 - (1-\epsilon)^{t+1}. \end{aligned}$$

Take $B = -\log 2 / \log(1-\epsilon)$. We then have $p_t^{\langle \text{eos} \rangle} > 1/2$ for all $t > B$, which implies that ⟨eos⟩ is always the most probable token after time step B. Hence, the sequence length is less than B with probability 1. □

Theorem 5.4.3. *Beam search with width k, $\mathcal{F}_{beam-k}$, is consistent with respect to any STRLM.*

Proof. Let $S(\rho)$ be the size-k set of sequences kept by $\mathcal{F}_{\text{beam}-k}$ that start with a prefix ρ.

Take $B = -\log 2 / \log(1-\epsilon)$ as in the proof of Theorem 4.2. Suppose that there exists at least one prefix $\hat{\rho} \in P_B^{\text{top}}$ which does not end with $\langle\text{eos}\rangle$.

We first want to show that $\hat{\rho}$ induces at most k more steps in beam search with width k, that is, $Y \in S(\hat{\rho})$ implies $|Y| \leq B + k$.

We know from the proof of Theorem 4.2 that an STRLM p_θ satisfies: for any context C and $v \in V \setminus \{\langle\text{eos}\rangle\}$,

$$p_\theta(\langle\text{eos}\rangle \mid \hat{\rho}, C) > p_\theta(v \mid \hat{\rho}, C).$$

For any subsequence $y = (y_1, \ldots, y_l)$ with $y_1 \neq \langle\text{eos}\rangle$,

$$\begin{aligned} p_\theta(\hat{\rho} \circ y \mid \hat{\rho}, C) &= \prod_{i=1}^{l} p_\theta(y_i \mid \hat{\rho} \circ y_{<i}, C) \\ &\leq p_\theta(y_1 \mid \hat{\rho}, C) \\ &< p_\theta(\langle\text{eos}\rangle \mid \hat{\rho}, C). \end{aligned}$$

Thus, $\hat{\rho} \circ \langle\text{eos}\rangle$ is the most probable sequence among sequences starting with the prefix $\hat{\rho}$, and it follows that $\hat{\rho} \circ \langle\text{eos}\rangle \in S(\hat{\rho})$.

Thus, in $S(\hat{\rho})$, there are $(k-1)$ sequences starting with $\hat{\rho} \circ v$ for $v \in V \setminus \{\langle\text{eos}\rangle\}$. By the same argument, at each step at least one sequence ending with $\langle\text{eos}\rangle$ is added to $S(\hat{\rho})$, and therefore at time step $(B+k)$, k sequences ending with $\langle\text{eos}\rangle$ are in $S(\hat{\rho})$.

Note that the result set S by $\mathcal{F}_{\text{beam}-k}$ (Definition 2.11) satisfies

$$S \subseteq \bigcup_{\rho \in P_B^{\text{top}}} S(\rho).$$

Since each $\rho \in P_B^{\text{top}}$ induces sequences of length at most $B + k$, we have

$$p_\theta(|Y| > B + k \mid C) = 0.$$

Taking the expectation over C yields the consistency of the model p_θ. □

5.5 Empirical Validation

The theoretical results rely on the existence of a model that results in inconsistency; it remains to be shown that inconsistency with respect to incomplete decoding occurs with recurrent language models encountered in practice. Moreover, while the proposed methods carry theoretical guarantees in terms of consistency, we must check whether they retain language modeling quality. To do so, we perform experiments using a sequence completion task. In each experiment, we use the beginning of a sequence as context, then decode continuations from a trained recurrent LM and measure the proportion of non-terminated sequences in order to approximately measure inconsistency. The first experiment (Section 5.5.1) shows that inconsistency occurs in practice, and the second experiment (Section 5.5.2) shows the effectiveness of the proposed approaches. Our third experiment (Section 5.5.3) shows that inconsistency also occurs frequently in GPT-2, a large-scale transformer language model.[2]

Sequence completion. We evaluate recurrent language models on a sequence completion task, which has previously been used to evaluate the effectiveness of sequence models, e.g., Graves [44], Holtzman et al. [52], Radford et al. [115], Sutskever, Martens, and Hinton [141], and Welleck et al. [150]. Sequence completion is a general setting for studying the behavior of language models, encompassing machine translation [6], story generation [36], and dialogue modeling [146]. The task consists of decoding a continuation $\hat{Y} \sim \mathcal{F}(p_\theta, C)$ given a length-k prefix $C = (c_1, \ldots, c_k)$,

[2]Code available at `https://github.com/uralik/consistency-lm`.

resulting in a completion $(c_1, \ldots, c_k, \hat{y}_1 \ldots, \hat{y}_T)$.

Dataset. Our first two experiments use Wikitext2 [90], which consists of paragraphs from English Wikipedia, since it has frequently been used to evaluate language models [43, 88, 89]. We consider both word and BPE[3] tokenization. We split each paragraph into sentences using Spacy[4]. We split each sequence, using the first k tokens as a context and the remaining tokens as a continuation. To ensure that each sequence contains a prefix, we prepend padding tokens to make it length k. Special $\langle\text{bos}\rangle$ and $\langle\text{eos}\rangle$ tokens are inserted at the beginning and end of each sequence. We use $k = 10$. Table 5.5 contains dataset statistics.

Context distribution. We define empirical context distributions with prefixes from the train, valid, and test sets: $p(C; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{n=1}^{|\mathcal{D}|} \mathbb{1}(C = C^{(n)})$, where $\mathcal{D} = \{(C^{(n)}, Y^{(n)})\}_{n=1}^{N}$ is a dataset split.

Evaluation metrics. We use finite sequences to approximately measure the consistency of a model paired with a decoding algorithm, since decoding an infinite-length sequence is impossible. We use the proportion of decoded continuations that are longer than a predefined limit,

$$r_L = \frac{1}{|\mathcal{D}|} \sum_{n=1}^{|\mathcal{D}|} \mathbb{1}(|\hat{Y}^{(n)}| \geq L),$$

where $\hat{Y}^{(n)} \sim \mathcal{F}(p_\theta, C^{(n)})$ for each context $C^{(n)}$ in $\mathcal{D}$. We call r_L the *non-termination ratio* of the decoding algorithm $\mathcal{F}$ for an underlying model and context distribution. A value of r_L greater than zero means that some sequences did not terminate within L steps. When L is infinity, this implies that the model paired with the decoding algorithm is inconsistent. In practice, we use a finite L that is substantially larger than the maximum training sequence length, and we interpret a non-zero r_L as evidence that the model paired with the decoding algorithm is inconsistent. We use $L = 1500$, more than 10 times the max training sequence length.

[3]github.com/huggingface/tokenizers
[4]https://spacy.io/

In each experiment, we report the mean and standard deviation of metrics across 10 independent initializations. Unless specified otherwise, we report metrics using the test context distribution, since the train, valid, and randomly generated context distributions had similar results.

Training. We train recurrent language models for sequence completion with maximum likelihood, using the loss $\mathcal{L}(p_\theta, Y) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid y_{<t}, c_1, \ldots, c_k)$, where $Y = (c_1, \ldots, c_k, y_1, \ldots, y_T)$. This amounts to running the full training sequence through a recurrent model and zeroing the loss for the first k tokens, so that the first k steps correspond to learning a g_θ that encodes the context.

model	**context**	**perplexity**
tanh-RNN	train	61.20 ± 1.2
tanh-RNN	test	186.44 ± 1.4
LSTM-RNN	train	72.72 ± 2.4
LSTM-RNN	test	178.39 ± 1.2

Table 5.1: Perplexities of trained recurrent language models (BPE tokenization).

model	**context**	**perplexity**
tanh-RNN	train	91.54 ± 7.9
tanh-RNN	test	136.57 ± 1.8
LSTM-RNN	train	45.80 ± 2.5
LSTM-RNN	test	91.86 ± 0.4

Table 5.2: Perplexities of trained recurrent language models (word tokenization).

Models. We consider recurrent neural networks with hyperbolic tangent activations [tanh-RNN; 34] and LSTM units [LSTM-RNN; 48]. We perform an initial hyper-parameter sweep (see Tables 5.3 and 5.4 for hyper-parameter ranges) and select the best set of hyper-parameters for each of tanh-RNN and LSTM-RNN based on the validation perplexities. With this best set of hyperparameters, we train each of these models with 10 different initializations. The choice of

Parameter	**Values**
Hidden Size	{*256*, **512**, 1024}
Dropout	{0.1, *0.3*, **0.5**}
Embedding Weight Tying	{***True***, False}

Table 5.3: Grid search specification. The values selected for the **LSTM-RNN** and tanh-*RNN* models are shown in bold and italics, respectively (word tokenization).

Parameter	**Values**
Hidden Size	{***256***, 512, 1024}
Dropout	{0.1, **0.3**, *0.5*}
Embedding Weight Tying	{True, ***False***}

Table 5.4: Grid search specification. The values selected for the **LSTM-RNN** and tanh-*RNN* models are shown in bold and italics, respectively (BPE tokenization).

tanh and LSTM RNNs implies that all of the recurrent language models that we train are consistent according to Lemma 5.3.2. Perplexities with models trained with word and BPE tokenization are presented in Tables 5.1 and 5.2. Our LSTM models achieve similar test perplexity (91.86 ± 0.4, word tokenization) to those reported in previous work [89].

Additionally, we train self-terminating tanh-RNN and LSTM-RNN variants (Definition 5.4.3) at various values of ε, which controls a lower bound on the termination probability at each step. We use $\sigma(x) = (1 - \varepsilon) \cdot \text{sigmoid}(x)$. We use the hyper-parameters selected in the preceding grid search. Below, we consider BPE tokenization; similar conclusions held for word tokenization.

5.5.1 Inconsistency of Recurrent LMs

In this experiment, we demonstrate evidence of inconsistency with incomplete decoding methods. Table 5.6 shows non-termination ratios for the recurrent language models using the decoding algorithms considered in this work. Decoding with ancestral sampling always resulted in sequences that terminated within L steps, since the induced distribution is the same as that of the consistent model. On the other hand, the non-zero non-termination ratios for the incomplete

Type	# Train	# Valid	# Test	$\|V\|$	Avg. len
Word	78274	8464	9708	33182	24
BPE	83344	8721	10156	19483	28

Table 5.5: Wikitext2 statistics.

	tanh-**RNN**	**LSTM-RNN**
ancestral	0.00 ± 0.0	0.00 ± 0.0
greedy	12.35 ± 5.18	1.53 ± 1.41
beam-2	1.38 ± 0.95	0.07 ± 0.06
beam-4	0.25 ± 0.19	0.00 ± 0.01
topk-2	0.01 ± 0.01	0.01 ± 0.01
topk-4	0.00 ± 0.0	0.00 ± 0.01
nucleus-0.2	0.06 ± 0.02	0.13 ± 0.15
nucleus-0.4	0.04 ± 0.02	0.02 ± 0.01
consistent topk-2	0.00 ± 0.0	0.00 ± 0.01
consistent topk-4	0.00 ± 0.0	0.00 ± 0.0
consistent nucleus-0.2	0.04 ± 0.02	0.01 ± 0.01
consistent nucleus-0.4	0.02 ± 0.02	0.01 ± 0.01

Table 5.6: Non-termination ratio (r_L (%)) of decoded sequences using ancestral sampling, incomplete, and consistent decoding methods.

decoding algorithms suggest inconsistency with respect to each algorithm, providing evidence for Theorem 5.3.1.

Using greedy decoding, roughly 12% of all contexts resulted in a non-terminating continuation with the tanh-RNN, and roughly 1% with the LSTM-RNN. Nucleus sampling also produced non-terminating sequences with the tanh-RNN (0.06%, nuc-0.2) and LSTM-RNN (0.13%, nuc-0.2). Top-k sampling yielded a small number of non-terminating samples. In general, non-termination approaches zero as k and μ increase, since $\langle\text{eos}\rangle$ has a lower chance of being excluded.

Beam search produced non-terminating sequences with both the tanh-RNN and LSTM-RNN models. This means that $\langle\text{eos}\rangle$ was outside of the top tokens (determined by the beam width) considered at each step, since in our experiments we terminated the beam search when a single beam prefix contained ⟨eos⟩. Larger beam widths reduce non-termination, similar to increasing k or μ.

Prefix	*One Direction delivered a performance of " Kiss You*
nucleus	" , and the album 's second album , " The X @-@ Files " , " The A. " , " The Preder " , " We 've Have You " , " I 've You Wanna Stay " , " The Dream " , " The Bide " , " My Achievement ", " The B. B. " , " A Life " . . .
c-nucleus	" , and " My Boo " was released on September 29 , 2010 . ⟨eos⟩
Prefix	*Boulter starred in two films in 2008 ,*
nucleus	and the band 's music , and " The Rise of Monkey " , " The One With the Way " , " The " Always " , " " Always Your " , " The Wift " , " The Baste " , " The Special With " , " The Way " , " The Special With You " . . .
c-nucleus	and the latter was released in the United States . ⟨eos⟩
Prefix	*This period of unhappiness was the making of*
Baseline	the " most important " of the " mad " , and the " " most important " of the " " " , " the most important " , and the " devil " , " The " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " . . .
STRLM	the first commandment of the poem . ⟨eos⟩
Prefix	*Du Fu 's mother died shortly after he was*
Baseline	a member of the Order of the Order of the Order of the Order of the Order of the Order of the Order of the Order of the Order of the Republic of the Republic of the Republic of the Republic of the Republic of . . .
STRLM	a member of the Order of the British Empire . ⟨eos⟩

Table 5.7: Continuations with consistent nucleus sampling ($\mu = 0.2$) and self-terminating LSTM ($\epsilon = 10^{-3}$).

5.5.2 Consistency of the Proposed Methods

Consistent sampling. Table 5.6 shows that consistent nucleus and top-k sampling (Section 5.4.1) resulted in only terminating sequences, except for a few cases that we attribute to the finite limit L used to measure the non-termination ratio. Consistent nucleus paired with tanh-RNN did not reduce r_L as much as when it was paired with LSTM-RNN. Example continuations are shown in Table 5.7. On prefixes that led to non-termination with the baseline method, the quality tends to improve with the consistent variant since the continuation now terminates. Note that since the model's non-⟨eos⟩ token probabilities at each step are only modified by a multiplicative constant, the sampling process can still enter a repetitive cycle (e.g., when the constant is close to 1), though

	ST	ϵ	r_L (%)	perplexity
tanh-RNN	✓	10^{-2}	00.00 ± 0.0	229.09 ± 9.2
	✓	10^{-3}	00.00 ± 0.0	191.63 ± 1.4
	✓	10^{-4}	00.02 ± 0.02	188.36 ± 2.2
	✗	–	12.35 ± 5.2	186.44 ± 1.4
LSTM	✓	10^{-2}	0.00 ± 0.0	219.71 ± 9.2
	✓	10^{-3}	0.00 ± 0.0	186.04 ± 1.6
	✓	10^{-4}	0.18 ± 0.35	183.57 ± 2.3
	✗	–	1.48 ± 1.43	178.19 ± 1.3

Table 5.8: Non-termination ratio (r_L (%)) of greedy-decoded sequences and test perplexity for STRLMs.

it is guaranteed to terminate.

Self-terminating RLM. As seen in Table 5.8, the self-terminating recurrent language models are consistent with respect to greedy decoding, at the expense of perplexity compared to the vanilla model. The value of ε from Definition 5.4.3, which controls a lower-bound on termination probability at each step, influences both r_L and perplexity. When ε is too large ($\varepsilon = 10^{-2}$), perplexity degrades. When ε is too small ($\varepsilon = 10^{-4}$), the lower-bound grows slowly, so $\langle\text{eos}\rangle$ is not guaranteed to be top-ranked within L steps, resulting in a positive r_L. An ε of 10^{-3} balanced consistency and language modeling quality, with a zero non-termination ratio and perplexity within 8 points of the baseline.

As shown in Figure 5.3, the self-terminating model matches the data length distribution better than the baseline. Example decoded sequences are shown in Table 5.7. For prefixes that led to non-termination with the baseline, the self-terminating models yields finite sequences with reasonable quality. The examples suggest that some cases of degenerate repetition [52, 150] are attributed to inconsistency.

	ST	ϵ	r_L (%)	**perplexity**
tanh-RNN	✓	10^{-2}	0.00 ± 0.0	150.07 ± 2.7
	✓	10^{-3}	0.00 ± 0.0	138.01 ± 0.6
	✓	10^{-4}	1.04 ± 0.6	138.67 ± 1.8
	✗	–	6.07 ± 5.6	136.57 ± 1.8
LSTM	✓	10^{-2}	0.00 ± 0.0	101.24 ± 0.3
	✓	10^{-3}	0.00 ± 0.0	94.33 ± 0.6
	✓	10^{-4}	0.94 ± 0.5	94.15 ± 0.8
	✗	–	1.03 ± 0.3	91.86 ± 0.4

Table 5.9: Non-termination ratio (r_L (%)) of greedy-decoded sequences and test perplexity for self-terminating recurrent models (word tokenization).

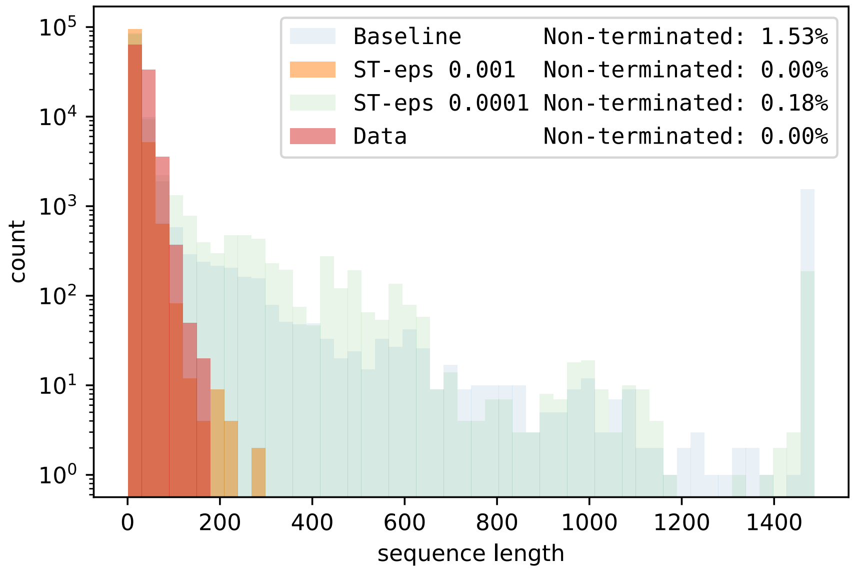


Figure 5.3: Lengths of generated sequences using greedy decoding from vanilla and self-terminating LSTMs.

5.5.3 Inconsistency of GPT-2

We perform a final experiment with GPT-2 117M, a transformer language model pre-trained with maximum likelihood on WebText, a collection of scraped web pages (see Radford et al. [116]). GPT-2 has been observed to produce repetitive text with greedy and beam search [52].

Experimental setup. We use the Wikitext-103 dataset [90], a large-scale collection of Wiki articles with over 100 million words and 260 thousand unique tokens. We split the dataset into sequences according to the dataset's newline boundaries, then split each sequence into a context C and continuation Y, resulting in a dataset of (C, Y) pairs. Each continuation ends in a special $\langle\text{eos}\rangle$ token. We use a context size of $k = 10$ tokens, and discard sequences that are length k or shorter. The resulting dataset contains 874,556 training, 1,896 validation, and 2,162 test pairs.

We fine-tune the pre-trained GPT-2 model using maximum likelihood for 400k steps, and select the model state with the lowest validation perplexity (evaluated every 5k steps). Each training batch contains a maximum of 1024 total tokens. We use the implementation and default hyper-parameters from the `transformers` library [153]. We fine-tune the self-terminating GPT-2 models in a similar manner, starting from the pre-trained GPT-2 model and using the same hyper-parameters.

Each model is evaluated using greedy decoding with a maximum sequence length of 500, which was selected so that each decoded validation batch could fit in GPU memory. We define the non-termination ratio (r_L) using $L = 500$; this limit is more strict than the limit used in the preceding experiments (1500), yet still allows us to see large differences in generation behavior between the model and the ground truth (e.g., see Figure 5.4).

Results. Table 5.10 shows the non-termination ratio and perplexity of the baseline and self-terminating GPT-2 models. The self-terminating variant prevents non-termination, at the cost of perplexity. The model here uses $\epsilon = 2.5 \times 10^{-3}$, which we selected after observing that at higher values of ϵ, e.g., 1.0×10^{-3}, the self-terminating model generated sequences longer than the limit used to determine termination (500). Figure 5.4 shows the length distributions of the baseline GPT-2 continuations and those of the self-terminating GPT-2. The GPT-2 117M model generates many sequences at or near the maximum sequence length (500), unlike the ground-truth data. Introducing self-termination shifts the mass towards shorter sequences, whose lengths are also

	r_L (%)	perplexity
GPT2-117M	37.91	20.92
GPT2-117M ST	00.00	27.25

Table 5.10: Non-termination ratio (r_L (%)) of greedy-decoded sequences and perplexity for GPT2-117M and the self-terminating variant (ST) on Wikitext-103.

present in the ground-truth data.

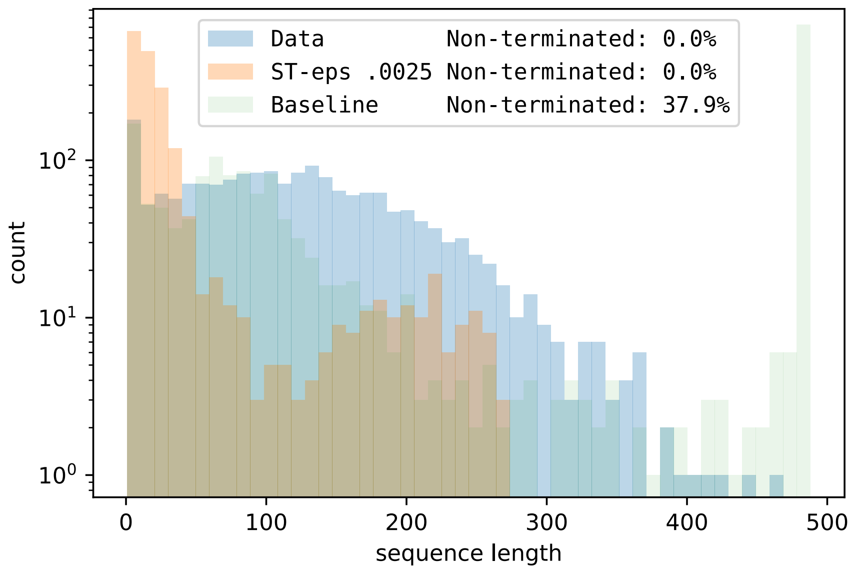


Figure 5.4: Lengths of ground-truth and greedy-decoded continuations from the baseline GPT-2 117M and self-terminating GPT-2 117M models ($\epsilon = 0.0025$).

5.6 Future Directions

The methods we proposed in this chapter resolve inconsistency by changing the decoding algorithm or model parametrization. Another approach is to address inconsistency in the *learning* phase. One interesting direction is to investigate whether the lack of decoding in maximum likelihood learning is a cause of inconsistency. Maximum likelihood learning fits the model p_θ using the data distribution, whereas a decoded sequence from the trained model follows the distribution $q_{\mathcal{F}}$ induced by a decoding algorithm. *Sequence-level* learning, however, uses a decoding algorithm during training (e.g., Ranzato et al. [117]), which we hypothesize can result in a good sequence generator that is consistent with respect to incomplete decoding.

5.7 Conclusion

We extended the notion of consistency of a recurrent language model put forward by Chen et al. [19] to incorporate a decoding algorithm, and used it to analyze the discrepancy between a model and the distribution induced by a decoding algorithm. We proved that incomplete decoding is inconsistent and proposed two methods to prevent this: consistent decoding and the self-terminating recurrent language model. Using a sequence completion task, we confirmed that empirical inconsistency occurs in practice, and that each method prevents inconsistency while maintaining the quality of the generated sequences. We suspect the absence of decoding in maximum likelihood estimation as a cause behind this inconsistency, and suggest investigating sequence-level learning as an alternative.

5.8 Developments since chapter release

We have published this work in the proceedings of EMNLP 2020 [148]. After the study of the non-termination issue and the analysis of consistency of the decoding-induced distribution, we attempted to characterize the discrepancy between the distributions induced by every stage of the learning pipeline [68]. In the context of mode-seeking decision rule, we proposed a novel mode recovery cost to quantify this discrepancy between any pair of sequence-level distributions e.g., the empirical data distribution and the model distribution. We, however, limited our experiments within a tractable toy setup, because the proposed measure is not computationally tractable with real-world distributions. In contrast to our work measuring the termination statistics, Pillutla et al. [112] designed "MAUVE" metric which measures the overlap between sequence-level distributions induced by large language models.

6 Importance of search and evaluation strategies in neural dialogue modeling

6.1 Introduction

There are three high-level steps to building a neural autoregressive sequence model for dialog modeling, inspired by work of Vinyals, Quoc, and Le [146]. First, decide on a network architecture which will consume previous utterances as well as any extra information such as speaker identifiers. Second, choose a learning strategy. Finally, decide on a search algorithm, as neural autoregressive sequence models do not admit a tractable, exact approach for generating the most likely response.

Recent research in neural dialogue modeling has often focused on the first two aspects. A number of variants of sequence-to-sequence models [21, 57, 142] have been proposed for dialogue modeling in recent years, including hierarchical models [131] and transformers [87, 157]. These advances in network architectures have often been accompanied by advanced learning algorithms. Serban et al. [129] introduce latent variables to their earlier hierarchical model and train it to maximize the variational lower bound, similar to Zhao, Zhao, and Eskenazi [164] who propose to build a neural dialogue model as a conditional variational autoencoder. Xu et al. [155]

and Li et al. [78] train a neural dialogue model as conditional generative adversarial networks [95]. These two learning algorithms, variational lower-bound maximization and adversarial learning, have been combined into a single model by Shen et al. [133], which has been followed by Gu et al. [46].

Despite abundant endeavors on modeling and learning, the search has received only a little attention [30]. Most of the work on search has focused on training an additional neural network that provides a supplementary score to guide either greedy or beam search. Li et al. [77] propose a maximum mutual information criterion for decoding using a reverse model. This has been extended by Li, Monroe, and Jurafsky [76], where an extra neural network is trained to predict an arbitrary reward given a partial hypothesis and used during decoding. Similarly, Zemlyanskiy and Sha [160] train a neural network that predicts the other participant's personality given a partial conversation and use its predictability as an auxiliary score for re-ranking a set of candidate responses. None of these approaches study how the choice of the underlying search algorithm, rather than its scoring function, affects the quality of the neural dialogue model.

In this paper, we investigate the effects of varying search and selection strategies on the quality of generated dialogue utterances. We start with an attention-based sequence-to-sequence model [6] trained on the recently-released PersonaChat dataset [161]. We evaluate three search algorithms: greedy search, beam search, and iterative beam search, the last of which we design based on earlier works by Batra et al. [10]. These algorithms are qualitatively different from each other in the size of the subspace over which they search for the best response.

We compare all of these alternatives using two families of metrics. First, we use human evaluation of a full, multi-turn conversation. The resulting distribution of annotator's scores has huge variance that is rarely discussed nor analyzed by other groups. This variance comes from each annotator's individual attitude towards and understanding of the task, which we call annotator bias. To address this bias, we propose model-based Bayesian calibration that explicitly factors in each annotator's bias and the algorithm's underlying score, and report the posterior mean and variance of each algorithm's score. Additionally, we also compare automatic metrics that capture the model's intrinsic preference (log-probability) and the diversity of responses (distinct-n).

We make two key observations from the experiments. A better search strategy can indeed generate responses that are both intrinsically preferred by the underlying model and diverse, without re-designing or re-training the neural dialogue model. However, this observation does not necessarily carry over to human evaluation, as the best performing strategy according to these automatic metrics was not the best strategy according to human annotators. These results highlight both the importance of search algorithms as well as the difficulty in evaluating neural dialogue systems in a realistic, full conversation setup.

Trained models, code and human evaluation transcripts publicly available[1].

6.2 Neural dialogue modeling

Since Vinyals, Quoc, and Le [146], a neural autoregressive sequence model based on sequence-to-sequence models Cho et al. [21] and Sutskever, Vinyals, and Le [142] have become one of the most widely studied approaches to dialogue modeling [see, e.g., 46, 75, 76, 78, 94, 129, 131, 133, 155, 160, 161, 164]. In this approach, a neural sequence model is used to model a conditional distribution over responses given a context which consists of previous utterances by both itself and a partner in the conversation as well as any other information about the speaker.

6.2.1 Neural autoregressive sequence modeling

A neural autoregressive sequence model learns the conditional distribution over all possible responses given the context. Each conditional distribution is modelled by a neural network, and popular choices include recurrent neural networks [6, 21, 93, 142], convolutional networks [28, 41] and self-attention [140, 143]. We explore search strategies and fix the model to a recurrent

[1]https://github.com/uralik/beamdream

neural network.

Learning: Maximum Likelihood Each example in the training set D consists of auxiliary information or context U (such as a persona profile or external knowledge context) and a sequence of utterances, each of which is marked with a speaker tag, i.e., $C = (U, (Y_1^a, Y_1^b, \ldots, Y_L^a, Y_L^b) \in D$, where Y_l^s is the utterance from the l-th turn by a speaker s. The conditional log-probability assigned to this example given by a neural sequence model is then written as

$$\log p(C) = \sum_{s\in\{a,b\}} \sum_{l=1}^{L} \log p(Y_l^s | Y_{<l}^s, Y_{\leq l}^{\bar{s}}, U), \tag{6.1}$$

where $\bar{s} = a$ if $s = b$ and otherwise $\bar{s} = b$.

Learning maximizes the log-probabilities of all conversations in the training set:

$$L = \frac{1}{|D|} \sum_{C\in D} \log p(C), \tag{6.2}$$

and is often done using stochastic gradient descent with backpropagation [123].

6.2.2 Inference (generation)

In this paper, we generate a response to the current state of the conversation (but do not attempt to plan ahead to future exchanges), maximizing

$$\log p(Y | Y_{<l}^s, Y_{<l}^{\bar{s}}, U) = \sum_{t=1}^{T} \log p(y_t | y_{<t}, Y_{<l}^s, Y_{<l}^{\bar{s}}, U).$$

Unfortunately, it is usually intractable to solve this problem due to the exponentially growing space of all possible responses w.r.t. the maximum length T. It is thus necessary to resort to approximate search algorithms.

Greedy search Greedy search has been the search algorithm of choice among the recent papers on neural dialogue modeling [46, 152, 155, 161, 164]. It moves from left to right selecting one token at a time, simply choosing the most likely token at the current time step:

$$\hat{y}_t = \arg\max_{v \in V} \log p(y_t = v | \hat{y}_{<t}, Y^s_{<l}, Y^{\bar{s}}_{<l}, U).$$

Greedy search has been found significantly sub-optimal within the field of machine translation [see, e.g., Table 1 in 20], where similar neural sequence models are frequently used.

Beam search Instead of maintaining a single hypothesis at a time, as in greedy search above, at time step t beam search maintains a set of K hypotheses $\mathcal{H}_t$:

$$\mathcal{H}_t = \{(y^1_1, \ldots, y^1_t), \ldots, (y^K_1, \ldots, y^K_t)\}. \tag{6.3}$$

Each hypothesis $h^i_{y^i_t}, i \in \{1, \ldots, K\}$ from $\mathcal{H}_t$ is expanded with all possible next tokens v from the vocabulary V to form candidate hypotheses. Each candidate is in the form of

$$\tilde{h}^i_v = h^i_{y_t} \| (v) = (y^i_1, \ldots, y^i_t, v), \tag{6.4}$$

and is assigned a score:

$$s(\tilde{h}^i_v) = s(h^i_{y^i_t}) + \log p(v | y^i_{\leq t}). \tag{6.5}$$

The new hypothesis set of K hypotheses is then constructed as

$$\mathcal{H}_{t+1} = \underset{i,v}{\text{arg-top-}k} \;\; s(\tilde{h}^i_v). \tag{6.6}$$

From the new hypothesis set, we find and copy *finalized* hypotheses (sequences ending with the special token ⟨eos⟩ for "end of sequence") to a candidate sequence set $\mathcal{M}_t$. That is,

$$\mathcal{M}_t = \left\{ h_v^i \in \mathcal{H}_{t+1} | v = \langle \text{eos} \rangle \right\}.$$

Beam search terminates when $|\cup_{t'=1}^{t} \mathcal{M}_t| \geq K'$, where K' is the maximum number of candidate sequences to be returned, or when $t \geq L_{\max}$, where $L_{\max}$ is the maximum length allowed for each candidate sequence. When terminated, beam search returns all the candidate sequences in $\mathcal{M} = \cup_{t'=1}^{t} \mathcal{M}_t$.

One can increase the size of the subspace over which beam search searches for a response and size of $\mathcal{M}$ by changing hyper-parameters $K, K', L_{\max}$. However, beam search is known to suffer from the problem that most of the hypotheses discovered in $\mathcal{M}$ are near each other in the response space [75, 77]. For tasks such as dialogue modeling, which are much more open-ended than, e.g., machine translation, this is particularly troublesome as many high-quality responses may be missing in the beam.

Final sequence selection We consider search strategies to produce a set of candidate responses for the model to choose from. While greedy search provides only a single possible sequence, beam search generates a candidate set of size $|\mathcal{M}|$. It is usual practice to use the score $s(h)$ used during the search to select the final sequence, but it is an open question whether there are better selection strategies for choosing between these final candidate responses.

Avoiding repeating n-grams Although this has not been reported in a formal publication in the context of neural dialogue modeling, to our knowledge, Paulus, Xiong, and Socher [108] and Klein et al. [60] implement so-called n-gram blocking. In n-gram blocking, a hypothesis in a beam $\mathcal{H}_t$ is discarded if there is an n-gram that appears more than once within it.

6.3 Uncovering hidden responses

We now propose an improved search strategy. To address the locality issue in beam search, we propose an iterative beam search to radically increase the size of the search space without introducing much computational overhead, inspired by earlier work on diverse beam search [10, 75, 145].

6.3.1 Iterative beam search

The search space over which beam search has operated can be characterized by the union of all partial hypothesis sets $\mathcal{H}_t$ in Equation (6.3): $\mathcal{S}_0 = \cup_{t=1}^{T} \mathcal{H}_t$, where we use the subscript 0 to indicate that beam search has been done without any other constraint. Re-running beam search with an increased beam width K would result in the search space that overlaps significantly with $\mathcal{S}_0$, and would not give us much of a benefit with respect to the increase in computation.

Instead, we keep the beam size K constant but run multiple iterations of beam search while ensuring that any previously explored space $\bar{\mathcal{S}}_{<l} = \cup_{l'=0}^{l-1} \mathcal{S}_{l'}$ is *not* included in a subsequent iteration of beam search. This is done by setting the score of each candidate hypothesis $s(\tilde{h}_{t+1}^{i})$ in Equation (6.5) to negative infinity, when this candidate is included in $\bar{\mathcal{S}}_{<l}$. We relax this inclusion criterion by using a non-binary dissimilarity metric, and say that the candidate is included in $\bar{\mathcal{S}}_{<l}$, if

$$\min_{h \in \bar{\mathcal{S}}_{<l}} \Delta(\tilde{h}_{t+1}^{i}, h) < \epsilon, \tag{6.7}$$

where Δ is a string dissimilarity measure, such as Hamming distance used in this work, and ϵ is a similarity threshold.

This procedure ensures that the new partial hypothesis set of beam search in the l-th iteration minimally overlaps with any part of the search space explored earlier during the first $l-1$

iterations of beam search. By running this iteration multiple times, we end up with a set of top hypotheses from each iteration of beam search, from which the best one is selected according to, for instance, the log-probability assigned by the model. We build a final candidate set $\mathcal{M}$ as a set of all these best hypotheses from beam search iterations.

Practical implementation A major issue with iterative beam search in its naive form is that it requires running beam search multiple times, when even a single run of beam search can be prohibitively slow in an interactive environment, such as in dialogue generation. We address this computational issue by performing these many iterations of beam search in parallel simultaneously. At each time step in the search, we create sets of candidate hypotheses for all iterations in parallel, and go through these candidate sets in a sequence from the $(l = 0)$-th iteration down to the last iteration, while eliminating those candidates that satisfy the criterion in Equation (6.7). We justify this parallelized approach by defining the similarity measure Δ to be always larger than the threshold ϵ when the previous hypothesis h is longer than $\tilde{h}^i_{t+1}$ in Equation (6.7).

6.4 Dialogue evaluation

Broadly, there are two ways to evaluate a neural dialogue model. The first approach is to use a set of (often human-generated) reference responses and compare a single generated response against them [83, 130]. There are several methods for this comparison: (1) measure the perplexity of reference responses using the neural dialogue model, (2) compute a string match-based metric of a generated response against reference responses, and (3) use human annotators to compare model-generated responses against reference or other models' responses. None of these approaches capture the effectiveness of a neural sequence model in conducting a full conversation, because the model responses are computed given a human-written context, i.e., it does not see its own responses in the dialogue history, but gold responses only.

We concentrate on a second approach for evaluation where a neural dialogue model has a multi-turn conversation with a human partner (or annotator) [30, 152, 160, 161]. Unlike other approaches, it requires active human interaction, as a conversation almost always deviates from a previously collected data even with the same auxiliary information (U in Equation (6.1)). This evaluation strategy reflects both how well a neural dialogue model generates a response given a correct context as well as how well it adapts to a dynamic conversation—the latter is not measured by the first strategy, where the model only has to generate a single response.

6.4.1 Human evaluation of a full conversation

An annotator is asked to make a conversation with a randomly selected model (search strategy) for at least five turns. At the end of the conversation, we ask the annotator three sets of questions:

1. Overall score ($\{1, 2, 3, 4\}$)
2. Marking of each *good* utterance-pair ($\{0, 1\}$)
3. Marking of each *bad* utterance-pair ($\{0, 1\}$)

The first overall score allows us to draw a conclusion on which algorithm makes a better conversation overall. We use a 4 point scale in order to avoid having a "catch-all" category in the answer [26]. The latter two questions are collected to investigate the relationship between the overall impression and the quality of each utterance-pair.

6.4.2 Bayesian calibration

Although human evaluation is desirable, raw scores collected by annotators are difficult to use directly due to the annotator bias. Some are more generous while others are quite harsh, as recently reported in Zemlyanskiy and Sha [160] and Zhang et al. [161]. We propose using

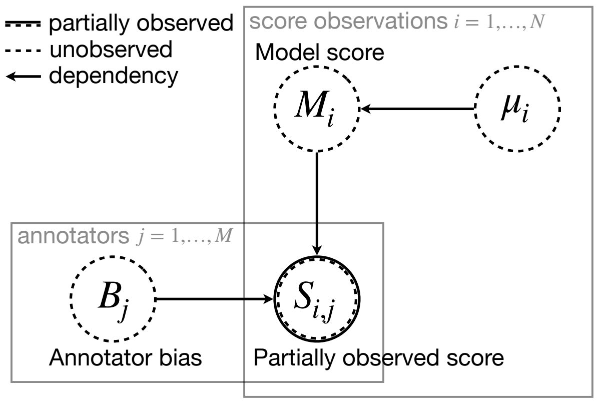


Figure 6.1: Graphical model used for Bayesian Calibration. M annotators participated such that in total N observed scores are presented.

Bayesian inference as a framework to account for the bias of each annotator and describe two instances of this framework.

1-4 STAR RATING OF A CONVERSATION We treat both the unobserved score M_i of each model, in our case each search algorithm, and the unobserved bias B_j of each annotator as latent variables. The score of the i-th model follows the following distribution: $\mu_i \sim \mathcal{U}(1,4)$ and $M_i \sim \mathcal{N}(\mu_i, 1^2)$, where $\mathcal{U}$ and $\mathcal{N}$ are uniform and normal distributions. It states that *a priori* each model is likely to be uniformly good or bad. The annotator bias B_j follows $B_j \sim \mathcal{N}(0, 1^2)$, where we are assuming that each annotator does not have any bias *a priori*.

Given the model score M_i and annotator bias B_j, the conditional distribution over an observed score S_{ij} given by the j-th annotator to the i-th model is then:

$$S_{ij} \sim \mathcal{N}(M_i + B_j, 1^2).$$

Due to the nature of human evaluation, only a few of S_{ij}'s are observed. Figure 6.1 shows the graphical model described above.

The goal of inference in this case is to infer the posterior mean and variance:

$$\mathbb{E}[M_i|\left\{S_{ij}|S_{ij} \in \mathcal{O}\right\}], \qquad (6.8)$$
$$\mathbb{V}[M_i|\left\{S_{ij}|S_{ij} \in \mathcal{O}\right\}],$$

where $\mathcal{O}$ is a set of observed scores.

Binary rating of an utterance When an annotator labels pairs of utterances from the conversation with a binary score $\{0, 1\}$ (such as whether that pair was a "good" exchange), we need to further take into account the turn bias T_k: $T_k \sim \mathcal{N}(0, 1^2)$. As we will use a Bernoulli distribution for each observed score rather than a 1-4 rating, we modify the prior of the model scores accordingly: $M_i \sim \mathcal{N}(0, 1^2)$.

The distribution of an observed utterance-pair score is then $S_{ijk} \sim \mathcal{B}(\text{sigmoid}(M_i + B_j + T_k))$, where $\mathcal{B}$ is a Bernoulli distribution. The goal of inference is then to compute

$$\mathbb{E}_{M_i|\left\{S_{ijk}|S_{ijk} \in \mathcal{O}\right\}}\left[\text{sigmoid}(M_i)\right], \qquad (6.9)$$
$$\mathbb{V}_{M_i|\left\{S_{ijk}|S_{ijk} \in \mathcal{O}\right\}}\left[\text{sigmoid}(M_i)\right],$$

which estimate the average number of positively labelled utterance-pairs given the i-th model and the uncertainty in this estimate, respectively.

Inference We use no-u-turn (NUTS) sampler [50] for posterior inference in Pyro [13].

6.5 Experiment Settings

6.5.1 Data: Persona-Chat

We use Persona-Chat, released recently by Zhang et al. [161] and the main dataset for the Conversational Intelligence Challenge 2 (ConvAI2),[2] to train a neural dialogue model. The dataset contains dialogues between pairs of speakers randomly assigned personas from a set of 1,155, each consisting of 4-5 lines of description about the part they should play, e.g., *"I have two dogs"* or *"I like taking trips to Mexico"*. The training set consists of 9,907 dialogues where pairs of partners play their roles, and a validation set of 1,000 dialogues. The ConvAI2 test set has not been released. Each dialogue is tokenized into words, resulting in a vocabulary of 19,262 unique tokens. See Zhang et al. [161] for more details.

6.5.2 Neural dialogue modeling

Model We closely follow Bahdanau, Cho, and Bengio [6] in building an attention-based neural autoregressive sequence model. The encoder has two bidirectional layers of 512 LSTM [49] units each direction-layer, and the decoder has two layers of 512 LSTM units each. We use global general attention as described by Luong, Pham, and Manning [85]. We use the same word embedding matrix for both the encoder and decoder, which is initialized from 300-dimensional pretrained GloVe vectors [109] for the 97% of the vocabulary which overlaps with GloVe. We allow word embedding weights to be updated during the training.

Learning We use Adam [58] with the initial learning rate set to 0.001. We apply dropout [137] between the LSTM layers with the dropout rate of 0.5 to prevent overfitting. We train the neural dialogue model until it early-stops on the validation set.[3]

[2] http://convai.io/

[3] When the validation loss does not improve for twelve epochs, we early-stop.

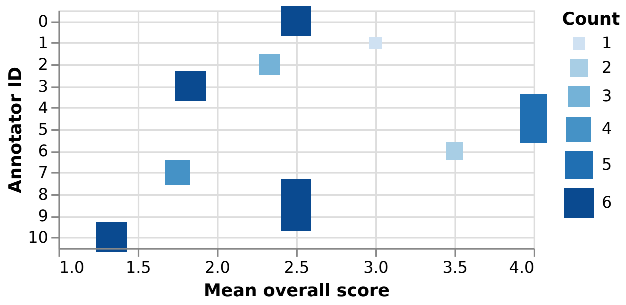


Figure 6.2: The distribution of averaged overall scores given by annotators to greedy search (**greedy**). Each row plots scores given by a single annotator over multiple conversations. Counts show how many dialogues each annotator performed.

The perplexity of trained model on the ConvAI2 validation set is 24.84, which is competitive compared to the other entries on the competition's leaderboard.[4] Our model serves well as an underlying system for investigating the effect of search algorithms.

6.5.3 Search Strategies

We test three search strategies; **greedy** and **beam** search from Section 6.2.2, and iterative beam search (**iter-beam**) from Section 6.3.1.

Beam search (**beam**) uses beam size $K = 5$ and $K' = 15$. This decision is based on preliminary experiments where we found that smaller beam sizes work better than larger ones do. We use the length penalty, described by Wu et al. [154] and n-gram blocking from Section 6.2.2.

Iterative beam search (**iter-beam**) uses 15 iterations of beam search with beam size 5 resulting in a candidate set of size 15. We use the same length penalty and n-gram blocking as in beam search (**beam**). Given the hyper-parameters above both **beam** and **iter-beam** produce 15 candidates and selects the final response based on log-probability.

[4] https://github.com/DeepPavlov/convai/blob/master/leaderboards.md

Search strategy	log-p↑	Overall Score (1-4)↑		% Good Pairs↑		% Bad Pairs↓	
		Raw	Calibrated	Raw	Calibrated	Raw	Calibrated
greedy	-9.66±2.73	2.56±0.98	2.30±0.24	0.45	0.28±0.07	0.38	0.54±0.07
beam	-7.26±2.28	2.67±0.86	2.70±0.27	0.58	0.44±0.08	0.35	0.27±0.01
iter-beam	**-5.95±1.35**	2.80±0.90	2.67±0.23	0.58	0.45±0.08	0.31	0.32±0.03
human	-42.95±18.87	3.62±0.71	3.37±0.22	0.76	0.76±0.06	0.07	0.04±0.003

Table 6.1: The average log-probabilities and model scores (average±standard deviation) assigned to the responses during human evaluation. Better search algorithms find responses with higher log-probabilities according to the model. Without observing standard deviations and calibrated scores one can make erroneous conclusions.

6.5.4 Evaluation

Human evaluation We use ParlAI [94] which provides seamless integration with Amazon Mechanical Turk (MTurk) for human evaluation. A human annotator is paired with a model with a specific search strategy, and both are randomly assigned personas out of a set of 1,155, and are asked to make a conversation of at least either five or six turns (randomly decided). We allow each annotator to participate in at most six conversations per search strategy and collect approximately 50 conversations per search strategy and additional human-human test.[5] Each conversation is given a single overall score and two sequences of binary utterance-pair flags, as described in Section 6.4.1.

Bayesian calibration In order to address annotator bias or inter-annotator variability, we use Bayesian calibration from Section 6.4.2. We take 50 warm-up steps and collect 150 samples using NUTS sampler for inferring the posterior mean and variance of the model score in Equation (6.8). We use 30 warm-up steps and 100 samples for inferring the mean and variance of the average portion of positively or negatively labelled utterance-pairs in Equation (6.9).[6]

[5]Some conversations were dropped due to technical errors, resulting in total 50, 51, 49 and 53 conversations for **greedy**, **beam**, **iter-beam** and **humans**, respectively.

[6]Variances of inferred posterior distribution and original data distribution are not comparable, as the former reflects the uncertainty in posterior inference rather than the spread of scores.

Automatic metrics In addition to human evaluation, we compute automatic metrics to quantitatively characterize each search algorithm. First, we report the **log-p**robability of a generated response assigned by the model which is a direct indicator of the quality of a search algorithm. Second, we compute the average number of unique n-grams generated per conversation normalized by the number of generated tokens in the conversation, called **distinct-n** from [77], with $n = 1, 2, 3$.

We compute distinct-n in two different settings. First, we compute distinct-n over the candidate set $\mathcal{M}$ given by the search algorithm. Second, we compute distinct-n over the final selected responses for each search strategy. The former shows diversity within the possible response candidates, while the latter shows diversity among the actual selected dialogue outputs.

6.6 Result

6.6.1 Human Evaluation

Annotator bias In Figure 6.2, we plot the averaged scores provided by the human annotators for one search strategy (**greedy**), where each row corresponds to each annotator. Consider three annotators with id 3, 4, 10. Their means are clearly separated from each other, which points to the existence of annotator bias. This observation supports the necessity of the Bayesian calibration described in §6.4.2.

Human evaluation In Table 6.1, we present the scores from human evaluation. In total, 41 unique annotators participated within 201 collected conversations. We make a major observation which is that greedy search (**greedy**), which has been the search algorithm of choice in neural dialogue modeling, significantly lags behind the variants of beam search (**beam, iter-beam**) in all metrics. This stark difference is worth our attention, as this difference is *solely* due to the choice of a search algorithm and is not the result of different network architectures nor learning

	distinct-n ↑					
Search	$n = 1$		$n = 2$		$n = 3$	
strategy	post	pre	post	pre	post	pre
greedy	0.47	-	0.61	-	0.62	-
beam	0.56	0.06	**0.69**	0.12	**0.63**	0.18
iter-beam	**0.59**	**0.18**	0.68	**0.41**	0.60	**0.58**
human	0.66	-	0.85	-	0.82	-

Table 6.2: Measuring the diversity of different search and selection strategies. Both **beam** and **iter-beam** produce up to 15 hypotheses each. Column **post** is distinct-n measured over the final selected output responses given by the model and allows us to compare the diversity of the best responses each search procedure produces. Column **pre** is distinct-n measured over the candidate set $\mathcal{M}$ given by the search algorithm and allows us to compare diversity generated within the search.

algorithms. In fact, this cannot even be attributed to different parameter initialization, as we use only *one* trained model for all of these results.

The model scores assigned to human conversations (**humans**) are far superior to all search strategies. It is clear with both overall score and utterance pairs proportion scores. This tells us that there are many open questions how to improve neural dialogue models.

6.6.2 Automatic Metrics

Search quality: log-probability (**log-p**) Better search algorithms find responses with higher log-probability according to the model, as shown in Table 6.1. This is a natural consequence of exploring a larger subset of the search space.

A notable observation from Table 6.1 is that the neural sequence model assigns very low log-probabilities to human responses. This implies that there is a limit to improving this specific neural dialogue model by using a better search algorithm and instead there is more room to improve the model and learning algorithms to place a high probability on human responses. It is necessary to test the proposed search strategies with new models and we leave this for the future.

Beam search	Iterative beam search
do you have any pets ?	that ' s cool , what do you like to eat ?
what is your favorite animal ?	do you have a favorite color ? mine is pink .
i like to talk about strangers .	what do you like to eat ?
do you like animals ?	i don ' t like fish , but my favorite color is pink .
do you like animals ? i want to live at the beach .	what color is your hair ?
do you like animals ? i ' ve a pet .	that does sound good , i like to go alone .
do you like animals ? i want to live on the beach .	i would love to eat fish .
do you like animals ? i want to live at the beach	that makes sense , do you have any hobbies ?
do you like animals ? i ' ve a monkey .	i hear you , my favorite color is pink .
do you like animals ? i want to be a monkey .	i want to be a yoga instructor .
do you like animals ? i want to live at the beach , but love it .	i did not eat meat , but my favorite color is pink .
do you like animals ? i want to live at the beach , but love monkeys .	what are your favorite foods ? mine is pink .
do you like animals ? i want to live at the beach , but they are my favorite .	what does your favorite color ? mine is pink .
do you like animals ? i want to live at the beach , but have a monkey .	what type of food do you like ?
do you like animals ? i want to live at the beach , but they are my favorite	i spend a lot of time alone .

Table 6.3: beam and **iter-beam** candidate sets $\mathcal{M}$. These are from one turn of one randomly selected conversation from human evaluation. **iter-beam** produces more diverse responses.

Diversity: distinct-n The diversity metric is measured before (pre) and after (post) selecting the final response from the candidate set $\mathcal{M}$ for both **beam** search and **iter-beam** search. Since **greedy** and **humans** produce only a single response, we compute the diversity metric only using those final responses for both greedy search and humans. In both pre and post settings, the normalization is done per each conversation.

As well as in human evaluation, **greedy** has lower diversity compared to all other strategies as shown in Table 6.2. We see large gap in pre-selection distinct-n for all n between **beam** and **iter-beam** while the difference is small in post-selection distinct-n. In other words, while providing more diverse set of candidates, the final selected output response with **iter-beam** is not particularly diverse. This agrees well with human evaluation, where both **iter-beam** and **beam** model scores were indistinguishable, as annotators could only see the final response after selecting from the candidate set. Table 6.3 shows pre-selection candidate sets for both beam search and iterative beam search.

Finally, we observe a significant gap between the best search strategy and humans in these diversity metrics. Together with the gap we observed in human evaluation scores, we suspect that the lack of diversity in the output responses is a major factor behind the low performance of the tested neural dialogue model in the human evaluation.

6.7 Conclusion and Discussion

We have performed a realistic human evaluation of the neural dialogue model to validate the importance of exploring better search strategies. We observed that careful design of human evaluation is necessary to properly evaluate the ability of the neural dialogue model to conduct a conversation with a human. The proposed Bayesian calibration of model scores helps to account the annotator bias observed in human evaluation.

Extensive analysis reveals that greedy search, which has been the inference algorithm of choice in neural dialogue modeling, significantly lags behind more sophisticated search strategies such as beam search and iterative beam search.

We have proposed the iterative beam search which produces a more diverse set of candidate responses w.r.t. pre-selection distinct-n metric. Post-selection final responses with iterative beam search have higher log-probability compared to other search strategies. In spite of this, there is only a marginal difference between iterative beam search and beam search w.r.t. scores from human evaluation and post-selection distinct-n. This suggests that the final response selection strategy is as important as the search strategy being used and can be a major factor in the inference pipeline of the neural dialogue model. We leave improving this strategy to future work.

Finally, the model assigns a lower probability to the reference responses, which implies suboptimality in the current neural dialogue model. It is necessary in the future to test the proposed search strategies with new models.

6.8 Developments since chapter release

We have published this work in the proceedings of INLG 2019 [69]. As a follow-up, we designed an alternative search algorithm called multi-turn beam search which aims at finding the best response in the context of rolled out future responses [67]. The task of modeling the partner persona has shown to be hard and requires further investigation. Common choices for model design and search strategy in open-ended dialogue modeling have drastically changed since then, but challenges remain. Large pretrained language models become the *de facto* standard initialization scheme of such systems [120]. The common choice for decoding strategy involves stochastic sampling, because the deterministic algorithms, such as beam search, produce dull and degenerate responses [1]. While the problem of a dull response is somewhat alleviated via stochastic sampling during search, the issue of contradictions in responses is still there [135]. Nie et al. [101] design a new task concentrating on this issue of contradictions directly.

Regarding the human evaluation, other groups suggested alternative protocols and methods to conduct the evaluation. For instance, Li, Weston, and Roller [80] suggest to do model judgement via pair-wise dialogue selection based on a set of questions which are optimized to maximize the robustness of judgements across different annotators. Sedoc and Ungar [124] employ item response theory (IRT) in the context of dialogue model evaluation. Importance of human evaluation protocol is actively studied beyond dialogue modeling. Howcroft et al. [54] discussed common practices which may lead to confusions during Natural Language Generation systems comparison. Howcroft and Rieser [53] proposed to use ordinal mixed effects models to analyze ordinal observations opposed to mapping those points on the interval scale.

7 Conclusion

In this work, we addressed degeneracies in autoregressive text generation from different aspects of the learning pipeline. We conclude that the widely used algorithms and techniques have practical issues resulting in degenerate predictions. During the analysis of these issues, we observed that some of the proposed solutions lack principled reasoning behind the issue's cause. We consider several types of degeneracy and demonstrate effective ways to mitigate by addressing the hypothetical cause. Every solution we propose is connected to some part of the learning pipeline: model parametrization, training, or decoding. We have addressed the oversmoothing problem, the problem of non-termination, the issue of repetition, and studied the lack of diversity in dialogue modeling. In addition to the proposed solutions, we provide new formulations, measures, and theoretical motivation behind some of these problems. Each of these contributions paves the way for future research. Next, we speculate on what future directions are promising in our view given the results we obtained so far.

Regarding the oversmoothing problem, our experiments are limited to the scope of machine translation. We encourage future work on studying the issue of oversmoothing with other text generation tasks. For instance, it is an open question whether the degree of oversmoothing depends on the amount of information in the input sequence. The unreasonable effectiveness of beam search with smaller beam size remains to be studied in the future.

Unlikelihood training can effectively reduce the number of repetitions in the generated sequences. As we discussed earlier and studied in Li et al. [81], the unlikelihood framework is applicable beyond the repetition issue. Future work may consider a more generic way of finding useful negative candidates. Even though the solution we proposed is effective, the question of why exactly the maximum likelihood learning results in repetition loops is still actively studied [40].

We have provided a theory behind the non-termination issue. It demonstrates the inconsistency of the decoding-induced distribution when the decoding algorithm is incomplete. By altering the model with self-terminating softmax, we make this distribution consistent with theoretical guarantees. The proposed solution has a hyper-parameter defining the lower bound on the probability of termination. While we used the same lower bound across all sentences, it is not necessarily the best way to go. We encourage future work on learning to compute this lower bound based on the current input sequence. Other than that, we confess that our approach puts a very rigid constraint on the probability of termination. The question of more flexible parametrization with consistency guarantees remains to be open for future research.

The multi-turn setting of text generation is even more challenging as the model obtains new user input as well as its own predictions. One of the promising recent directions is the retrieval augmented generation [74] which has been studied in dialogue modeling too. For instance, Komeili, Shuster, and Weston [64] suggest to augment a dialogue response generator with the internet-based context retriever compared to the usual fixed cache. We found that more likely responses were not more engaging in the end. One future direction is to consider alternative decision rules with different scoring or reranking methods, because scaling the model alone does not completely eliminate inconsistencies in generated responses.

Bibliography

[1] Daniel Adiwardana et al. *Towards a Human-like Open-Domain Chatbot.* 2020.

[2] Roee Aharoni, Melvin Johnson, and Orhan Firat. “Massively Multilingual Neural Machine Translation”. In: *ArXiv* abs/1903.00089 (2019).

[3] Daniel Andor et al. “Globally normalized transition-based neural networks”. In: *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers.* Vol. 4. 2016, pp. 2442–2452. ISBN: 9781510827585. DOI: 10.18653/v1/p16-1231.

[4] Ebrahim Ansari et al. “FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN”. In: *IWSLT.* 2020.

[5] Alexei Baevski and Michael Auli. “Adaptive Input Representations for Neural Language Modeling”. In: *International Conference on Learning Representations.* 2019.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* 2015.

[7] Hareesh Bahuleyan and Layla El Asri. “Diverse Keyphrase Generation with Neural Unlikelihood Training”. In: *COLING.* 2020.

[8] Anton Bakhtin et al. “Residual Energy-Based Models for Text”. In: *Journal of Machine Learning Research* 22.40 (2021), pp. 1–41.

[9] Rafael E. Banchs. “Movie-DiC: a Movie Dialogue Corpus for Research and Development”. In: *ACL*. 2012.

[10] Dhruv Batra et al. “Diverse m-best solutions in markov random fields”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 1–16.

[11] Emily M. Bender et al. “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?” In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (2021).

[12] Samy Bengio et al. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. 2015.

[13] Eli Bingham et al. “Pyro: Deep Universal Probabilistic Programming”. In: *arXiv preprint arXiv:1810.09538* (2018).

[14] T. L. Booth and R. A. Thompson. “Applying Probability Measures to Abstract Languages”. In: *IEEE Transactions on Computers* C-22.5 (May 1973), pp. 442–450. ISSN: 2326-3814. DOI: 10.1109/T-C.1973.223746.

[15] Kaj Bostrom and Greg Durrett. “Byte Pair Encoding is Suboptimal for Language Model Pretraining”. In: *FINDINGS*. 2020.

[16] John S Bridle. “Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition”. In: *Neurocomputing*. Springer, 1990, pp. 227–236.

[17] Tom B Brown et al. “Language models are few-shot learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[18] William Chan et al. “An Empirical Study of Generation Order for Machine Translation”. In: *ArXiv* abs/1910.13437 (2020).

[19] Yining Chen et al. “Recurrent neural networks as weighted language recognizers”. In: *arXiv preprint arXiv:1711.05408* (2017).

[20] Yun Chen et al. “A Stable and Effective Learning Strategy for Trainable Greedy Decoding”. In: *arXiv preprint arXiv:1804.07915* (2018).

[21] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[22] Kyunghyun Cho et al. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012.

[23] Yejin Choi. “The Missing Representation in Neural (Language) Models”. In: *3rd Workshop on Representation Learning for NLP (RepL4NLP)* (2018).

[24] Michael Collins. “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms”. In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*. 2002, pp. 1–8. DOI: 10.3115/1118693.1118694.

[25] Michael Collins. “Statistical machine translation: IBM models 1 and 2”. In: *Columbia Columbia Univ* (2011).

[26] Dev K Dalal, Nathan T Carter, and Christopher J Lake. “Middle response scale options are inappropriate for ideal point scales”. In: *Journal of Business and Psychology* 29.3 (2014), pp. 463–478.

[27] Hal Daumé, John Langford, and Daniel Marcu. “Search-based structured prediction”. In: *Machine learning* 75.3 (2009), pp. 297–325.

[28] Yann N Dauphin et al. “Language modeling with gated convolutional networks”. In: *arXiv preprint arXiv:1612.08083* (2016).

[29] Adji B Dieng et al. “Learning with Reflective Likelihoods”. In: (2018).

[30] Emily Dinan et al. “The Second Conversational Intelligence Challenge (ConvAI2)”. In: *arXiv preprint arXiv:1902.00098* (2019).

[31] Emily Dinan et al. “Wizard of Wikipedia: Knowledge-Powered Conversational agents”. In: *ArXiv* abs/1811.01241 (2019).

[32] Sergey Edunov et al. “Classical structured prediction losses for sequence to sequence learning”. In: *arXiv preprint arXiv:1711.04956* (2017).

[33] Bryan Eikema and Wilker Aziz. “Is map decoding all you need? the inadequacy of the mode in neural machine translation”. In: *arXiv preprint arXiv:2005.10283* (2020).

[34] Jeffrey L Elman. “Finding structure in time”. In: *Cognitive science* 14.2 (1990), pp. 179–211.

[35] Angela Fan, David Grangier, and Michael Auli. “Controllable abstractive summarization”. In: *arXiv preprint arXiv:1711.05217* (2017).

[36] Angela Fan, Mike Lewis, and Yann Dauphin. “Hierarchical neural story generation”. In: *arXiv preprint arXiv:1805.04833* (2018).

[37] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. “Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism”. In: *NAACL*. 2016.

[38] Markus Freitag et al. *Experts, Errors, and Context: A Large-Scale Study of Human Evaluation for Machine Translation.* 2021.

[39] Markus Freitag et al. “Minimum Bayes Risk Decoding with Neural Metrics of Translation Quality”. In: *arXiv preprint arXiv:2111.09388* (2021).

[40] Zihao Fu et al. “A Theoretical Analysis of the Repetition Problem in Text Generation”. In: *AAAI*. 2021.

[41] Jonas Gehring et al. “Convolutional sequence to sequence learning”. In: *arXiv preprint arXiv:1705.03122* (2017).

[42] Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. *An Empirical Investigation of Global and Local Normalization for Recurrent Neural Sequence Models Using a Continuous Relaxation to Beam Search*. 2019.

[43] Edouard Grave, Armand Joulin, and Nicolas Usunier. “Improving neural language models with a continuous cache”. In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. 2017.

[44] Alex Graves. “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850* (2013).

[45] Klaus Greff et al. *LSTM: A Search Space Odyssey*. 2015.

[46] Xiaodong Gu et al. “DialogWAE: Multimodal Response Generation with Conditional Wasserstein Auto-Encoder”. In: *arXiv preprint arXiv:1805.12352* (2018).

[47] Tianxing He and James Glass. “Negative Training for Neural Dialogue Response Generation”. In: *arXiv preprint arXiv:1903.02134* (2019).

[48] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[49] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[50] Matthew D Hoffman and Andrew Gelman. “The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1593–1623.

[51] Ari Holtzman et al. “Learning to Write with Cooperative Discriminators”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 1638–1649.

[52] Ari Holtzman et al. “The curious case of neural text degeneration”. In: *arXiv preprint arXiv:1904.09751* (2019).

[53] David M Howcroft and Verena Rieser. “What happens if you treat ordinal ratings as interval data? Human evaluations in NLP are even more under-powered than you think”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 8932–8939.

[54] David M. Howcroft et al. “Twenty Years of Confusion in Human Evaluation: NLG Needs Evaluation Sheets and Standardised Definitions”. In: *Proceedings of the 13th International Conference on Natural Language Generation*. Dublin, Ireland: Association for Computational Linguistics, Dec. 2020, pp. 169–182.

[55] Sébastien Jean et al. “On Using Very Large Target Vocabulary for Neural Machine Translation”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 1–10.

[56] Shaojie Jiang et al. “TLDR: Token Loss Dynamic Reweighting for Reducing Repetitive Utterance Generation”. In: *ArXiv* abs/2003.11963 (2020).

[57] Nal Kalchbrenner and Phil Blunsom. “Recurrent continuous translation models”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1700–1709.

[58] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[59] Diederik P Kingma and Jimmy Lei Ba. “Adam: A Method For Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[60] Guillaume Klein et al. “OpenNMT: Open-Source Toolkit for Neural Machine Translation”. In: *Proc. ACL*. 2017. DOI: 10.18653/v1/P17-4012.

[61] Guillaume Klein et al. “Opennmt: Open-source toolkit for neural machine translation”. In: *arXiv preprint arXiv:1701.02810* (2017).

[62] Philipp Koehn and Rebecca Knowles. “Six challenges for neural machine translation”. In: *arXiv preprint arXiv:1706.03872* (2017).

[63] Philipp Koehn, Franz J Och, and Daniel Marcu. *Statistical phrase-based translation*. Tech. rep. UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 2003.

[64] Mojtaba Komeili, Kurt Shuster, and Jason Weston. “Internet-augmented dialogue generation”. In: *arXiv preprint arXiv:2107.07566* (2021).

[65] Taku Kudo and John Richardson. “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. DOI: 10.18653/v1/D18-2012.

[66] Ilia Kulikov, Maksim Eremeev, and Kyunghyun Cho. *Characterizing and addressing the issue of oversmoothing in neural autoregressive sequence modeling*. 2021.

[67] Ilia Kulikov, Jason Lee, and Kyunghyun Cho. *Multi-Turn Beam Search for Neural Dialogue Modeling*. 2019.

[68] Ilia Kulikov, Sean Welleck, and Kyunghyun Cho. “Mode recovery in neural autoregressive sequence modeling”. In: *ArXiv* abs/2106.05459 (2021).

[69] Ilia Kulikov et al. *Importance of Search and Evaluation Strategies in Neural Dialogue Modeling*. 2019.

[70] John Lafferty, Andrew McCallum, and Fernando C N Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: *ICML ’01 Proceedings of the Eighteenth International Conference on Machine Learning* (2001). ISSN: 1750-2799. DOI: 10.1038/nprot.2006.61.

[71] Yann LeCun et al. “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0 (2006).

[72] Ann Lee, Michael Auli, and Marc’Aurelio Ranzato. “Discriminative Reranking for Neural Machine Translation”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 7250–7264. DOI: 10.18653/v1/2021.acl-long.563.

[73] Katherine Lee et al. “Deduplicating Training Data Makes Language Models Better”. In: *ArXiv* abs/2107.06499 (2021).

[74] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *arXiv preprint arXiv:2005.11401* (2020).

[75] Jiwei Li, Will Monroe, and Dan Jurafsky. “A simple, fast diverse decoding algorithm for neural generation”. In: *arXiv preprint arXiv:1611.08562* (2016).

[76] Jiwei Li, Will Monroe, and Dan Jurafsky. “Learning to decode for future success”. In: *arXiv preprint arXiv:1701.06549* (2017).

[77] Jiwei Li et al. “A diversity-promoting objective function for neural conversation models”. In: *arXiv preprint arXiv:1510.03055* (2015).

[78] Jiwei Li et al. “Adversarial learning for neural dialogue generation”. In: *arXiv preprint arXiv:1701.06547* (2017).

[79] Margaret Li, Jason Weston, and Stephen Roller. “ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-turn Comparisons”. In: *arXiv preprint arXiv:1909.03087* (2019).

[80] Margaret Li, Jason Weston, and Stephen Roller. “ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-turn Comparisons”. In: *ArXiv* abs/1909.03087 (2019).

[81] Margaret Li et al. *Don’t Say That! Making Inconsistent Dialogue Unlikely with Unlikelihood Training*. 2019.

[82] Xiang Lin, Simeng Han, and Shafiq R. Joty. “Straight to the Gradient: Learning to Use Novel Tokens for Neural Text Generation”. In: *ArXiv* abs/2106.07207 (2021).

[83] Chia-Wei Liu et al. “How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation”. In: *arXiv preprint arXiv:1603.08023* (2016).

[84] Ryan Lowe et al. “The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems”. In: *SIGDIAL Conference*. 2015.

[85] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. “Effective approaches to attention-based neural machine translation”. In: *arXiv preprint arXiv:1508.04025* (2015).

[86] Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. “Sparse Text Generation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 4252–4273. DOI: 10.18653/v1/2020.emnlp-main.348.

[87] Pierre-Emmanuel Mazaré et al. “Training Millions of Personalized Dialogue Agents”. In: *arXiv preprint arXiv:1809.01984* (2018).

[88] Gábor Melis, Chris Dyer, and Phil Blunsom. “On the state of the art of evaluation in neural language models”. In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. 2018.

[89] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. “Regularizing and optimizing LSTM language models”. In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. 2018.

[90] Stephen Merity et al. “Pointer Sentinel Mixture Models”. In: *ArXiv* abs/1609.07843 (2016).

[91] Sabrina J. Mielke et al. *Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP*. 2021.

[92] Tsvetomila Mihaylova and André F. T. Martins. “Scheduled Sampling for Transformers”. In: *ArXiv* abs/1906.07651 (2019).

[93] Tomáš Mikolov et al. “Recurrent neural network based language model”. In: *Eleventh Annual Conference of the International Speech Communication Association*. 2010.

[94] Alexander H Miller et al. “ParlAI: A dialog research software platform”. In: *arXiv preprint arXiv:1705.06476* (2017).

[95] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

[96] Mathias Müller and Rico Sennrich. *Understanding the Properties of Minimum Bayes Risk Decoding in Neural Machine Translation*. 2021.

[97] Kenton Murray and David Chiang. “Correcting Length Bias in Neural Machine Translation”. In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 212–223. DOI: 10.18653/v1/W18-6322.

[98] Benjamin Newman et al. *The EOS Decision and Length Extrapolation*. 2020.

[99] Nathan Ng et al. “Facebook FAIR’s WMT19 News Translation Task Submission”. In: *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)* (2019). DOI: 10.18653/v1/w19-5333.

[100] Toan Q. Nguyen, Kenton Murray, and David Chiang. *Data Augmentation by Concatenation for Low-Resource Translation: A Mystery and a Solution*. 2021.

[101] Yixin Nie et al. “Addressing Contradictions in Dialogue Modeling”. In: 2021.

[102] Franz Josef Och. “Minimum error rate training in statistical machine translation”. In: *Proceedings of the 41st annual meeting of the Association for Computational Linguistics*. 2003, pp. 160–167.

[103] Franz Josef Och and Hermann Ney. “The alignment template approach to statistical machine translation”. In: *Computational linguistics* 30.4 (2004), pp. 417–449.

[104] Myle Ott et al. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: *Proceedings of NAACL-HLT 2019: Demonstrations*. 2019.

[105] Myle Ott et al. “Scaling Neural Machine Translation”. In: *Proceedings of the Third Conference on Machine Translation: Research Papers* (2018). DOI: 10.18653/v1/w18-6301.

[106] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *ACL*. 2002.

[107] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[108] Romain Paulus, Caiming Xiong, and Richard Socher. “A deep reinforced model for abstractive summarization”. In: *arXiv preprint arXiv:1705.04304* (2017).

[109] Jeffrey Pennington, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[110] Ben Peters and André FT Martins. “Smoothing and Shrinking the Sparse Seq2Seq Search Space”. In: *arXiv preprint arXiv:2103.10291* (2021).

[111] Ben Peters, Vlad Niculae, and André F. T. Martins. *Sparse Sequence-to-Sequence Models*. 2019.

[112] Krishna Pillutla et al. *MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers*. 2021.

[113] Yuval Pinter. “Integrating Approaches to Word Representation”. In: *ArXiv* abs/2109.04876 (2021).

[114] Lutz Prechelt. “Early Stopping - but when?” In: *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*. Springer-Verlag, 1997, pp. 55–69.

[115] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[116] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: 2019.

[117] Marc’Aurelio Ranzato et al. “Sequence Level Training with Recurrent Neural Networks”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016.

[118] Ricardo Rei et al. "COMET: A Neural Framework for MT Evaluation". In: *EMNLP*. 2020.

[119] Herbert E. Robbins. "A Stochastic Approximation Method". In: *Annals of Mathematical Statistics* 22 (2007), pp. 400–407.

[120] Stephen Roller et al. "Recipes for Building an Open-Domain Chatbot". In: *EACL*. 2021.

[121] Stephen Roller et al. "Recipes for building an open-domain chatbot". In: *arXiv preprint arXiv:2004.13637* (2020).

[122] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 627–635.

[123] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[124] João Sedoc and Lyle H. Ungar. "Item Response Theory for Efficient Human Evaluation of Chatbots". In: *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems* (2020).

[125] Abigail See et al. *Do Massively Pretrained Language Models Make Better Storytellers?* 2019.

[126] Abigail See et al. "What makes a good conversation? How controllable attributes affect human judgments". 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 1702–1723. DOI: 10.18653/v1/N19-1170.

[127] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. "BLEURT: Learning Robust Metrics for Text Generation". In: *ACL*. 2020.

[128] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.

[129] Iulian Vlad Serban et al. "A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues." In: *AAAI*. 2017, pp. 3295–3301.

[130] Iulian Vlad Serban et al. "A survey of available corpora for building data-driven dialogue systems". In: *arXiv preprint arXiv:1512.05742* (2015).

[131] Iulian Vlad Serban et al. "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models." In: *AAAI*. Vol. 16. 2016, pp. 3776–3784.

[132] Shiqi Shen et al. "Minimum risk training for neural machine translation". In: *arXiv preprint arXiv:1512.02433* (2015).

[133] Xiaoyu Shen et al. "Improving Variational Encoder-Decoders in Dialogue Generation". In: *arXiv preprint arXiv:1802.02032* (2018).

[134] Xing Shi, Yijun Xiao, and Kevin Knight. *Why Neural Machine Translation Prefers Empty Outputs*. 2020.

[135] Kurt Shuster et al. "Am I Me or You? State-of-the-Art Dialogue Models Cannot Maintain an Identity". In: 2021.

[136] Pavel Sountsov and Sunita Sarawagi. "Length bias in Encoder Decoder Models and a Case for Global Conditioning". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1516–1525. DOI: 10.18653/v1/D16-1158.

[137] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[138] Felix Stahlberg and Bill Byrne. "On NMT Search Errors and Model Errors: Cat Got Your Tongue?" 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3354–3360. DOI: 10.18653/v1/D19-1331.

[139] Yixuan Su et al. "Non-Autoregressive Text Generation with Pre-trained Language Models". In: *EACL*. 2021.

[140] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. "End-to-end memory networks". In: *Advances in neural information processing systems*. 2015, pp. 2440–2448.

[141] Ilya Sutskever, James Martens, and Geoffrey Hinton. "Generating text with recurrent neural networks". In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*. 2011. ISBN: 9781450306195.

[142] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[143] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017.

[144] Jesse Vig. "Deconstructing bert: Distilling 6 patterns from 100 million parameters." In: *Medium* (2018).

[145] Ashwin K Vijayakumar et al. "Diverse Beam Search for Improved Description of Complex Scenes." In: *AAAI*. 2018.

[146] Oriol Vinyals, Google Quoc, and V Le. "A Neural Conversational Model". In: *ICML Deep Learning Workshop*. 2015.

[147] Changhan Wang, Kyunghyun Cho, and Jiatao Gu. “Neural machine translation with byte-level subwords”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 9154–9160.

[148] Sean Welleck et al. “Consistency of a Recurrent Language Model With Respect to Incomplete Decoding”. In: *arXiv preprint arXiv:2002.02492* (2020).

[149] Sean Welleck et al. “Dialogue Natural Language Inference”. In: *ACL*. 2019.

[150] Sean Welleck et al. “Neural Text Generation With Unlikelihood Training”. In: *International Conference on Learning Representations*. 2020.

[151] Sean Welleck et al. *Non-Monotonic Sequential Text Generation*. 2019.

[152] Jason Weston, Emily Dinan, and Alexander H Miller. “Retrieve and Refine: Improved Sequence Generation Models For Dialogue”. In: *arXiv preprint arXiv:1808.04776* (2018).

[153] Thomas Wolf et al. “Transformers: State-of-the-art Natural Language Processing”. In: *arXiv preprint arXiv:1910.03771* (2019).

[154] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[155] Zhen Xu et al. “Neural Response Generation via GAN with an Approximate Embedding Layer”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 617–626. DOI: 10.18653/v1/D17-1065.

[156] Linting Xue et al. “ByT5: Towards a token-free future with pre-trained byte-to-byte models”. In: *ArXiv* abs/2105.13626 (2021).

[157] Yinfei Yang et al. “Learning Semantic Textual Similarity from Conversations”. In: *arXiv preprint arXiv:1804.07754* (2018).

[158] Yang You et al. “Large batch optimization for deep learning: Training bert in 76 minutes”. In: *arXiv preprint arXiv:1904.00962* (2019).

[159] Lantao Yu et al. “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”. In: *ArXiv* abs/1609.05473 (2016).

[160] Yury Zemlyanskiy and Fei Sha. “Aiming to Know You Better Perhaps Makes Me a More Engaging Dialogue Partner”. In: *arXiv preprint arXiv:1808.07104* (2018).

[161] Saizheng Zhang et al. “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, ACL*. 2018, pp. 2204–2213.

[162] Tianyi Zhang et al. “BERTScore: Evaluating Text Generation with BERT”. In: *ArXiv* abs/1904.09675 (2020).

[163] Yizhe Zhang et al. *DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation*. 2019.

[164] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. “Learning discourse-level diversity for neural dialog models using conditional variational autoencoders”. In: *arXiv preprint arXiv:1703.10960* (2017).

[165] Chunting Zhou et al. “Distributionally Robust Multilingual Machine Translation”. In: *EMNLP*. 2021.