A Plug and Play Approach to Age-Adaptive Dialogue Generation

Lennert Jansen
lennertjansen95@gmail.com
Working draft

Contents

1 Introduction

In recent years, we have witnessed promising advances in downstream natural language processing (NLP) tasks, such as language modeling, reading comprehension, machine translation, controllable text generation, and conversational response generation (**TODO: ADD REFERENCES TO SOME OF THESE ADVANCES**). Vaswani et al.'s Transformer architecture plays a central role in many of the state of the art (SotA) solutions to these problems, emerging as the succession to recurrent neural network (RNN) models for NLP tasks. Transformer-based language models (LMs) pre-trained on massive amounts of textual data, most famously OpenAI's GPT-2 [Radford et al., 2019], have demonstrated their usefulness for several of the aforementioned NLP tasks. For instance, controllable text generation and producing dialogue responses have improved greatly because of GPT-based hybrid models. Given the non-trivial requirements for sufficiently massive training corpora and computational resources, GPT-based solutions are worth considering for these two problems.

Controllable text generation (CTG) tries to enforce abstract properties, like writing style, on the passages being produced. Fine-tuning large-scale LMs for writing-style adaptation is extremely expensive, but Dathathri et al. [2020] and Li et al. [2020] propose methods that both excel at the task, while bypassing significant retraining costs. Dialogue response generation is the task of producing replies to a conversational agent's prompts, in a manner that is ideally both non-repetitive and relevant to the course of the conversation. With DialoGPT, Zhang et al. [2020] also manage to leverage GPT-2's powerful fluency for dialogue tasks, by framing them as language modeling tasks where multi-turn dialogue sessions are seen as long texts.

CTG and dialogue response generation share the overarching objective of producing grammatically correct text that is distinct from any training instance. A blend of these two tasks, i.e., controllable dialogue response generation, is an interesting and only partially explored route. It ties closely to one of Artificial Intelligence's long-standing goals of achieving human-like conversation with machines, as humans are known to adapt their language use to the characteristics of their interlocutor (**TODO: FIND A REFERENCE FOR THIS STATEMENT**). Adaptive dialogue generation is difficult due to the challenge of representing traits, like age or gender, via language expression, and the lack of persona trait labeled dialogue datasets. Zheng et al. [2019] explore the problem of personalized dialogue generation, and introduced `PersonalDialog` a large-scale multi-turn Chinese Mandarin dialogue dataset with personality trait labeling, and persona-aware adaptive dialogue generation models using RNNs and attention mechanisms.

1.1 Research Objectives

TODO: incorporate the hypotheses of every research question in this paragraph. In this thesis, I investigate the problem of controllable dialogue generation, with a focus on adapting responses to users' age. As a preliminary research objective, I aim to study to what extent a classifier can detect age-related linguistic differences in natural language And which features are most helpful in age-group detection? Do they (i.e., the linguistic or latent features exploited by the classifier) match the age-related informative features reported in previous work? After empirically confirming that speaker age detection is possible, I explore whether large-scale LMs, e.g. GPT-2, can be leveraged for text generation, controlled for age-groups. And what role does the used data play in the differences in output and performance between regular GPT-2 and controllable GPT-2? Finally, and most importantly, my research focuses on the degree to which such a CTG model is successful in generating dialogue that is adaptive w.r.t. age, such that it has a detectable effect on the perception of the user.

1.2 Contributions

TODO: contribute something to the field. No pressure.

The main contributions of this paper are ...

The remainder of this work is structured as follows, ...

2 Literature review

This is where you position your research in the relevant background and related research.

2.1 Background

What is the relevant theoretical background material to cover?

- Language modelling.
- (Controllable) Text Generation.

- GPT-x.
- Transformers.
- Dialogue Modelling.
  - Adaptive dialogue systems.
- Age and language.
  - The theoretical relationship between age and language.
  - Age(-group) detection from text.

Writing goals of this subsection:

- Position my research (i.e., dialogue response generation that is adaptive to age-groups) in its relevant background.
- Explain concepts that must be understood to grasp my research topic (i.e., give theoretical background information for the rest of your thesis).

Controllable dialogue generation encompasses several concepts in natural language processing and linguistics that must be understood to approach the problem. This subsection highlights these topics and positions the central problem of this thesis in its relevant theoretical background.

Language Models Language modelling is central to many NLP tasks. A language model is a probability distribution over words in a sentence or document. They are trained to predict the probability of the next word in an sentence, given the preceding sequence of words. The language modelling task is formulated as an unsupervised distribution estimation problem of datapoints $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ (e.g., documents), each representing sequences (of e.g., symbols or tokens) of varying lengths $(s_{i,1}, ..., s_{i,n}), i \in \{1, ..., N\}$. Note that N denotes the corpus size, and n the sequence length of datapoint i. To avoid cluttered notation, the subscript i will sometimes be omitted when discussing an arbitrary datapoint. The probability distribution over an observation $\mathbf{x}$ (i.e., the joint probability of an ordered sequence) can then be factorised as the product of its constituent conditionals [Radford et al., 2019]:

$$p(\mathbf{x}) = \prod_{j=1}^{n} p(s_j | s_1, ..., s_{j-1}). \tag{1}$$

This task allows language models to detect and learn patterns in language. The learned representations of these patterns can then be used for a plethora of applications, such as classification, and text generation. Moreover, this results in a framework for tractable sampling from the unconditional language model $p(\mathbf{x})$. $p(\mathbf{x})$ can therefore be seen a base generative model that can generate sample sentences [Dathathri et al., 2020].

In recent years, the attention-based models, Transformers [Vaswani et al., 2017], have replaced recurrent neural networks (RNNs) as the dominant architecture for LMs, with major improvements in distribution estimation, long-range dependency handling, and sample diversity.

Another recent development in language modelling is that of pre-training LMs on massive corpora. So-called large-scale general purpose LMs have demonstrated significant improvements in downstream tasks (i.e., other NLP tasks for which the model was not specifically trained or fine-tuned). Most famously the OpenAI's series of Generative Pre-trained Transformer (GPT) models have improved numerous NLP benchmarks [Radford et al., 2018, 2019, Brown et al., 2020].

- What are language models? / What is language modelling?
- Why is it important for many NLP tasks?
- How is it typically formulated or approached (very brief mathematical explanation)?
- Why is it important (to understand this) for my research?
  - Tractable sampling from the joint $\rightarrow$ (text) generation.
- Introduce pre-trained language models.
- Briefly introduce the Transformer (but not in too much technical detail, as it could be necessary to explain it in Methods).
- Same for GPT-2.

(Controllable) Text Generation

- Describe the text generation task.
- What is controllability in text generation? Why is it challenging?
- One of the main challenges in CTG using large-scale LMs is the cost of fine-tuning. Expand on this. How can this be solved?

In text generation, a LM $p(\mathbf{x})$ is asked to produce text $\mathbf{x}$ given a primer text by using the language model to sample from the distribution of words that are assigned the highest likelihood of following the primer text.

Text generation in itself is the task of generating a piece of text given a primer/input text. This process can be seen as sampling from a conditional distribution. Controllable text generation refers to the more restrictive problem of enforcing higher-level linguistic features on the generated text during sampling. This can be seen as a sub-problem of vanilla text generation, because the conditioning factor for the produced test is further constrained to also include some predefined textual attribute. This attribute can be many things: sentiment, topic, writing style, etc.

- base language model $p(\mathbf{x})$
- attribute model $p(a|\mathbf{x})$
- CTG model $p(\mathbf{x}|a) \propto p(a|\mathbf{x})p(\mathbf{x})$

Controllable text generation or CTG is a more challenging problem than vanilla text generation for a number of reasons. First, defining the desired attribute to be controlled for in a manner that is intelligible for a machine is a challenge in self. Second, like many and NLP problems, there do not exist many parallel corpora for which accurate attribute labelling is available. Furthermore, the measure of attribute coherence is a very vague and ambiguous concept. Namely, a text can be written in an extremely positive sentiment in multiple formulations, all of which adhere to the positive sentiment. That is, evaluating the level of attribute coherence is challenging. Another important hurdle for controllable text generation, especially when CTG is combined to leverage the linguistic power of large-scale language models, is that the cost of fine-tuning or pretraining a model to control for a linguistic attribute can be very high. PPLM is an example of a recent work that has the primary focus of leveraging powerful large-scale language models and making them controllable for a wide variety of linguistic attributes, all while avoiding incurring significant costs of fine-tuning. Nevertheless avoiding these costs is anything but trivial.

The plug and play set up of the PPLM model forms one of the main theoretical foundations of this work. It is both logical and promising for every day engineers to be able to leverage the grammatical fluency of pre-trained language models for more discriminative specific downstream tasks, e.g., specifying linguistic characteristic to enforce on an automatically written text. Their setup consist of a symbiosis of GPT-2 as their powerful large-scale language model (of course any large-scale generative language model can be used and the set up is not limited to a transformer based model see OPTIMUS reference), and a significantly smaller and therefore easier to train and fine-tune attribute model. This attribute model is often a small classifier or discriminator model, and can range in complexity from e.g. a simple bag of words model with a logistic classifier to a more complicated transformer encoder head. The main benefit of this setup is the extensibility it brings with it. Namely, such large-scale language models are open-source and available online and can be tailored to their specific needs using a significantly easier to train attribute model of your own liking.

They demonstrate the applicability of their model by beating numerous relevant state-of-the-art results as well as showing its applicability on a wide variety of tasks ranging from text style transfer to language detoxification (all of which can be seen as sub problems of controllable text generation).

This also has a wide variety of applications in the real world, for instance, being able to automatically re-write or adjust a draft text for an editorial, automatic generation of brand specific vacancy ads, or personalised chatbot assistance or even personalised education.

What this work also provides is a starting point for new applications, namely controllable dialogue generation.

Transformers The transformer architecture in recent years has dominated numerous NLP tasks and has for the basis for many of the state of the art architecture is in natural language processing. Its masked self-attention and compatibility with parallel processing have made it both effective and handling long-range dependencies in texts, as well attractive in terms of training time. This ability to handle long-range dependencies is exploited in particular in the domain of pretraining large language models. Namely this allows for transformer models to be pre-trained by applying language modelling objectives to long stretches of texts that contain longer range dependencies compared to e.g. tweets or short reviews.

The transformer architecture mainly consists of a encoder and decoder structure. Where the encoder processes the embedded text inputs and produces a hidden state using self attention mechanisms and fully connected layers. This

hidden state is then passed to the decoder layer which Produces an output which contains a distribution over the vocabulary and can be used for next word prediction.

GPT Following the success of open AI's pre-trained transformer model they went on to produce a series of generative pre-train transformers that I have been trained in a unsupervised language modelling session.

Dialogue Response Generation **TODO: add references to eg dialogpt** The domain of text generation encapsulates a number of sub-tasks, e.g., machine translation, abstractive summarisation, and paraphrasing. Dialogue response generation is also a special case of language generation. It can be seen as language generation where the prompt is a turn in a dialogue session. Conversational response generation shares open-domain text generation's overarching objective of producing grammatically correct fluent text that is distinct from any training instance, while remaining relevant to the prompt. However, computational dialogue modelling distinguishes itself from most NLP domains due to the challenges associated with modelling human conversation: informal, noisy, unstructured, and even erroneous real-world responses, possibly competing goals of interlocutors, or an inherently more diverse set of acceptable responses. The last point also emphasises that neural response generation has a much more 1-*to-many* nature than most text generation tasks.

Despite these differences, conversational response generation can be modelled in similar ways to open-domain text generation. Zeng et al. [2020] suggest to either formulate it in terms of source-target pairs, much like neural machine translation, or as a language modelling objective, where the next token or utterance is conditioned on the dialogue history. To remain close to the training objectives of my baseline models (GPT-2 and DialoGPT), I choose to adopt the language modelling formulation for conversation modelling. I.e., concatenate all dialogue turns in a multi-turn dialogue session into a long text: $x_1, ..., x_N$. Denote the source sentence or dialogue history as $S = x_1, ..., x_m$ and the target sentence (ground truth response) as $T = x_{m+1}, ..., x_N$. The conditional probability of dialogue continuation given its history $P(T|S)$ can be written as

$$p(T|S) = \prod_{n=m+1}^{N} p(x_n|x_1, ..., x_{n-1}). \tag{2}$$

A multi-turn dialogue session $T_1, ..., T_K$ can be written as $p(T_K, ..., T_2|T_1)$ which is essentially the product of all source-target pairs probabilities $p(T_i|T_1, ..., T_{i-1})$. This formulation also shows that optimising the single objective $p(T_K, ..., T_2|T_1)$ is equivalent to optimising all source-target pair probabilities.

- What is (computational) dialogue modeling?
- What do we aim to understand with CDM?
- How is (controllable) text generation related to CDM?
- Is dialogue generation "simply" a special case of text generation? If so, explain what are the conditions that constitute this special case.

Controllable dialogue generation Endowing a dialogue system with personality traits to generate human-like conversation is a long-standing goal in AI. This objective is difficult to reach because of the challenge of representing personality traits via language expression and the lack of large-scale persona-labeled dialogue datasets [Zheng et al., 2019].

Assuming an encoder-decoder setup, Zheng et al. [2019] argue that most personalized neural conversation models can be classified as one of two types: implicit and explicit personalisation models. For implicit personalization models, each speaker has its own vector representation, which implicitly captures the speaking style of the speaker in the decoding process. These models enjoy the benefit of having a more granular and realistic representation of speaking style, as opposed to a simple discrete set of traits (as is the case for explicit personalization models). On the other hand, it is unclear how speaker style is captured and should be interpreted, as all the information about a speaker's style is encoded in a real-valued vector. Furthermore, these methods suffer from a data sparsity issue, because each dialogue should be tagged with a speaker identifier and there should be sufficient dialogues from each trait-group to train a reliable trait-adaptive model. This last drawback is a bigger hurdle for the Zheng et al. [2019] than it is for mine, as their work deals with personalization for intersections of multiple traits, whereas my work focuses on adaptation to a small number of age groups. (**Todo: or should this sentence be in Related Work?**)

For explicit personalization models, the generated responses are conditioned either on a given personal profile, text-described persona, or simply an attribute label. I.e., speaker traits are represented as key-value pairs or descriptions about age, gender, etc. So this can be seen as conditioning the decoder's output on an attribute a, much like the PPLM setup of Dathathri et al. [2020]. Speakers with same set of personality traits can share attribute representations. So it does not require a speaker-specific representation vector. Such structured character descriptions are more explicit,

straight-forward, and interpretable. However, explicit personalization models require manually labeled or crowdsourced datasets for development, making it difficult to scale these models to large-scale dialogue datasets.

Language and speaker age.

- What is known about the relationship between a speaker's (or author's) age and their linguistic characteristics? I.e., how does language use develop with age according to the existing literature?
- How can we automatically detect a speaker's age(-group) from their utterances using machine learning?

2.2 Related work

Controllable language generation

- PPLM Dathathri et al. [2020]
- Optimus [Li et al., 2020]

Dathathri et al. [2020] Achieve controllable language generation using a plug and play model set up.Their architecture uses GPT to as their base language model which provides grammatical fluency, combined with a significantly easier to train attribute model (i.e., a simple BoW or single-layer classifier). Using gradient updates from the much smaller attribute model that are back propagated through the large pre-trained bass language model they manage to generate language combines (some of) the fluency of GPT-2 with the stylistic control of the attribute model, without the cost of retraining a specialised architecture.

They demonstrate desirable fluency as measured by perplexity with respect to GPT as well as measurable attribute control. Their architecture is applied to among other Tasks controlled story writing and language detoxification. They also show a clear trade-off between attribute control and grammatical correctness and diversity.

Dialogue generation is not explored as an application of PPLM, nor is their tested with more complex attribute models to hopefully allow for less deterioration of fluency as attribute control increases.

OPTIMUS

Writing-style Transfer

- Dai et al. [2019]

Writing-style transfer is a closely related problem to controllable language generation. It involves rewriting an input text with a different style of writing than it originally had. More formally, given a text $\mathbf{x}$, its corresponding style $\mathbf{s}^{(i)}$, the number of different styles K over which there exists a distribution, and a desired style $\hat{\mathbf{s}} \in \{\mathbf{s}^{(i)}\}_{i=1}^{K}$, the goal of style transfer is to rewrite the input sentence to $\hat{\mathbf{x}}$ with style $\hat{\mathbf{s}}$, while preserving the information content of $\mathbf{x}$. Using a trained model to rewrite the positively formulated input sentence "*I like fish sticks*" as it negative equivalent "*I do not like fish sticks*" is an example of writing-style transfer. Writing-style transfer can also be seen as a special case of (abstractive) summarization, for which Transformers have also demonstrated applicability [Baan et al., 2019].

? introduce the Style Transformer, an alternative to the previous RNN-based encoder-decoder frameworks for text style transfer. In previous work, neural text style transfer was done by passing input text through an encoder, yielding a style-dependent latent representation z. These previous approaches then attempt to "disentangle" z into a style-independent content representation and a latent representation of the stylistic properties of the input text. The following decoder then receives the content representation and a new latent style variable as input, to ultimately produce a style-altered output text with unchanged content.

1. It is difficult to evaluate the quality of disentanglement of the latent space.
2. Disentanglement is unnecessary, as contemporary work by **?** has shown a good decoder can perform controllable text generation from an entangled latent representation by "overwriting" the original style.
3. It is hard to capture rich semantic information in the latent representation due to limited capacity of vector representations (especially for long texts).
4. To disentangle style and content in the latent representations, all previous approaches have to assume all input texts can be encoded by a fixed-size latent vector.
5. Since most previous approaches use RNN-based encoder-decoder frameworks, they have problems capturing long-range dependencies in the input sentences.

These aforementioned problems are addressed by Dai et al. [2019] using Transformers [Vaswani et al., 2017] as the building block for text style transfer. The authors' approach does not require any manipulation (i.e., disentanglement) of the latent space, eliminates the need for a fixed-size vector representation of the input, and handles long-range dependencies better due to Transformers' attention mechanism. Aside from this being the first application of Transformers for text style transfer, the work contributes a novel training algorithm for such models, and boasts significant improvements of results on two text style transfer datasets.

Dialogue Generation

- DialoGPT

In Zhang et al. [2020], the authors introduce DialoGPT, a tunable large-scale language model for generation of conversational responses, trained on Reddit discussion chain data. DialoGPT therefore extends GPT-2 Radford et al. [2019] to address a more restrictive sub-category of text generation, i.e., conversational response generation.

DialoGPT inherits from GPT-2 a 12-to-48 layer transformer with layer normalisation, a custom initialisation scheme that accounts for model depth, and byte pair encodings Sennrich et al. [2016] as a tokeniser.

The generation task remains framed as language modelling, where a multi-turn dialogue session is modelled as a long text.

To address the well-known problem of open-domain text generation models producing bland and uninformative samples, Zhang et al. implement a maximum mutual information (MMI) scoring function. MMI uses a pre-trained backward model to predict $P(\text{Source}|\text{target})$: i.e., the source sentences (dialogue history) given the target (responses, dialogue continuation).

First, top-K sampling is used to generate a set of hypotheses. Then the probability $P(\text{Source}|\text{Hypothesis})$ is used to re-rank all hypotheses. As frequent and repetitive hypotheses can be associated with many possible queries/sources (i.e., a hypothesis that frequently occurs is one that is apparently applicable to many queries), and maximising backward model likelihood penalises repetitive hypotheses, MMI yields a lower probability for any specific query, thereby reducing blandness and promoting diversity.

DialoGPT is evaluated on the Dialog System Technology Challenge (DSTC) 7 track; an end-to-end conversational modelling task in which the goal is to generate conversation responses that go beyond chitchat by injecting information that is grounded in external knowledge. The model achieves state-of-the-art results on both the human and automatic evaluation results, by achieving near human-like responses that are diverse, relevant to the prompt, much like GPT-2 for open-domain language generation. They train 3 models of parameter sizes 117M, 345M, and 762M. The medium-sized 345M model achieves the best automatic evaluation results across most metrics, and is used as a baseline in later experiments in this work. Their Hugging Face PyTorch implementation can be tested here: https://huggingface.co/microsoft/DialoGPT-medium.

Adaptive Dialogue Generation

- `PersonalDialogue`
- MedDialog(?)

Zheng et al. [2019] investigate the problem of incorporating explicit personal characteristics in dialogue generation to deliver personalised (i.e., adaptive) conversation. They introduce `PersonalDialog`, a large-scale multi-turn dialogue dataset with personality trait labelling (i.e., `Age`, `Gender`, `Location`, `Interest Tags`, etc.) for a large number of speakers. They propose persona-aware adaptive dialogue generation models within the sequence-to-sequence learning framework. During the decoding process, they suggest two novel techniques: *persona-aware attention* and *persona-aware bias*.

Plug-and-play language and dialogue generation

3 Methodology

3.1 Transformers

The Transformer architecture plays a central role in most of the recent advances in NLP. The same holds for the methods used in this thesis to investigate controlled dialogue generation and speaker/author age detection. A brief explanation about the Transformer therefore in order. For a more detailed review of the model architecture, the reader is referred to the original paper ([Vaswani et al., 2017]) or this excellent blog post: `https://jalammar.github.io/illustrated-transformer/`.

The Transformer, like most neural sequence processing models, has an encoder-decoder structure. On a high level, the encoder receives an input sequence $\mathbf{x} = (x_1, ..., x_n)$ (e.g., a sentence), and maps this to a sequence of latent continuous variables $\mathbf{z} = (z_1, ..., z_n)$. The decoder then takes $\mathbf{z}$ as input, and maps this to an output sequence $\mathbf{y} = (y_1, ..., y_m)$. Note that the use of positional encodings of the input and output embeddings enables the Transformer to process and generate sequences in arbitrary order, allowing for a high degree of parallelization. The generation of $\mathbf{y}$ happens element-by-element in an auto-regressive fashion, where at step t, element y_{t-1} is also taken as input.

Both the encoder and decoder are comprised of N identical layers (denoted by the 'N $\times$' in the left part of Figure 1). Every sub-layer performs a succession of transformations using multi-head self-attention mechanisms and point-wise, fully connected layers, along with residual connections [He et al., 2016] around every sub-layer followed by layer normalization [Ba et al., 2016]. The decoder's first self-attention sub-layer is masked to ensure that the output predictions at sequence position i cannot depend on output positions greater than i. Finally, the decoder passes its output through a linear and softmax layer to produce a probability distribution over the problem space (e.g., the vocabulary) from which the most likely symbols for the generated output sequence $\mathbf{y}$ can be sampled.

A key aspect of the Transformer architecture is its use of attention [Bahdanau et al., 2015]. This allows the encoder-decoder architecture to selectively focus on parts of the input sequence to produce a more informative hidden representation. Vaswani et al. formulate an attention function as a mapping of queries and sets of key-value pairs to an attention output, where matrices represent the queries Q, keys K, and values V. The attention output is a weighted sum of the values, based on the relevance of the corresponding keys to a query. In particular, they employ scaled dot-product attention:

$$\texttt{Attention}(Q, K, V) = \texttt{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3}$$

Furthermore, Vaswani et al. propose to use multi-head attention by using learned linear projections to project the queries, keys and values h times, and apply the aforementioned attention function to these projections in parallel. The concatenation of these attention outputs, passed through a linear layer, ultimately produces the final output of the Transformer's attention sub-layers. This allows the model to attend to the relevant information from all representation sub-spaces at various sequence positions. See Figure 1 for an schematic illustration of the Transformer's structure described above.

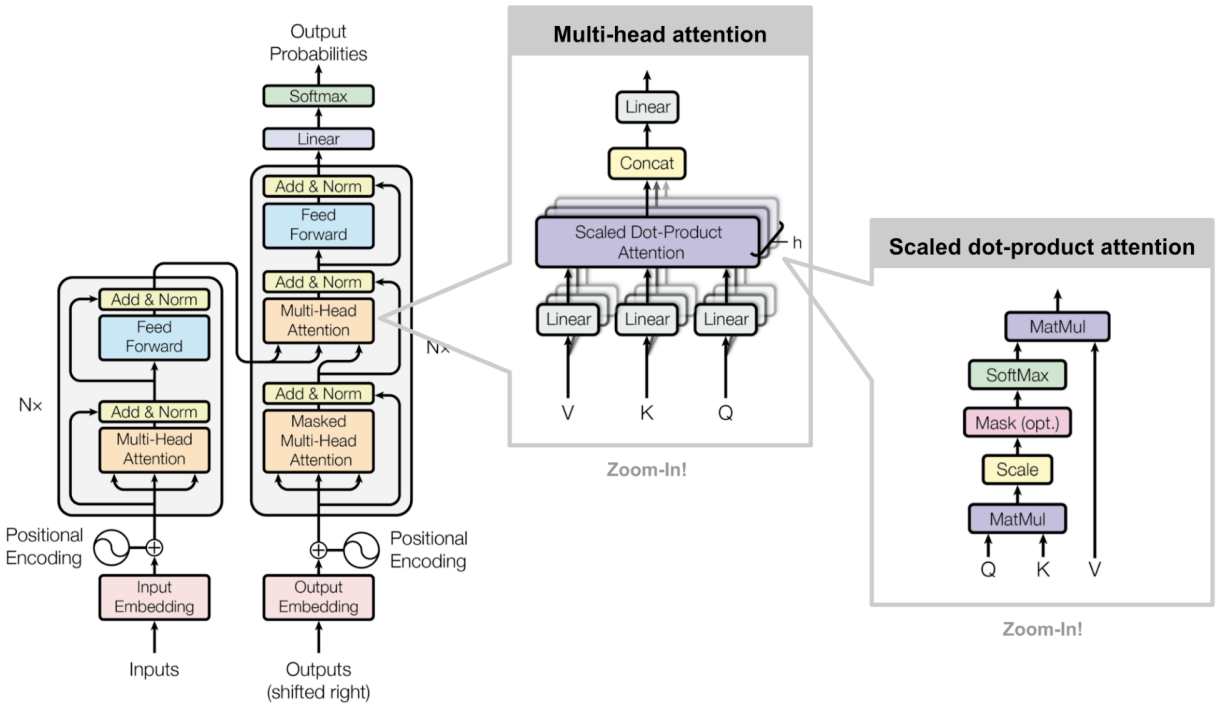

Figure 1: An overview of the full Transformer model architecture. *Collated image source:* Fig. 17 in this blog post `https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html`. *Original image source:* Figures 1 and 2 in Vaswani et al. [2017]

3.2 Causal language modeling with Transformers

Following the conventions of Dathathri et al. [2020] and Madotto et al. [2020], a dialogue is comprised of multiple alternating turns (sometimes referred to as utterances) between more than one speaker. For simplicity, this project only focuses on dialogues between two speakers. The conversation history at turn t is defined as $\mathcal{D}_t = \{S_1^{(1)}, S_1^{(2)}, ..., S_t^{(1)}\}$, where $S_t^{(j)}$ is speaker j's utterance at time t. Madotto et al. [2020] denote speaker 1 as the user U, and speaker 2 as the conversational system S, yielding dialogue history $\mathcal{D}_t = \{U_1, S_1, ..., U_t\}$. This notational convention will also be used for the user-system experiments later on in this report.

A Transformer-based language model (denoted `LM`) is used in this thesis to model the distribution of dialogues, using dialogue history at time t, $\mathcal{D}_t$, as a prompt to auto-regressively generate the dialogue continuation S_t. More specifically, let the concatenation of the dialogue history at t and its continuation, $\{\mathcal{D}_t, S_t\}$, be represented as a sequence of tokens $\mathbf{x} = \{x_0, ..., x_n\}$. Then, by recursively applying the product rule of probability (Bishop [2006]), the unconditional probability of the sequence $p(\mathbf{x})$ can be expressed as:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_0, ..., x_{i-1}). \tag{4}$$

Dathathri et al. [2020] and Madotto et al. [2020] define the Transformer's decoding process in a recursive fashion. Let H_t denote the conversation history's key-value pairs, i.e., $H_t = \left[(K_t^{(1)}, V_t^{(1)}), ..., (K_t^{(l)}, V_t^{(l)})\right]$, with $(K_t^{(i)}, V_t^{(i)})$ representing the key-value pairs from the `LM`'s i-th layer generated at all time steps 0 through t. This results in the recurrent dedocing process being expressed as:

$$o_{t+1}, H_{t+1} = \texttt{LM}\left(x_t, H_t\right), \tag{5}$$

where o_{t+1} is the hidden state of the last layer. Finally, after applying a softmax transformation, the next token x_{t+1} is sampled from the resulting probability distribution, i.e., $x_{t+1} \sim p_{t+1} = \texttt{softmax}\left(W o_{t+1}\right)$, where W is a linear mapping from the model's last hidden state to a vector of vocabulary size. This recursive formulation allows for efficient text generation by leveraging cached memories, without repeated forward passes.

3.3 Plug-and-play modeling

Plug-and-play language model (PPLM) Dathathri et al. [2020] works by using a text classifier, referred to as an attribute model, to control the text generated by a language model. Let $p(X)$ denote the distribution of a Transformer-based language model (e.g., GPT-2 or DialoGPT), where X represents the generated text. And $p(a|X)$ denotes the attribute model (e.g., a single-layer or BoW classifier) that represents the degree of adherence of text X to a certain attribute a (e.g., style, sentiment, or age-group characteristics). Then PPLM can be seen as modeling the conditional distribution of generated text X given attribute a, i.e., $p(X|a)$. Note that Bayes' theorem ties these three definitions together as follows:

$$p(X|a) \overbrace{=}^{\text{Bayes' theorem}} \frac{p(X)p(a|X)}{p(a)} \propto p(X)p(a|X). \tag{6}$$

To control the generated text, PPLM shifts the aforementioned history H_t (i.e., all Transformer key-value pairs generated up to time t) in the direction of the sum of two gradients:

1. Ascending $\nabla \log p(a|X)$: maximizing the log-likelihood of the desired attribute a under the conditional attribute model. This enforces attribute control.
2. Ascending $\nabla \log p(X)$: maximizing the log-likelihood of the generated language under the original (possibly conversational) language model. This promotes fluency of the generated text.

These two incentive-representing gradients are combined with various coefficients, yielding a set of tunable knobs to steer the generated text in the direction of the desired fluency, attribute control, and length.

Let's first focus on the first of the two gradients, i.e., the attribute control promoting $\nabla \log p(a|X)$. ΔH_t represents the update to history H_t that pushes the distribution of the generated text X in the direction that has a higher likelihood of adhering to desired attribute a. The gradient update rule can be expressed as:

$$\Delta H_t \leftarrow \Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)}{\|\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)\|^{\gamma}} \tag{7}$$

where α is the step size, and γ denotes the normalization term's scaling coefficient. Both step size (α) and the scaling coefficient (γ) influence attribute control. Attribute control can be softened by either decreasing α or increasing γ and vice versa. Note that $\alpha = 0$ recovers the original uncontrolled underlying language model (e.g., GPT-2 or DialoGPT). In practice, ΔH_t is initialized at zero, and the update rule in Equation 7 is applied m times (usually 3 to 10), resulting in the updated key-value pair history $\tilde{H}_t = H_t + \Delta H_t$. Then the updated history $\tilde{H}_t$ is passed through the language model, yielding the updated logits (final Transformer-layer): $\tilde{o}_{t+1}, H_t = \texttt{LM}(x_t, \tilde{H}_t)$. And finally the shifted $\tilde{o}_{t+1}$ is linearly mapped through a softmax layer to produce a new, more attribute-adherent, distribution from which to sample, i.e., $x_{t+1} \sim \tilde{p}_{t+1} = \texttt{softmax}\left(W\tilde{o}_{t+1}\right)$.

The method described until now will generate attribute-adherent text, but will likely yield fooling examples [Nguyen et al., 2015] that are gibberish to humans, but get assigned high $p(a|x)$ by the attribute model [Dathathri et al., 2020]. That is why Dathathri et al. [2020] apply two methods to ensure fluency of the generate text. The first is to update ΔH_t such to minimize the Kullback-Leibler (KL) divergence (denoted D_{KL}) between the shifted and original distributions. In practice, D_{KL} is scaled by a coefficient λ_{KL}, typically found to work well for most tasks when set to 0.01. Repetitive text generation (i.e., high $p(a|x)$ but low $p(x)$) can therefore sometimes be avoided by increasing λ_{KL}. The second method to ensure fluency is Post-norm Geometric Mean Fusion [Stahlberg et al., 2018] which, instead of directly influencing ΔH_t like minimizing D_{KL}, fuses the altered generative distribution $\tilde{p}_{t+1}$ with the unconditional language distribution $p(x)$. This is done during generation by sampling the next token as follows:

$$x_{t+1} \sim \frac{1}{\beta}\left(\tilde{p}_{t+1}^{\gamma_{gm}} p_{t+1}^{1-\gamma_{gm}}\right) \tag{8}$$

where β is a normalization constant, p_{t+1} and $\tilde{p}_{t+1}$ denote the original and modified distributions, respectively, and γ_{gn} is a scaling term that interpolates between the two distributions. Because the new sampling distribution in Equation 8 converges towards the unconditional language model as $\gamma_{gm} \to 0$, repetitive text generation can be avoided by decreasing the scaling term.

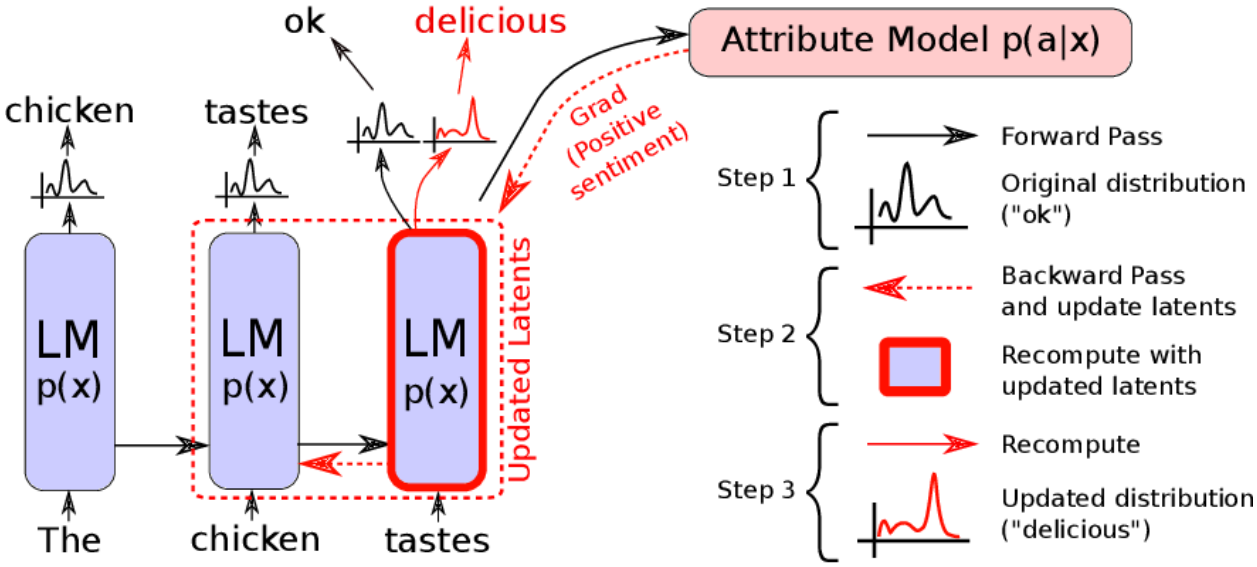

Figure 2: **TODO:**Caption... *Original image source:* Figure 1 of Dathathri et al. [2020]

It is important to realize that the plug-and-play method applied by Dathathri et al. [2020] and Madotto et al. [2020] is different from fine-tuning. Note that in Equation 7 the gradient updates are restricted to the history H_t, and do not affect the model's parameters. Because the key-value pairs $(K_t^{(i)}, V_t^{(i)})$ that comprise H_t are activations and not model-weights, the updates only take place in the activation-space. This means that PPLM leaves the underlying (conversational) language model untouched.

Contrary to fine-tuning often massive LMs, PPLM does not incur a significant training cost (depending of course on the complexity of the discriminator or attribute model). However, Madotto et al. [2020] show that PPLM needs a fixed number of m update-steps to for every generated token. This makes the original PPLM setup unsuitable for online interactive applications, like conversational systems. Addressing this problem, they introduce plug-and-play conversational models (PPCM), which extends PPLM by using the original model setup to generate dialogue datasets with the desired attribute a, and then use optimized residual adapters [Bapna and Firat, 2019] to control `LM`'s output distribution.

Residual adapters are optimizable modules stacked on every Transformer-layer of a pre-trained (language) model. The adapter module then steers the Transformer's output distribution without changing the pre-trained model's weights. A Layer Normalization module [Ba et al., 2016] followed by an auto-encoder with residual a connection constitutes a residual adapter module. More specifically, the residual adapter block can be expressed as the following function composition:

$$\begin{aligned} f_{\theta_i}(x) &= \texttt{ReLU}(\texttt{LayerNorm}(x) \cdot W_i^E) \cdot W_i^D, \\ \texttt{Adapter}(o_{:t}^i) &= f_{\theta_i}(o_{:t}^i) + o_{:t}^i \end{aligned} \tag{9}$$

where $o_{:t}^i \in \mathbb{R}^{t \times d}$ denotes the Transformer's i-th layer's latent output at step t, d is the hidden state's size, W_i^E and W_i^D are learnable parameter-matrices of sizes $d \times m$ and $m \times d$, respectively. Finally, m is the auto-encoder's bottle-neck dimension, which is a tunable hyper-parameter for changing the residual adapter's capacity. In practice, Madotto et al. [2020] use PPLM to generate n attribute-adherent dialogue datasets $\mathscr{D}^a = \{\mathcal{D}^1, ..., \mathcal{D}^n\}$, for attribute a. These generated dialogue datasets are then used to train the residual adapter, which they aptly name a plug-and-play adapter, so it can be used to control the language model's output distribution. So for every attribute a, they train the plug-and-play adapter's parameters $\Theta_a := \{\theta_0^a, ..., \theta_l^a\}$, where $\theta_i^a := \{W_i^{E,a}, W_i^{D,a}\}$, such that negative log-likelihood over the corresponding dialogue dataset $\mathscr{D}^a$ is minimized:

$$\Theta_a \text{ s.t. } \min \mathcal{L}(\mathscr{D}^a) = -\sum_{k}^{|\mathscr{D}^a|} \sum_{i}^{n} \log p(s_i^k | s_{<i}^k, \mathcal{D}_t^k), \tag{10}$$

where s_i^k is the i-th generated token of response $S_t^k = \{s_0^k, ..., s_n^k\}$ with maximum sequence length n.

4 Data and experimental setup

Two datasets are considered during the experiments of this thesis: (1) the spoken component of the British National Corpus (BNC or BNC2014) [Love et al., 2017], and (2) the Blog Authorship Corpus (BAC, or sometimes referred to as 'blog corpus') [Schler et al., 2006]. The first corpus is a collection of transcriptions of everyday conversations in British English, gathered between 2012 and 2016. The second is a dataset of blogs posted on `https://www.blogger.com`, gathered in or before August 2004.

The texts (i.e., blogs or dialogue transcriptions) in both corpora are labeled by age, among other labels. This makes them suitable candidates for training and testing age classifiers. In both cases, the texts are written in a somewhat informal manner, making them more representative of everyday speech. Only the BNC is used for the controllable dialogue generation phase of the experiments, as the BAC is not a conversational dataset. What follows is an overview of both datasets' motivation for use, drawbacks, metadata, descriptive statistics, and pre-processing steps before analyses.

4.1 The British National Corpus (BNC)

The conversations of the spoken component of the BNC were typically recorded at home and took place between friends and family of participants. Participants recorded their conversations using their smartphones, allowing for spontaneous recording. These two properties of the BNC's sampling procedure make the recorded dialogues representative of contemporary everyday British English speech.

A total of 1251 conversations of 672 speakers constitutes the full corpus, accounting for 10.4 million words. During the recruitment process of the BNC, prospective participants were asked to disclose personal information like age, gender, highest completed level of education, employment, and perceived accent[1]. Other metadata accompany the conversations themselves. Namely, number of speakers taking part in the conversation, ages of those speakers, conversation length, and topic.

[1]For a detailed description of the sampling decisions made during data collection, the reader is referred to the BNC2014 user manual: `http://corpora.lancs.ac.uk/bnc2014/doc/BNC2014manual.pdf`

Nine age-brackets are present in the corpus: 11-18, 19-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, and 90-99. To make the objective of learning to classify and generate conversational responses more straightforward, I only consider dialogues between two participants. Namely, it will be more difficult to distinguish which utterances constitute the relevant responses to something said in a conversation with more than two participants. This filter step results in a remaining dataset of 622 dialogues, almost 460K utterances (i.e., a single turn in a dialogue), and nearly 5M tokens. On average, a dialogue has 736 turns, meaning that the conversations are relatively long. Furthermore, due to minors not being allowed to participate in this project's interactive experiments later on, the age-bracket 11-18 is dropped. This leaves us with a dataset of 522 dialogues, 418K utterances, and 4.4M tokens. Finally, the age-brackets get regrouped into brackets 19-29 and 50 plus, and conversations with speakers aged 30-49 are removed. This gap between to two age-groups is made to minimize the chance of overlapping linguistic characteristics being present between separate age-brackets (e.g., the difference in language use between speakers aged 19-29 and 30-39 is probably smaller than that between age-brackets 19-29 and 50-59). Moreover, only conversations between two participants of the same (new regrouped) age-bracket is kept. This is done to avoid the confounding factor that interlocutors of a certain age-group might adjust their choice of words to the age of the person they are talking to. Only considering dialogues between participants of similar ages is expected to keep the utterances as representative of the age-bracket's linguistic characteristics as the corpus allows. These final filtration steps result in a subset consisting of 237 dialogues, roughly 172.000 utterances, and approximately 1.8M tokens.

- Talk about pre-processing steps.
- The main limitations of the BNC are its size and imbalance.
- Weighted loss and weighted random sampling
- Data outdated? Not representative of modern speech.

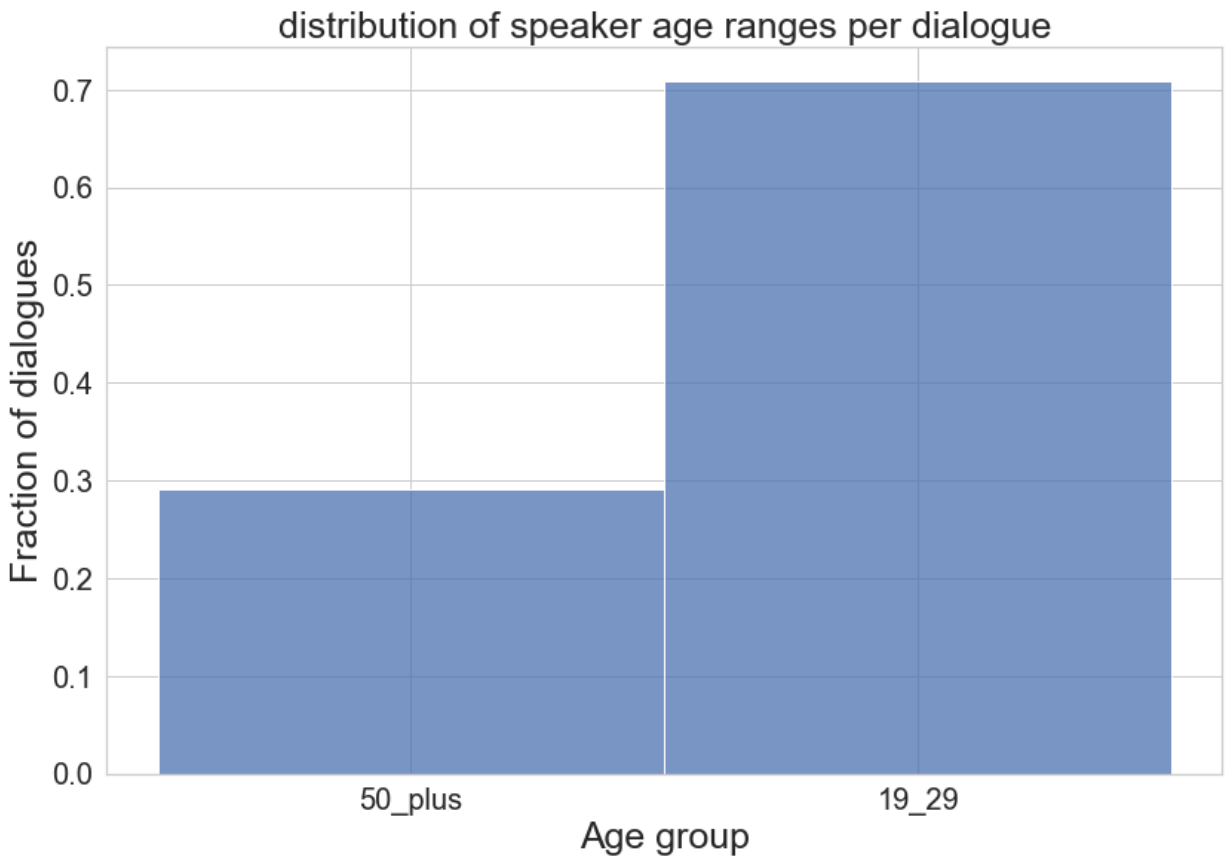

Figure 3: Distribution of age-brackets per dialogue in 19-29 vs. 50-plus subset of BNC.

4.2 The Blog Authorship Corpus (BAC)

Corpus	BNC	BAC
Dialogue?	Yes	No
No. words	$10.4 \cdot 10^6$	$140 \cdot 10^6$

Table 1: Summary of corpora properties.

5 Results

The experiments of this study are divided into two phases: (1) automated age detection from written texts or dialogue transcriptions, and (2) age-adaptive dialogue generation.

5.1 Automated age-detection from text

Three classes of architectures are trained for the task of predicting a writer's or speaker's age given a written text or speech transcription; (1) logistic n-gram models, (2) recurrent neural networks (RNNs), specifically, long short-term memory (LSTM), and (3) attention-based BERT sequence classifiers.

Model	**Accuracy** ↑ better	$\boldsymbol{F}_1^{(19-29)}$ ↑ better	$\boldsymbol{F}_1^{(50-plus)}$ ↑ better
Baseline (random guessing)	0.500 (0.00000)	0.500 (0.00000)	0.500 (0.00000)
unigram	0.702 (0.00574)	0.713 (0.00559)	0.690 (0.00642)
bigram	0.703 (0.00616)	0.713 (0.00459)	0.693 (0.00849)
trigram	0.709 (0.00683)	0.718 (0.00680)	0.700 (0.00766)
LSTM	0.696 (0.00452)	0.689 (0.01810)	**0.701** (0.01614)
BiLSTM	0.684 (0.00675)	0.688 (0.01791)	0.679 (0.01638)
BERT-base uncased	**0.710** (0.00757)	**0.723** (0.00477)	0.690 (0.01222)

Table 2: Balanced BNC age classifiers. **Test set** results averaged over 5 random initializations. Format: *average metric (standard error)*

Model	**Accuracy** ↑ better	$\boldsymbol{F}_1^{(13-17)}$ ↑ better	$\boldsymbol{F}_1^{(23-27)}$ ↑ better	$\boldsymbol{F}_1^{(33-plus)}$ ↑ better
Baseline (predict majority class)	0.470	...	...	...
unigram	0.603 (0.00143)	0.760 (0.00267)	0.706 (0.00075)	0.491 (0.00268)
bigram	0.627 (0.00072)	0.788 (0.00129)	0.715 (0.00130)	0.504 (0.00223)
trigram	0.625 (0.00201)	**0.789** (0.00136)	0.716 (0.00193)	0.485 (0.00280)
(2 layer bi-)LSTM	**0.719**	0.780	**0.742**	**0.509**
BERT	...	...	...	...

Table 3: Blog corpus age classifiers. **Test set** results averaged over 5 random initializations. Format: *average metric (standard error)*

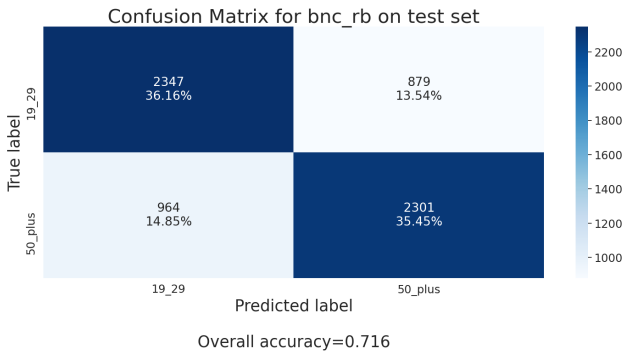

Figure 4: Confusion matrix BERT age classifier on balanced BNC **test** set.

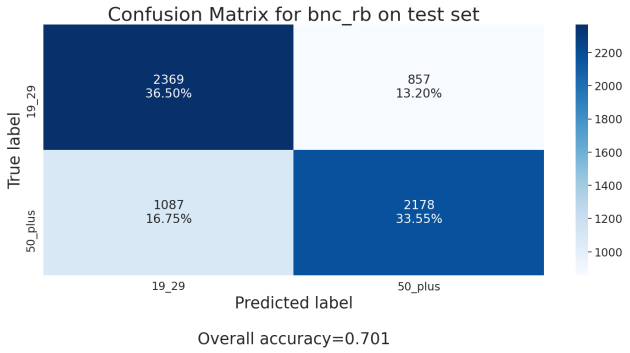

Figure 5: Confusion matrix LSTM age classifier on balanced BNC **test** set.

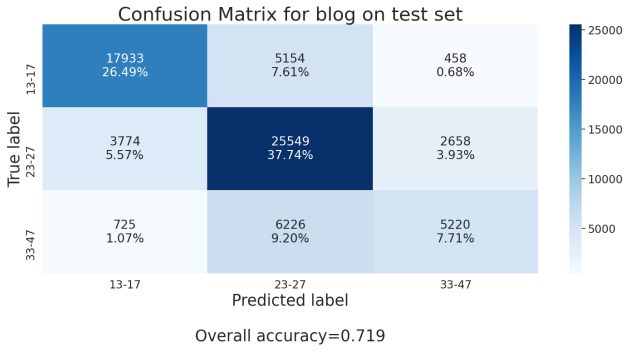

Figure 6: Confusion matrix bi-LSTM age classifier on blog corpus **test** set.

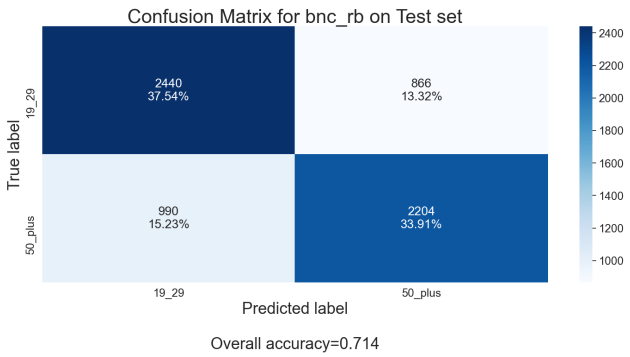

Figure 7: Confusion matrix for best trigram age classifier on **balanced** BNC **test** set.

Notes on the imbalanced BNC.

- Attempts were made to account for the original BNC's bias (i.e., the 19-29 age bracket accounts for roughly 80% of the total considered subset).
- Method 1: weighted loss.
- Method 2: weighted random sampling (i.e., up-sampling of the minority class).
- Weighted random sampling outperformed weighted loss in terms of validation accuracy and F_1 scores, but still failed to surpass the baseline.
- *In terms of test accuracy*, the n-gram models succeeded in beating the baseline (predicting the majority class), whereas the best LSTM and fine-tuned BERT-based failed to do so.
- However, the neural discriminators still outperformed all the other models with respect to minority class F_1 scores, indicating that (1) the n-gram models aren't very useful for correctly classifying the minority class, and that (2) weighted random sampling improved the models' efficacy with respect to the minority class.
- See Appendix A.1 for these results.

5.2 Controlled dialogue generation

Model settings to compare. **TODO:** Also add a representative model from Dathathri et al. [2020] as baseline.

- Uncontrolled language models:
  - **Baseine:** "Vanilla" uncontrolled GPT-2-medium
  - Uncontrolled GPT-2-large (?)
  - Uncontrolled DialoGPT-medium
  - Uncontrolled DialoGPT-large (?)
- Bag-of-words (BoW) controlled language models:
  - GPT-2-medium + Full BNC BoW
  - GPT-2-medium + 19-29 BNC BoW
  - GPT-2-medium + 50 plus BNC BoW
  - DialoGPT-medium + Full BNC BoW
  - DialoGPT-medium + 19-29 BNC BoW
  - DialoGPT-medium + 50 plus BNC BoW
- Discriminator-based controlled language models. **TODO:** Which age discriminator(s) should I consider? E.g., 1-layer linear, or Transformer-based discriminators.
  - GPT-2-medium + BNC age discriminator
  - DialoGPT-medium + BNC age discriminator

References

L. J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

J. Baan, M. ter Hoeve, M. van der Wees, A. Schuth, and M. de Rijke. Do transformer attention heads provide transparency in abstractive summarization? *CoRR*, abs/1907.00570, 2019. URL http://arxiv.org/abs/1907.00570.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

A. Bapna and O. Firat. Simple, scalable adaptation for neural machine translation. 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)*, pages 1538–1548, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1165. URL https://www.aclweb.org/anthology/D19-1165.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

N. Dai, J. Liang, X. Qiu, and X. Huang. Style transformer: Unpaired text style transfer without disentangled latent representation. *arXiv preprint arXiv:1905.05621*, 2019.

S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, and J. Gao. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.378. URL https://www.aclweb.org/anthology/2020.emnlp-main.378.

R. Love, C. Dembry, A. Hardie, V. Brezina, and T. McEnery. The spoken bnc2014: designing and building a spoken corpus of everyday conversations. *In International Journal of Corpus Linguistics*, 22(3):319–344, 2017.

A. Madotto, E. Ishii, Z. Lin, S. Dathathri, and P. Fung. Plug-and-play conversational models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2422–2433, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.219. URL https://www.aclweb.org/anthology/2020.findings-emnlp.219.

A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

J. Schler, M. Koppel, S. Argamon, and J. W. Pennebaker. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 6, pages 199–205, 2006.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

F. Stahlberg, J. Cross, and V. Stoyanov. Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 204–211, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6321. URL https://www.aclweb.org/anthology/W18-6321.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

G. Zeng, W. Yang, Z. Ju, Y. Yang, S. Wang, R. Zhang, M. Zhou, J. Zeng, X. Dong, R. Zhang, H. Fang, P. Zhu, S. Chen, and P. Xie. MedDialog: Large-scale medical dialogue datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9241–9250, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.743. URL https://www.aclweb.org/anthology/2020.emnlp-main.743.

Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL, system demonstration*, 2020.

Y. Zheng, G. Chen, M. Huang, S. Liu, and X. Zhu. Personalized dialogue generation with diversified traits. *arXiv preprint arXiv:1901.09672*, 2019.

A Appendix

A.1 Age discrimination on the imbalanced British National Corpus

Model	Accuracy	F_1 (19-29)	F_1 (50-plus)
Baseline (predict majority class)	0.807	...	0 (0)
unigram	0.833 (0.00241)	0.903 (0.00142)	0.373 (0.01198)
bigram	**0.840** (0.00101)	**0.907** (0.00050)	0.406 (0.00933)
trigram	0.838 (0.00151)	0.906 (0.00109)	0.403 (0.00731)
bi-LSTM	0.780 (...)	0.863 (...)	**0.440** (...)
BERT	0.786	...	...

Table 4: BNC age classifiers. Results averaged over 5 random initializations. Format: *average metric (standard error)*

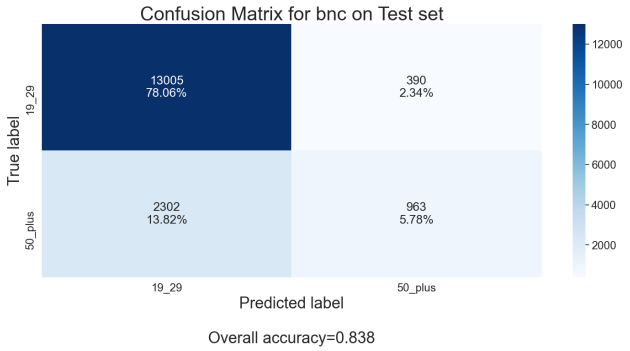

Figure 8: Confusion matrix for best bigram age classifier on BNC test set.

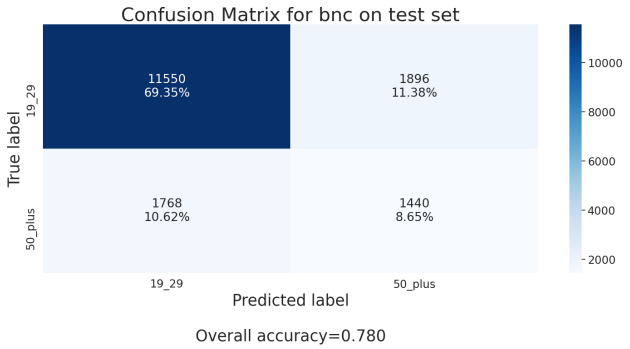

Figure 9: Confusion matrix bi-LSTM age classifier on BNC test set.

TODOs:

- Add validation set results to appendix for all datasets.