Voice Query Auto Completion

Raphael Tang,[1] **Karun Kumar,**[1] **Kendra Chalkley,**[*2] **Ji Xin,**[3] **Liming Zhang,**[1]
Wenyan Li,[*4] **Gefei Yang,**[1] **Yajie Mao,**[1] **Junho Shin,**[1] **G. Craig Murray,**[1] **Jimmy Lin**[3]

[1]Comcast AI [2]SparkCognition Government Systems [3]University of Waterloo

[1]firstname_lastname@comcast.com [2]kendra.chalkley@sparkgov.ai
[3]{ji.xin,jimmylin}@uwaterloo.ca [4]wenyanl62@gmail.com

Abstract

Query auto completion (QAC) is the task of predicting a search engine user's final query from their intermediate, incomplete query. In this paper, we extend QAC to the streaming voice search setting, where automatic speech recognition systems produce intermediate transcripts as users speak. Naïvely applying existing methods fails because the intermediate transcripts often don't form prefixes or even substrings of the final transcript. To address this issue, we propose to condition QAC approaches on intermediate transcripts to complete voice queries. We evaluate our models on a speech-enabled smart television with real-life voice search traffic, finding that this ASR-aware conditioning improves the completion quality. Our best method obtains an 18% relative improvement in mean reciprocal rank over previous methods.

1 Introduction

Query auto completion (QAC) is the task of predicting a user's complete query given the present, incomplete prefix of the query. For example, suppose a user types "COVID vaccine" into Google. Then, a QAC system proposes the most likely completions for that prefix, e.g., "COVID vaccine near me," saving the user time when the prediction is correct. Existing state-of-the-art approaches generate completions using language models conditioned on the prefix (Park and Chiba, 2017), with simple prefix trees serving as a strong baseline.

In the streaming voice search setting, such as on the Google Voice Assistant, naïvely adapting these existing approaches fails because the key assumptions differ. Most glaringly, incomplete queries comprise partial speech, not text. In place of human users, an automatic speech recognition (ASR) system produces the textual transcripts, resulting in intermediate queries that often *don't* form prefixes or even substrings of the final query. Consider the voice query "Hulu" as an example. Passing it through a streaming ASR system, we observe the transcripts "who," then "Hulu." Traditional QAC approaches fail to complete "Hulu" from "who," since they use orthographic prefixes and substrings (Cai and de Rijke, 2016) instead of the true phonetic prefix, which is generally unavailable at training time.

Nevertheless, the intermediate transcript "who" is still informative toward predicting "Hulu" because it frequently precedes "Hulu" in the sample. Based on this observation, we hypothesize that, for improved QAC quality, we must additionally model the dynamics between the intermediate and the final transcripts from the ASR system.

In this paper, we precisely design and evaluate ASR-aware QAC models. The main contributions of our work are as follows: First, we are the first to describe the task of QAC for streaming, bidirectional ASR systems in voice search. Second, we propose and evaluate novel, ASR system-aware QAC models for the task, showing that incorporating context from intermediate transcripts helps. On the Xfinity X1, a voice-enabled smart TV serving more than twenty million American customers, our best approach attains an 18% relative improvement in mean reciprocal rank over the previous best.

2 Voice Query Auto Completion

Our novel task is to predict the final voice queries that users issue, given some mid-utterance, intermediate transcripts of their incomplete speech from the ASR system. As is typical, these systems are streaming, with the speech being transcribed to text in real time. Concretely, for some utterance, we are given a k-tuple of string transcripts $\boldsymbol{X} := (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(k)})$ representing the streaming outputs of the ASR system across the utterance, where $\boldsymbol{x}^{(i)}$ is a string of words. We index $\boldsymbol{X}$ in chronological order, e.g., ("Who", "Hulu", "Hulu now") for the utterance "Hulu now." We wish

[*] Work done while employed at Comcast AI.

c	Input		Output
1	(“A”)	$\mapsto$	(“A [\$] A”)
1	(“A”, “B”, “C”)	$\mapsto$	(“A [\$] C”, “B [\$] C”, “C [\$] C”)
2	(“A”, “B”, “C”)	$\mapsto$	(“A [\$] C”, “A [|] B [\$] C”, “B [|] C [\$] C”, “C [\$] C”)
3	(“A”, “B”, “C”)	$\mapsto$	(“A [\$] C”, “A [|] B [\$] C”, “B [|] C [\$] C”, “C [\$] C”)
3	(“A”, “B”, “C”, “D”)	$\mapsto$	(“A [\$] D”, “A [|] B [\$] D”, “A [|] B [|] C [\$] D”, “B [|] C [|] D [\$] D”, “C [|] D [\$] D”, “D [\$] D”)

Table 1: Input–output pairs for c-concatenations. For brevity, we substitute `[|]` and `[$]` for `[SEP]` and `[EOS]`.

to predict $\boldsymbol{x}^{(k)}$ from $\boldsymbol{x}^{(j)}$ for each $1 \leq j \leq k$; that is, we model

$$p(\boldsymbol{x}^{(k)}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(j)}). \tag{1}$$

In keyboard-input QAC, there is only a single transcript, the submitted query, which researchers model with prefix trees (Mitra and Craswell, 2015) and autoregressive language models (Park and Chiba, 2017) to generate completions from prefixes. A tacit assumption is that partial queries are prefixes (or within a small edit distance of an observed prefix; Chaudhuri and Kaushik, 2009) of the final query. Since this property is invalid for us, we propose voice query-oriented flavors of two state-of-the-art QAC approaches, most popular completion (MPC) and neural query language models (NQLMs), representing a statistical and a neural approach, respectively.

2.1 Concatenated Sequence Transformation

To model Eqn. (1) using autoregressive models, we propose to concatenate some intermediate transcripts with the final transcript for each utterance. Let `[SEP]` and `[EOS]` be the separator and the end-of-sequence tokens, respectively. Define all c-concatenations of $\boldsymbol{X}$ as the set of strings where all contiguous windows of $\leq c$ transcripts are joined together with `[SEP]` and prepended to “`[EOS]` $\boldsymbol{x}^{(k)}$.” This transformation, when viewed in an autoregressive, left-to-right manner, relates the final transcript to c intermediate ones. For input–output examples, see Table 1. Given the context size c, we then construct a training set of strings $\mathcal{D}_c$ from the training corpus of N transcript tuples $\mathcal{D} := (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N)$ as the set of all c-concatenations of each $\boldsymbol{X} \in \mathcal{D}$.

We differentiate the motivation of our method from that of Tsunematsu et al. (2020), who treat speech completion as the same as typographical QAC. Typing is much more linear than automatic speech recognition is, with intermediate keystrokes (or queries) being within a short edit distance of the final query. In other words, typographical sequence completion already has the full history of the query inherently, in the final query text itself. As a result, our motivation is to include the same information that is available to typographical systems by including the full transcript history, captured from the ASR system.

2.2 Our Models

Most popular completion. In keyboard-input most popular completion, researchers construct a trie over the characters of each query in the training corpus, keeping track of the frequency. At inference time, given some prefix of the query, the top-K completions from the trie are returned. In our case, given the context size c, we build the trie from $\mathcal{D}_c$, naming this method “con**cat**enated MPC,” or “CAT-MPC” for short.

Neural query language models. The clear drawback of MPC is that it fails to complete unseen prefixes. The current state-of-the-art workaround is to apply neural language models (NLMs; Park and Chiba, 2017) rather than relying on observed statistics, thus allowing for unseen suffixes to be generated. We propose to model $\mathcal{D}_c$ using lightweight transformers (Vaswani et al., 2017), which represent the state-of-the-art architecture in language modeling (Brown et al., 2020). For learning the statistical distribution $p(W_1, \ldots, W_n)$ over the word sequence $W_1, \ldots, W_n$, NLMs typically use the negative log-likelihood (NLL) objective

$$\sum_{\boldsymbol{x} \in \mathcal{D}_c} \sum_{i=1}^{|\boldsymbol{x}|} -\log p_\theta(W_i = \boldsymbol{x}_i | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}), \tag{2}$$

where $\mathcal{D}_c$ is a training corpus as previously defined, p_θ is an NLM, and $\boldsymbol{x}$ is a tokenized string. We call this approach “con**cat**enated **n**eural **q**uery **l**anguage **m**odel,” or “CAT-NQLM.”

Neural trie objective. NLL is pointwise in the sense that the likelihood of a single word (or outcome) is maximized at each iteration, ignoring the full distribution across the vocabulary. On a long corpus, this is the best we can do, for the data is too sparse to provide an estimate beyond the next-best token conditioned on all its previous ones. On

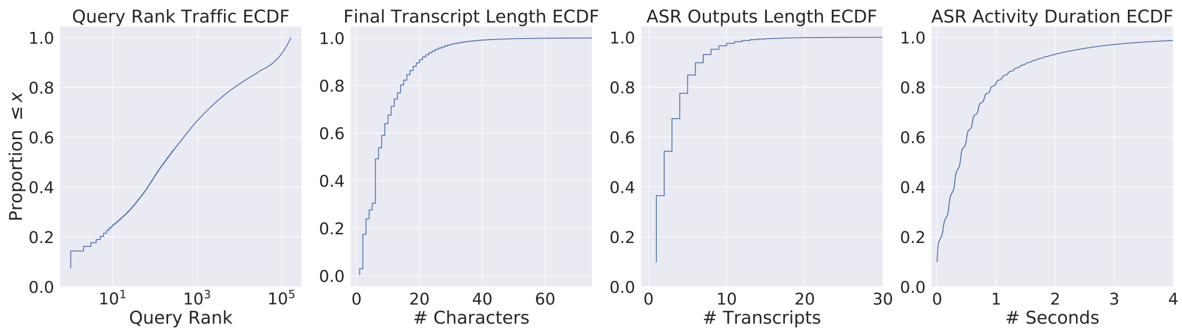

Figure 1: Distribution statistics of queries to the ASR system and its outputs.

query logs, the data is instead short, dense, and well modeled by tries (as the high quality of MPC shows), enabling us to estimate the distribution across the vocabulary for each new token. We construct a trie $p_{\text{trie}}(W_1, \ldots, W_n)$ over the dataset $\mathcal{D}_c$ and introduce the objective

$$\sum_{\boldsymbol{x} \in \mathcal{D}_c} \sum_{i=1}^{|\boldsymbol{x}|} \text{KL}(p_\theta(W_i | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}) \mid\mid p_{\text{trie}}(W_i | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1})), \quad (3)$$

where KL$(\cdot||\cdot)$ denotes the Kullback–Leibler divergence. Unlike Eqn. (2), this loss uses the entire distribution across the vocabulary. In the *data-sparse* case, this objective degenerates to the negative log-likelihood loss since $p_{\text{trie}}(W_i = \boldsymbol{x}_i | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}) = 1$. We name Eqn. (3) the neural trie (NT) objective.

Model inference. At inference time, given some intermediate transcripts, we join the final c transcripts with `[SEP]` and append the `[EOS]` sentinel. For CAT-MPC, we return the top-K completions following the sentinel; for CAT-NQLM, following Park and Chiba (2017), we feed the transformed string into the NLM, run beam search with a width of K, and return the generated tokens.

3 Experiments

3.1 Experimental Setup

We run experiments using PyTorch and Transformers (Wolf et al., 2019) on machines with Titan RTX GPUs. For CAT-NQLM, we use the same architecture as GPT-2-base (Radford et al., 2019) but with an embedding and hidden size of 256, 8 attention heads, 4 layers, and 8,000 tokens. This model runs in real time (much less than 100ms, the limit for instantaneous perception) and totals 4.7 million parameters, which is slightly larger than the small 3.8M-parameter model from Park and Chiba (2017) and much smaller than their large 30M variant.

We train this model using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 5×10^{-4}, a batch size of 128, and 5 epochs. We denote the models trained using the neural trie objective with the "NT" subscript. For tokenization, following the current state of the art, we apply byte-pair encoding (Kudo and Richardson, 2018) to solve the out-of-vocabulary problem. We tune c from Section 2.1 for $c = 1, \ldots, 5$ for all CAT-* approaches and pick a beam search width of 10. For more specific training details, refer to the appendix.

Dataset. We curate a proprietary dataset from real-life voice queries to our smart TV, the X1 entertainment system, which users interact with using a voice remote. Example queries include navigating to a specific channel ("Channel 5"), searching YouTube ("YouTube funny videos"), and program lookups ("Cowboy Bebop").

In Figure 1, we present the empirical cumulative distribution functions (ECDFs) of important statistics. From left to right, we plot the frequency rank of the query, the lexical length of the query, the number of ASR outputs, and the duration between the first and the last output. Note that this output activity duration is zero seconds for single-output utterances. We observe that the top-10 queries make up 25% of the traffic, which is roughly equal to the long tail past a rank of 1000 (see the leftmost figure). As the middle two figures show, the final queries are mostly short with few intermediate ASR outputs, with 80% of them having fewer than 15 characters and yielding no more than 5 intermediate transcripts. Thus, the ASR system actively outputs for less than a second on most queries, as plotted in the rightmost figure. For a detailed analysis of the queries, see our previous work (Li and Ture, 2020; Tang et al., 2019; Rao et al., 2018).

#	Model	All	Seen	Unseen	c
1	MPC	0.494	0.656	0.0	–
2	MPC-KExt	0.518	0.689	0.0	–
3	NQLM	0.531	0.626	0.245	–
4	CAT-MPC	0.547	0.726	0.0	1
5	CAT-NQLM	0.625	0.741	0.273	5
6	CAT-NQLM$_{NT}$	**0.629**	**0.744**	**0.278**	5

Table 2: The test set results (MRRs) on EntSys1M, taking the median across five randomly seeded runs, with the best bolded. c denotes the best context size from Section 2.1, picked based on the dev set results.

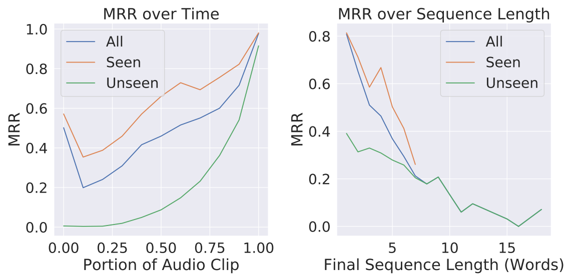

Figure 2: The test set MRRs of CAT-NQLM$_{NT}$, the best model, on EntSys1M as a function of time (left) and the length of the final transcript (right).

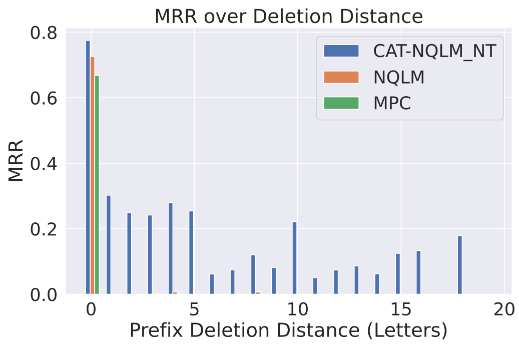

Figure 3: The test set MRR plotted against the prefix deletion distance, truncated at 20 for illustration.

For the training and the development sets, we collect the streaming transcripts (provided by a third-party ASR system) of one million voice queries sampled uniformly at random from April 14th, 2021, setting aside 10% of it for development and the rest for training. This set represents roughly 3% of our daily traffic, containing 163K unique final transcripts . For the test set, we sample 100k queries from April 15th, 2021, a different date from the training set's, as is common in QAC datasets for testing generalizability (Adar, 2007). We call this dataset "EntSys1M."

To evaluate the models, we compute for each prediction the mean reciprocal rank (MRR), defined as MRR $:= \frac{1}{|\tilde{\mathcal{D}}_c|} \sum_{u_i \in \tilde{\mathcal{D}}_c} \mathrm{RR}_{u_i}$, where $\tilde{\mathcal{D}}_c$ is the transformed test set using Section 2.1, and RR$_{u_i}$ is the reciprocal rank (index) of the first correct prediction in the top-K completions, as produced following Section 2.2. If no correct prediction exists, then RR$_{u_i} = 0$. We further split the test set into two distinct subsets: final transcripts seen at training time and unseen ones.

Baseline models. We implement MPC and NQLMs (Park and Chiba, 2017) trained on the final transcript of each utterance, with the NLM architecture and training procedure matching our CAT-NQLM's for a fair comparison. We also implement and train the error-tolerant version of MPC (Chaudhuri and Kaushik, 2009), named MPC-KExt, which allows for up to an edit distance of $\hat{k}$ between the lookup prefix and the observed prefixes. We tune $\hat{k}$ on the development set for $\hat{k} = 1, \ldots, 10$.

3.2 Results and Discussion

We present the overall quality of the models in Table 2. Our proposed approaches (rows 4–6) outperform the existing ones (rows 1–3) by 0.01–0.1 points in MRR on the full set. In absolute terms, our best model, CAT-NQLM$_{NT}$, improves over the previous best, NQLM, by about a third of a rank of the correct prediction, on average. MPC-KExt, which is tolerant to some changes in the prefix relative to the final transcript, still underperforms our proposed methods, likely because the intermediate transcript isn't necessarily close in edit distance to the final (e.g., "Who" and "Hulu"). These results highlight the importance of conditioning the model on the intermediate transcripts.

We confirm that the neural trie objective (Eqn. 3; row 6) improves over the negative log-likelihood loss (row 5), showing that training against the entire vocabulary distribution for each word helps. While not shown due to space constraints, these gains are consistent for each $c = 1, \ldots, 5$ and require no extra machinery (and hence latency) at inference time, thus making CAT-NQLM$_{NT}$ Pareto-better. We note that picking a larger context size (c) does not always result in a better model: CAT-MPC is best when $c = 1$ because the statistics are more sparse for $c = 2$ and above, yielding less robust predictions for count-based methods.

Subgroup analysis. We further study how the model quality changes with different characteristics of the transcripts. In Figure 2, we choose CAT-NQLM$_{NT}$, the best model, and plot the MRR against the portion of the audio clip remaining, as well as the length of the final transcript in the right subfigure. Note that the seen set cuts off after 7 words due to a natural mismatch between the statistics of the training set and the test set. We find that the MRR falls off as less audio is available (i.e., the first few transcripts are generally uninformative), with the unseen set's quality decreasing the fastest. We note a sharp uptick on the seen set and the all set when the full audio clip is remaining, mainly because many simple, short queries have a single intermediate transcript. Similarly, due to increased query diversity and audio clip length, the MRR worsens with increasing final transcript length (see the right subfigure).

Finally, in Figure 3, we graph the MRR split across different prefix deletion distances, defined as the number of characters to delete from the end of the intermediate transcript for it to be a prefix of the final one. The NQLM and the MPC approaches fail when the intermediate transcript is *not* a prefix of the final transcript, and our best model surprisingly outperforms them even when the deletion distance is zero—see the leftmost bucket. These results suggest that our proposed approach is robust across all prefix deletion distances, including zero.

4 Related Work

Tsunematsu et al. (2020) study speech transcript completion for unidirectional ASR systems on non-query data, while our focus is QAC on real-life voice queries with a typical ASR system where intermediate transcripts don't necessarily form prefixes of the final one. Park and Chiba (2017) are the first to apply neural language models to QAC, representing the state of the art; Fiorini and Lu (2018) extend this work with user personalization. Other more restricted examinations include improving QAC for rare prefixes (Mitra and Craswell, 2015), QAC in the presence of typographical errors (Chaudhuri and Kaushik, 2009), efficient QAC (Wang et al., 2020), and the effects of conversations on voice QAC (Vuong et al., 2021).

5 Conclusions and Future Work

We study the task of QAC for voice queries on bidirectional ASR systems. Along with an improved language modeling objective for query logs, we propose several novel methods which relate the mid-utterance transcripts to the final one, attaining relative gains of 18% over the previous best.

Voice QAC lends itself to a variety of end applications: For one, on voice-controlled smart televisions, it can guide viewers toward final queries in real time, much like the now-retired Google Instant feature. For another, in general voice query processing pipelines, it can serve as part of a latency reduction method, where the most likely voice queries are speculatively processed and their responses precomputed as the user speaks. If we know what the user is going to say before they finish speaking, then we can speculatively send those predicted final queries to the rest of the information retrieval system, while the user is speaking. We plan to explore these lines of research in future work.

References

Eytan Adar. 2007. User 4xxxxx9: Anonymizing query logs. In *Proceedings of the Query Log Analysis Workshop, International Conference on World Wide Web*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv:2005.14165*.

F. Cai and M. de Rijke. 2016. A survey of query auto completion in information retrieval. *Foundations and Trends in Information Retrieval*.

Surajit Chaudhuri and Raghav Kaushik. 2009. Extending autocompletion to tolerate errors. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*.

Nicolas Fiorini and Zhiyong Lu. 2018. Personalized neural language models for real-world query auto completion. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Taku Kudo and John Richardson. 2018. 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*.

Wenyan Li and Ferhan Ture. 2020. Auto-annotation for voice-enabled entertainment systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Bhaskar Mitra and Nick Craswell. 2015. Query auto-completion for rare prefixes. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*.

Dae Hoon Park and Rikio Chiba. 2017. A neural language model for query auto-completion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. 2019. Better language models and their implications. *OpenAI Blog*.

Jinfeng Rao, Ferhan Ture, and Jimmy Lin. 2018. What do viewers say to their TVs? an analysis of voice queries to entertainment systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Raphael Tang, Ferhan Ture, and Jimmy Lin. 2019. Yelling at your TV: An analysis of speech recognition errors and subsequent user behavior on entertainment systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Kazuki Tsunematsu, Johanes Effendi, Sakriani Sakti, and Satoshi Nakamura. 2020. Neural speech completion. In *Proceedings of Interspeech 2020*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

Tung Vuong, Salvatore Andolina, Giulio Jacucci, and Tuukka Ruotsalo. 2021. Spoken conversational context improves query auto-completion in web search. *ACM Transactions on Information Systems (TOIS)*, 39(3).

Sida Wang, Weiwei Guo, Huiji Gao, and Bo Long. 2020. Efficient neural query auto completion. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv:1910.03771*.

A Specifications

Our machines use Titan RTX GPUs, CUDA 10.2, PyTorch 1.8.1, Transformers 4.5.0, and Python 3.8.8. Our NLMs are based on a smaller version of DistilGPT-2 from the HuggingFace Transformers library, with a SentencePiece vocabulary of 8,000 learned from our datasets. Specifically, we make the following changes to the configuration:

```
from transformers import GPT2Config, GPT2LMHeadModel

cfg = GPT2Config.from_pretrained('distilgpt2')
cfg.vocab_size = 8000
cfg.num_labels = 8000
cfg.n_embd = 256
cfg.n_layer = 4
cfg.n_head = 8
model = GPT2LMHeadModel(cfg)
```

We train these models using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 5×10^{-4} and a linear triangular learning rate schedule with a 300 warmup steps and a linear decay until the end, as implemented by the `get_linear_schedule_with_warmup` function from Transformers.