Abstract

Large language models (LLMs) are often trained on extensive, temporally indiscriminate text corpora, reflecting the lack of datasets with temporal metadata. This approach is not aligned with the evolving nature of language. Conventional methods for creating temporally adapted language models often depend on further pre-training static models on time-specific data. This paper presents a new approach: a series of point-in-time LLMs called Time Machine GPT (TiMaGPT), specifically designed to be nonprognosticative. This ensures they remain uninformed about future factual information and linguistic changes. This strategy is beneficial for understanding language evolution and is of critical importance when applying models in dynamic contexts, such as time-series forecasting, where foresight of future information can prove problematic. We provide access to both the models and training datasets.

1 Introduction

Time-series forecasting and event prediction aim to infer a future state of the world from past data. When evaluating models for these purposes through historical data analysis, often referred to as "back-testing", it is crucial to maintain strict data partitioning. This ensures that no future information influences the model’s predictions. Whilst strict data partitioning is standard in most fields that use time-series information, time-series forecasting methods that use transformer-based LLMs have tended to make an assumption that the language model itself cannot be the vector for information leakage from a future state to a past state. However, within a language model, implicit associations, such as linking "Enron" with "bankrupt" or possessing knowledge of terms like "COVID-19" might exist (Figure 1). This poses a challenge for models tested on data predating such events, as their presence could lead to an overestimate in model performance within the validation stage, which could lead to disappointing results when a system is used in a live setting.

The evolution of language models in recent years has been shaped by increases in both the size of these models and their training datasets (Wei et al., 2022). The trend towards larger and more complex datasets has made in-depth analysis of their content increasingly difficult. A significant challenge is the contamination of training datasets, which can include the accidental inclusion of benchmark datasets (Dodge et al., 2021) and private data (Xu et al., 2020). To tackle the issue of temporal data contamination, this paper introduces language models that have been pre-trained on data exclusively published before specified cutoff dates. These models serve two key purposes: analyzing diachronic embeddings over time and facilitating the use of language models in dynamic tasks that demand strict separation of temporal data. Language models are capable of learning both factual information

Figure 1: The perplexity of coronavirus and COVID-19, using TiMaGPT models (•) and Conventional Temporally Adapted (CTA) models (×). The calculation for perplexity is outlined in Appendix H and the methodology for temporally adapting models is explained in Section 5. The CTA models have significant knowledge of these words before the pandemic.
and linguistic patterns (Petroni et al., 2019; Mahowald et al., 2023), which could influence their performance in predictive tasks. The primary application for our TiMaGPT models is in evaluating a system that uses generative language models for dynamic downstream tasks. The yearly models, developed uniformly, display similar performances on well-established benchmarks, meaning that the main difference between the models is the information in the training datasets.

1.1 Research Contributions

Contribution 1 - To our knowledge, the models released in conjunction with this paper are the first series of temporally correct pre-trained LLMs exclusively pre-trained on historical data.

Contribution 2 - Identification of an unacceptable level of foresight in conventional temporally adapted models.

2 Related Work

2.1 Diachronic Embeddings

The meaning of words change subject to the context in which they appear, a fact that initiated the adoption of contextualized embeddings over static embeddings within LLMs (Mikolov et al., 2013a,b; Devlin et al., 2019). This contextual dependency extends beyond the surrounding tokens, as words can change meaning according to the venue (Zeng et al., 2018), domain (Lee et al., 2019a; Yang et al., 2020), time (Pierrehumbert, 2012; Bybee, 2015), location (Dunn, 2023; Hofmann et al., 2023), or task (Gururangan et al., 2020).

Consequently, there has been significant research focused on understanding how embeddings shift through time. Procrustean alignments enabled Kutuzov et al. (2017) to assess the way in which word meaning shifted by diachronically training static embeddings. Some have tried to incorporate these temporal dynamics into LLMs by dynamically adapting word embeddings (Rudolph and Blei, 2017; Hofmann et al., 2021a). Numerous studies have investigated how these embeddings evolve over time (Hamilton et al., 2016; Kutuzov et al., 2018), with practical applications such as detecting change points in language use (Goutte et al., 2018). These studies demonstrate that embeddings can reveal the temporal context of data, underscoring the importance of carefully selecting the data included in training datasets.

2.2 Temporal Adaptation of Language Models

Efforts to temporally adapt language models to date have primarily involved modifying existing statically trained models (Lazaridou et al., 2021; Röttger and Pierrehumbert, 2021; Dhingra et al., 2022). Given that transformer-based LLMs have been predominantly trained since 2017, following the seminal work of Vaswani et al. (2023), and largely on data from post-2017, temporal adaptation has generally involved either further training these models with newer data (Jang et al., 2022) or adjusting them to represent a past state by training on historical data for a fixed number of steps (hereafter "CTA models" - Conventional Temporally Adapted) (Wenjun Qiu and Xu, 2022a; Martin et al., 2020). Both methods have significant limitations, since either any resultant downstream analysis is limited to the very short time after the models were trained, or the temporally adapted models have seen future data within the pre-training stage. This paper restores language models to a prior state in time by pre-training a series of models on data that has strict temporal inclusion criteria.

3 Training Process

3.1 Training Datasets

The lack of temporal metadata in natural language processing (NLP) presented a challenge in selecting datasets for training our models. However, news data and Wikipedia version history emerged as valuable resources. Detailed token counts for each year’s deduplicated datasets are provided in Appendix B. Each year from 2011 to 2022 contained sufficient data to train a GPT-2 small model.

Wikipedia: By utilizing the revision information from Wikipedia XML dumps provided by Wikimedia2, we reconstructed every existing Wikipedia page as they would have appeared on 31/12 of each year from 2004 to 2023. This reconstruction accounted for changes in page titles. The identified revisions were then processed to remove links, HTML, and other non-standard stylistic elements, using the following code repository 3.

WMT News: The WMT News dataset, typically used in machine translation (Kocmi et al., 2022), was processed in its monolingual, document-split English version. We applied deduplication to this dataset, eliminating repeated articles via an

\(^{2}\)https://dumps.wikimedia.org

\(^{3}\)http://tinyurl.com/2exawtkf
SHA-256 hashing function (Mou et al., 2023). The dataset ranges from 2007 to 2022.

### 3.2 Dataset Aggregation

Several studies have demonstrated that the data types used in training an LLM significantly influence its performance in downstream tasks. This insight led to the development of domain-specific language models such as BioBERT (Lee et al., 2019), SciBERT (Beltagy et al., 2019), FinBERT (Yang et al., 2020), and more recent models like BloombergGPT (Wu et al., 2023). Acknowledging this, we maintained a consistent token allocation from each domain in our annual datasets. This approach ensured that the language models’ performance wasn’t skewed by shifts in the relative size of different data domains over time. Consequently, the only differences among the various training datasets are the new information and time-specific stylistic changes unique to each period.

#### 3.2.1 Sampling

To maintain a predetermined domain allocation ratio of 0.6:0.4 (WMT News to Wikipedia), a ratio that was determined by model tuning outlined in Appendix F, we employed specific sampling strategies for each dataset.

We randomly sampled Wikipedia articles from each year, ensuring articles were not chosen twice. Additionally, we included the "Vital Level 4" pages — the top 10,000 most important Wikipedia articles — in each training dataset. The Level 4 articles changed slightly over time, so our selection was based on the list available at the end of each year.

For the WMT News dataset, ordered as a text stream, we have included data according to a negative exponential probability function over a 5-year period to prioritize recent data over older data. We first identify the start date of the 5-year window and calculate the number of days from the cutoff date, represented as $\tau$. For each entry $e_i$ with an age of $D_i$ days in the dataset, we compute a weight that is assigned to each entry based on its age, given by:

$$W_i = \exp\left(-\frac{D_{\text{max}} - D_i}{\tau}\right)$$

(1)

The probability of selecting each entry, $P_i$, is inversely proportional to its weight, such that:

$$P_i = \frac{1}{N} \frac{1}{W_i} \sum_{j=1}^{N} W_j$$

(2)

where $N$ is the number of entries in the dataset.

In the process of sampling, our goal is to accumulate a certain number of tokens, denoted as $T_{\text{needed}}$. Starting with an initial token count of $T_{\text{current}} = 0$, we repeatedly sample with probability $P_i$ until:

$$T_{\text{current}} \geq T_{\text{needed}}$$

(3)

If adding the tokens of a chosen entry does not exceed $T_{\text{needed}}$, we add the entry to the training dataset and update the token count $T_{\text{current}}$.

#### 3.3 Pre-training Details

The full training details for replicating our work are provided in Appendix C. In line with the Chinchilla ratio, which recommends a 1:20 parameter-to-token ratio for efficient training (Hoffmann et al., 2022), a GPT-2 model with 117 million parameters requires 2.34 billion tokens for optimal training. We trained each of our models on 2.5 billion tokens and used a BPE tokenizer as was used in the original GPT-2 paper Radford et al. (2019). To confirm that this amount of data was sufficient, we performed a comparative analysis of models trained with varying token counts, detailed in Appendix B. Considering the numerous models we had to train, we optimized our training framework for computational efficiency. Therefore when two samples could be combined into the 1024 token sequence we concatenated them. A similar methodology only saw a marginal reduction in performance when training RoBERTa (Liu et al., 2019).

### 4 Model Verification

Verifying that each of our models achieves an adequate level of performance is essential. To conduct meaningful analysis on downstream tasks, it is vital to ensure consistent performance on static benchmarks from models from different years. This consistency means that we can assume that the majority of any observed changes are due to variations in the information within the training datasets, not fluctuations in model efficacy. When selecting candidate benchmarks, we observed that for some newer and more complex benchmarks models of this size have a performance similar to the random baseline. This is due to the rapid progress in language model performance in recent years, and the need to create new benchmarks to match that progress. A more detailed description of the tasks that were included is in Appendix E. Table 3 demonstrates that while our models are far from the state-of-the-art, they
Table 1: Performance of the models on static benchmarks to validate performance. HellaSwag, TruthfulQA, PIQA, Winogrande, WSC (Appendix E). Comparison models: GPT-2 (Radford et al., 2019), OPT (Zhang et al., 2022), GPT-Neo (Black et al., 2021).

<table>
<thead>
<tr>
<th>Model</th>
<th>Av. HellaSwag</th>
<th>PIQA</th>
<th>TruthfulQA</th>
<th>Winogrande</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>39.5</td>
<td>25</td>
<td>50</td>
<td>22.5</td>
<td>50</td>
</tr>
<tr>
<td>GPT-2 Small</td>
<td>45.85</td>
<td>31.14</td>
<td>62.51</td>
<td>40.69</td>
<td>51.62</td>
</tr>
<tr>
<td>OPT 125m</td>
<td>44.60</td>
<td>31.34</td>
<td>62.02</td>
<td>42.87</td>
<td>50.20</td>
</tr>
<tr>
<td>GPT-Neo 125m</td>
<td>45.08</td>
<td>30.26</td>
<td>62.46</td>
<td>45.58</td>
<td>50.43</td>
</tr>
<tr>
<td>TiMaGPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TiMaGPT-11</td>
<td>48.74</td>
<td>25.14</td>
<td>50.87</td>
<td>52.83</td>
<td>51.38</td>
</tr>
<tr>
<td>TiMaGPT-12</td>
<td>48.69</td>
<td>25.26</td>
<td>50.98</td>
<td>53.30</td>
<td>50.99</td>
</tr>
<tr>
<td>TiMaGPT-13</td>
<td>48.62</td>
<td>25.12</td>
<td>50.82</td>
<td>53.11</td>
<td>50.36</td>
</tr>
<tr>
<td>TiMaGPT-14</td>
<td>48.61</td>
<td>25.04</td>
<td>50.27</td>
<td>52.88</td>
<td>50.04</td>
</tr>
<tr>
<td>TiMaGPT-15</td>
<td>48.75</td>
<td>24.98</td>
<td>50.76</td>
<td>52.74</td>
<td>50.59</td>
</tr>
<tr>
<td>TiMaGPT-16</td>
<td>48.99</td>
<td>25.00</td>
<td>50.27</td>
<td>52.60</td>
<td>51.62</td>
</tr>
<tr>
<td>TiMaGPT-17</td>
<td>48.98</td>
<td>25.09</td>
<td>50.76</td>
<td>52.25</td>
<td>51.62</td>
</tr>
<tr>
<td>TiMaGPT-18</td>
<td>48.43</td>
<td>25.13</td>
<td>51.31</td>
<td>52.41</td>
<td>49.64</td>
</tr>
<tr>
<td>TiMaGPT-19</td>
<td>48.66</td>
<td>25.30</td>
<td>50.98</td>
<td>52.30</td>
<td>50.83</td>
</tr>
<tr>
<td>TiMaGPT-20</td>
<td>48.65</td>
<td>25.07</td>
<td>50.77</td>
<td>52.88</td>
<td>51.14</td>
</tr>
<tr>
<td>TiMaGPT-21</td>
<td>48.58</td>
<td>25.38</td>
<td>51.52</td>
<td>52.55</td>
<td>50.67</td>
</tr>
<tr>
<td>TiMaGPT-22</td>
<td>48.52</td>
<td>25.34</td>
<td>51.47</td>
<td>52.90</td>
<td>50.04</td>
</tr>
</tbody>
</table>

perform in line with other similarly-sized models like GPT-2, OPT-125m and GPT-Neo 125m on several established benchmarks but crucially also maintain this performance over time. TiMaGPT has a slightly different performance profile to the comparison models, with better performance on the WSC and TruthfulQA benchmarks and worse performance on the HellaSwag and PIQA datasets. Interestingly, both of the benchmarks that TiMaGPT performed badly on were challenging common-sense reasoning datasets. Perhaps the factual bias and lack of diversity of our training data led to poor performance on these benchmarks. All of the models perform just slightly above random for the Winogrande benchmark, indicating that this benchmark is too challenging for models of this type and size. The lack of variance of the TiMaGPT results on the WSC benchmark can be attributed to the dataset’s size - only 273 samples in total.

5 Temporal Evaluation

Previously, models were adapted by further training a statically trained model on period-specific data (Wenjun Qiu and Xu, 2022b; Dhingra et al., 2022), giving them foresight from the pre-training stage, which could be problematic for tasks where temporal segregation is important. We compared our models with Conventionally Temporally Adapted (CTA) models to show the extent of the information leakage when adopting the traditional methodology, by assessing their perplexity in recognizing the names of country leaders around their inauguration. The perplexity measurement is outlined in Appendix H and the dataset identifies leaders that came into power between 2013 and 2020 (Herre, 2023). 310 leaders are considered, corresponding to 154 countries.

We contrasted our TiMaGPT models with CTA models, which are versions of the TiMaGPT model further pre-trained on 1 billion tokens from the same datasets used for pre-training the yearly TiMaGPT models. Figure 2 shows the differences in methodologies, with CTA models retaining un-
realistic knowledge of the leaders well in advance of their inauguration. The lower CTA perplexity scores come from seeing the leaders in the pre-training dataset that trained TiMaGPT_{2022}, providing the CTA models with information that would not have been available to models that were trained at the time. The relatively low perplexity from TiMaGPT_{2022} supports this claim. It is clear from the performance delta between the CTA models and the TiMaGPT_{2022} model two years before inauguration that whilst temporal adaptation does shift the model’s perplexity distribution closer to that of the TiMaGPT models, some information is preserved post-adaptation.

Beyond named entities, we show that CTA models have an unrealistic knowledge of concepts like "COVID-19" or "coronavirus"; Figure 1 exposes a very significant difference between our models and traditional adaptation methods. The CTA model perplexity scores are lower than our TiMaGPT models, which reflect what could have been produced at the time. The TiMaGPT dataset partitioning means that the information leakage seen in Figure 1 and 2 does not occur.

6 Discussion

This paper provides a tool for researchers focused on tracking knowledge and association shifts in language, and also in evaluating the performance of temporally dynamic models. The recent trend of using GPT-2 as a backbone for time-series forecasting, as highlighted in recent literature (Cao et al., 2023; Chang et al., 2023; Zhou et al., 2023; Liu et al., 2024), underscores the growing interest in integrating language models and textual features to enhance forecast accuracy (Drinkall et al., 2022; Cao et al., 2023). The models developed in conjunction with this paper are particularly valuable in this context. They serve as an effective means to minimize look-ahead bias in time-series models that concurrently process textual and time-series data. By ensuring these models are devoid of future linguistic information, they enable a more accurate and authentic assessment of a model’s forecasting ability, crucial for applications where current data must be interpreted without the influence of future events. Further work could explore the magnitude of the effect of this look-ahead bias by measuring the performance delta between models that have and have not seen future information in their pre-training.

7 Limitations

The paper uses the small GPT-2 architecture, which is outperformed by many newer language models. To create larger TiMa models, it is necessary to expand the size and number of datasets with temporal metadata. This expansion is crucial because each parameter in these models requires around 20 tokens for optimal pre-training (Hoffmann et al., 2022). In addition, we have only explored generative models in this paper, but a significant amount of research still relies on encoder-based LLMs which limits the scope of this paper.

To scale to even larger models, processing the annual Common Crawl datasets is a necessary step, though the dataset has proved problematic due to its scale and lack of consistent formatting (Luccioni and Viviano, 2021). These problems prompted the C4 dataset (Dodge et al., 2021), but replicating that consistent quality over several partitioned years would be a significant challenge. Aside from this, cleaning Common Crawl would also demand significant computational resources.

Acknowledgements

We are grateful to Graphcore, and their team, for their support in providing us with compute for this project. The first author was funded by the Economic and Social Research Council of the UK via the Grand Union DTP. This work was supported in part by a grant from the Engineering and Physical Sciences Research Council (EP/T023333/1). We are also grateful to the Oxford-Man Institute of Quantitative Finance and the Oxford e-Research Centre for their support.

References


Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.


Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. *Towards continual knowledge learning of language models*.

Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki


Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fodoreoko. 2023. Dissociating language and thought in large language models.


Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.


A Risks

The risks associated with this paper are not that significant due to the type of data used. The datasets and benchmarks in this paper are all open source. Long-term use of the WMT News datasets minimizes the chance of persisting errors. However, Wikipedia data, editable by anyone, could be less reliable. The selected December 31st revision may have inaccuracies. We haven’t taken additional measures to verify the truthfulness of the content. A study by Kräenbring et al. (2014) found Wikipedia’s pharmacology information 99.7% accurate, but this may not hold true for other subjects.

B Token counts

The base datasets grow and shrink over time. Our sampling method from Section 3.2.1 means that the domain split of the data stays static across our models. Table 2 tabulates the overall token counts of the cleaned deduplicated datasets from which the training datasets are made.

C Training details

All models are trained on a Graphcore IPU-POD16 using the gpt2-small-ipu config, which employs tensor sharding for efficient distribution across multiple IPUs. We use the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 10^{-8}$, and a weight decay of 0.1. We adopt a linear warm up from $0.1 \cdot LR_{max}$ to $LR_{max}$ over 10 percent of the training data. The subsequent learning rate was determined by a linear scheduler from $LR_{max}$ to $0.1 \cdot LR_{max}$ over the rest of the training data.

The models were designed with a context span of 1024 and configured to generate sequences of up to 50 tokens. We adopted the GPT2LMHeadModel with the GELU new activation function, comprised of 12 layers and 12 attention heads, and an embedding dimension of 768. A single seed was used to initialise the training of all of the models.

D Wikipedia Processing

In conjunction with this paper, we are releasing the yearly partitions of Wikipedia that were instrumental in creating our training datasets. WikiMedia routinely publishes dumps of Wikipedia, each containing the revision history of articles. With approximately 60 million articles on Wikipedia,
many having thousands of revisions, processing these revisions demands substantial computational resources. To streamline this process, we first defined the relevant revision before extracting the article information. Specifically, we select the most recent revision as of December 31st for each year. Consequently, some revisions in our datasets, such as those in the 2020 training set, date back to before 2006, as illustrated in Figure 3. While this inclusion of older revisions might initially appear problematic, it is important to note that these are the existing versions of Wikipedia pages as of the cut-off date. The content of these pages was considered current enough at that time, implying that a more recent revision was not necessary. This approach ensures that our training datasets reflect the most up-to-date information available on Wikipedia at each year’s end, providing a realistic snapshot of knowledge for that specific point in time.

Once each revision has been identified we clean the page using the code from wiki-dump-reader⁶, which parses the page and outputs clean text. During the cleaning phase a number of unwanted features and attributes are removed: file links, emphaeses, comments, indents, HTML, references etc.

### E Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Type</th>
<th>Num. Shots</th>
<th>Acc.</th>
<th>Num. Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HellaSwag</strong> (Zellers et al., 2019)</td>
<td>10-shot</td>
<td>acc_norm</td>
<td>10,042</td>
<td>44,000</td>
</tr>
<tr>
<td><strong>TruthfulQA</strong> (Lin et al., 2022)</td>
<td>0-shot</td>
<td>mc2</td>
<td>817</td>
<td>44,000</td>
</tr>
<tr>
<td><strong>PIQA</strong> (Bisk et al., 2019)</td>
<td>1-shot</td>
<td>acc_norm</td>
<td>1,838</td>
<td>44,000</td>
</tr>
<tr>
<td><strong>WSC</strong> (Levesque et al., 2012)</td>
<td>5-shot</td>
<td>acc</td>
<td>273</td>
<td>44,000</td>
</tr>
<tr>
<td><strong>Winogradne</strong> (Sakaguchi et al., 2019)</td>
<td>5-shot</td>
<td>acc</td>
<td>44,000</td>
<td>44,000</td>
</tr>
</tbody>
</table>

---

⁶https://github.com/CyberZHG/wiki-dump-reader/tree/master

---

Table 2: Token counts of the base domain datasets after cleaning and deduplication; WMT: the token count of the articles from that year; WMT Cumulative: represents the token count of all WMT articles before each cut-off date; WMT 5-MS: the moving sum of the preceding 5 years of WMT data, which is all the data that we sample from for each year; Wiki: the token count from the whole Wikipedia yearly partition; Wiki Core: the token count of the Level 4 Vital Wikipedia pages.
Table 3: Performance comparison of different models trained, including GPT-2 for reference. Benchmarks: HellaSwag, TruthfulQA, PIQA, and WSC.

### F Model Tuning

The following section outlines the process for deciding which assumptions to make and parameters to use in the creation of our training datasets and models. We used the 2020 for the majority of the tuning and tested the tokenizer, dataset size, and data domain split ratio. The tuning was not rigorous since training every configuration of the models would have been computationally prohibitive and unproductive.

#### F.1 Tokenizer

(Radford et al., 2019) used a BPE tokenizer to originally train GPT-2. However there have been many papers that have shown that BPE is problematic in the way it segments words (Hofmann et al., 2021b). As a result, we tested the BPE tokenizer against a Sentence Piece tokenizer. The search for the optimal tokenizer was far from extensive, but from the two tokenizers BPE performed better so it was selected to train the rest of the models.

#### F.2 Dataset Size

Although (Hoffmann et al., 2022) showed that the ratio of tokens to parameters should be 20:1 for complete pre-training, we wanted to test the effect of adding more data than the required amount. Therefore we tested the performance of using a 5B and 10B token training dataset and ran the training for 1 epoch. The datasets were constructed in exactly the same way as the 2.5B token dataset and were just sampled for longer until the required token count was met. Table 3 shows clearly that dataset size does not affect the downstream performance on our benchmark datasets.

#### F.3 Domain Split

We also fine-tuned the proportion of each data domain within the training dataset. Previous research, as noted in Section 3.2, has shown that the type of domain in the training data can influence downstream performance. Therefore, we determined the optimal proportion of each dataset that yielded the best results for both the 2011 and 2020 data. The comparison between two models at different extremes of our time period meant that we could feel more confident that the optimal ratio split was consistent across time. Given that the 0.6:0.4 ratio of WMT to Wiki data was the best performing in 2011 and the second best in 2020 we went with this domain split for all of our models.

### G Dataset Histogram

The two base datasets, WMT and Wikipedia, used to create the training dataset used different timestamp formats. For Wikipedia, the most recent revi-
sion to the cut off date was the version used in creating the yearly datasets. This meant that although Wikipedia was treated as a snapshot in time and was accordingly randomly sampled, some revision versions were older than others. The histogram in Figure 3 outlines the revision publication dates of each of the samples in the 2020 dataset, with the maximum date 2020-12-31. The WMT dataset only exists in yearly buckets, which limits the granularity of the dataset. There is no data used past the cutoff date but the exact distribution across the months and weeks is not possible to know.

H Perplexity Calculation

In our perplexity calculations, we deviate from the conventional methodology of computing perplexity (PPL) of a language model, where some preceding context is usually considered. Instead, we calculate the PPL with zero context.

Formally, the perplexity of a sequence \( X = (x_0, x_1, \ldots, x_t) \) without considering any preceding tokens is given by:

\[
PPL(X) = \exp \left\{ -\frac{1}{t} \sum_{i=0}^{t} \log p(x_i) \right\}
\]

where \( p(x_i) \) is the model’s estimated probability of the token \( x_i \), independent of any preceding sequence.

This zero-context perplexity enables us to understand the models’ comprehension of an individual word, without being biased by the context that precedes it.

To visualise the effect that the training data has on the model, Figure 1 shows the perplexity of the words "COVID-19" and "coronavirus" using the TiMaGPT models. We would expect the model to have no real knowledge of what COVID-19 is before 2020 and then a significant understanding during and after. Figure 1 shows that this is the case for TiMaGPT models, as the models all have very high perplexity before the pandemic and very low perplexity after. This is due to the differences in the training datasets. Figure 4 shows the different exposures the models had to the words "COVID-19" and "coronavirus".

I Licenses

I.1 External Licenses

In the making of our training datasets and training of our models we used data and code that were licensed in ways that might be of interest to the reader. The Wikipedia dump data is licensed under a GNU Free Documentation License (GFDL) and the Creative Commons Attribution-Share-Alike 3.0 License, two very permissive licenses. The WMT News data is released under the same terms as the ParaCrawl dataset \(^7\), meaning that WMT claim no ownership over the text and that the packaging of the data is released under a Creative Commons CC0 Licence, which means that they do not reserve any rights over the way the data is assembled. There is however some copyrighted material in the dataset, which we use under Fair Use \(^8\) and Fair Dealing \(^9\) principles.

We have also used various software packages when creating these models, which can all be accessed under permissive licenses. The lm-eval-harness package, which was used to evaluate the models, is released under an MIT License \(^10\). The

\(^7\)https://www.paracrawl.eu
\(^8\)Fair Use (US): http://tinyurl.com/497jze9m
\(^9\)Fair Dealing (UK): http://tinyurl.com/5f7nw4tu
\(^10\)http://tinyurl.com/bdeapaze
transformers and optimum-graphcore packages, which were used to train the models, are released under and Apache 2.0 License \(^\text{11} \quad \text{12}\).

I.2 Our Licenses

We release our models and datasets in accordance with the licenses of the original works. We do not claim ownership over any of the material used. We license the packaging of the data and models under a Creative Commons CC0 license ("no rights reserved"). The datasets for the models are of academic interest and therefore fall under Fair Use/Dealing principles. However we will comply with any legal requests pertaining to our data if we are legally compelled to do so.

J Emissions

Training models has both computational and environmental implications. The energy consumption of training large language models can be substantial. To quantify this, we calculated the energy consumption of training all of our models and the associated carbon emissions. The computational costs for cleaning the datasets are not considered but are significant: the Wikipedia datasets took several days to extract and clean. The computational costs for evaluation are also not considered but are significant: each model was evaluated extensively.

Our models were trained using a GraphCore Pod with 16 IPU-M2000 chips, which each consumes a maximum of 6kW of power (Graphcore). To train all of the models in this paper the POD-16 was consumed 388.40 kWh.

The IPU POD-16 is situated in Charlotte, North Carolina. Given the carbon emissions from this grid is 328gCO\(_2\)eq/kWh \(^\text{13}\), the carbon emissions associated with the energy used for a single model training can be deduced:

\[
\text{Emissions} = E \times \text{Carbon Intensity} \\
= 388.40\text{kWh} \times 342\text{gCO}_2\text{eq/kWh} \\
= 132,832.80\text{gCO}_2\text{eq}
\]

or equivalently, 132.83kgCO\(_2\)eq.

Whilst this is significant, the emissions are significantly reduced by the hardware that the models were trained on. The Graphcore POD-16 is very efficient which means that the emissions associated with the training of the models are less than the average transatlantic flight (Mazareanu, 2020).

\(^{11}\)http://tinyurl.com/yc7mvkny
\(^{12}\)http://tinyurl.com/mspzm9jp
\(^{13}\)http://tinyurl.com/2bnsv8yh