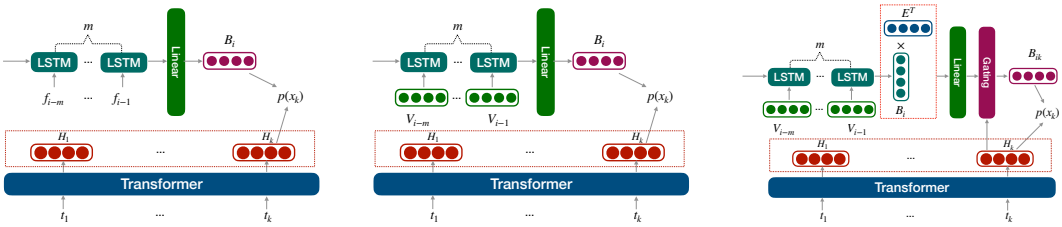

(a) Word Frequency Model (§3.2).

(b) Temporal Contextual Model (§3.3)

(c) Doubly Contextualized Model (§3.4)

Figure 2: Our proposed models.

bias. This bias is added to the final softmax to bias the model towards historical trends. Figure 1b shows the frequency by year for some example words, which reflects topic/approach changes over the years, i.e., the word "pretrain" started to dramatically go up after 2018 because of BERT, and the word "neural" became popular after 2013 because of the deep learning.

Our intuition is to use a temporal neural network to try to predict biases for words based on historical frequency data of words. This model uses an auto-regressive deep learning model to predict the change over time. We use an auto-regressive RNN-style model, specifically an LSTM, rather than a Transformer model for this because it is more naturally suited to our temporal task, as LSTMs do not use position embeddings.[3] For a balance of simplicity, scalability, and expressivity, we use an LSTM.

We predict the temporal word bias for each year using an LSTM and use it as a feature to bias the generation probability. Figure 2a shows the model overview. Let $f_{iw} \in \mathbb{R}$ be the frequencies of the wth word for the ith year, and let m be the window size which determines how many previous years to consider to predict next year's bias. For each year, we compute a temporal bias $B_{iw} \in \mathbb{R}$ from m previous year's word embedding by using an LSTM where weights are shared across word types. We use the last hidden vector of the LSTM followed by a dot product with a learned vector $A \in \mathbb{R}^d$ to compute the bias:[4]

$$B_{iw} = A^T \, \text{LSTM}(\log(f_{i-m,w}), ..., \log(f_{i-1,w})) \tag{6}$$

This temporal bias is added to the output of the Transformer as a bias in the softmax, as described in §3.1 Equation 3.

3.3 The Temporal Contextual Model

While the previous method models the change in the frequency of words over time, it does not have contextualized information to help it make its predictions. So while it may see words such as "pretraining" increase over time, it is ignoring contextual information in prior abstracts like "pretraining has led to significant performance gains" that could help it make predictions (see Fig. 1a).

To account for contextualized information contained in prior abstracts, we develop a temporally contextualized model. For each word, we create a pooled representation for each year. We use an average of the contextualized embeddings, averaged over all instances of that word over the year. For each word, we then feed the contextualized embedding into an LSTM to predict the temporal word bias. Figure 2b shows the model overview.

In more detail, using our notation from §2, let $d_{ij} = \langle x_{ij1}, x_{ij2}, ..., x_{ijk} \rangle$ be the jth text in ith year where x_{ijk} is kth token in d_{ij}. For the token x_{ijk}, let E_{ijk} be the corresponding contextualized vectors from a pre-trained language model. Our representation for the wth word in the vocabulary for the ith year is the average of the contextualized embeddings, which can be expressed as:[5]

$$V_{iw} = \frac{\sum_{j=1}^{|D_i|} \sum_{k'=1}^{|d_{ij}|} E_{ijk'} I[x_{ijk'} = w]}{\sum_{j=1}^{|D_i|} \sum_{k'=1}^{|d_{ij}|} I[x_{ijk'} = w]} \tag{7}$$

[3]We are aware of work demonstrating autoregressive Transformers can be trained without position embeddings, but we leave this style of model for predicting biases to future work.

[4]To make this more efficient, we batch the LSTM across words in our implementation.

[5]We use the indicator function $I[\cdot]$ which is 1 if the condition is true and 0 if it is false.

To use the temporal contextualized word embeddings, we use the fact that more recent years have more influence on future texts and propose a window-sized modeling approach. The window size determines how many previous years for word embedding we consider to predict the next year's temporal bias. Let m be the window size for each year, then we compute a temporal bias $B_{iw} \in \mathbb{R}^d$ from m previous year's word embedding as follows:

$$B_{iw} = A^T LSTM(V_{(i-m,w)}, ..., V_{(i-1,w)}) \tag{8}$$

where we take the last hidden vector of the LSTM and $A \in \mathbb{R}^d$ is a learnable parameter. The temporal bias is added to the output of the Transformer as a bias in the softmax, as described in §3.1 Equation 3.

We also experimented with combining the word frequency model and the temporal contextual model, but we did not observe any improvement by additively combining them.

3.4 The Doubly Contextualized Model

The temporal contextual model does a good job of predicting the rise and fall in the frequencies of terms. However, we observe that it does a poor job of deciding *when* to use the terms while generating. The `Contextual` output in Table 2 shows an example of this. The model repeatedly introduces new terms that are fashionable, but in an incoherent manner (saying that the paper will focus on IE, but then saying that special attention will be on multi-document summarization).

We hypothesize that the contextual model can predict good terms to use, but cannot decide *when* to rely on the temporal contextual model versus relying on the prior state in the language model (for example, reusing a previous term in the document versus introducing a new fashionable term). The model appears to need a "gating" mechanism to decide when to use the new suggested terms. To address this, we introduce a mechanism that contextualizes the temporal contextual model when generating a document – a *doubly contextualized model* that is contextualized both temporally and in the document generation. Figure 2c shows the model overview.

We start with matching B_{iw} with the pre-trained model embedding space and reduce the dimension of vocabulary size. To implement this, we enable temporal bias B_{iw} to tie with the word embedding layer weights for each word $E_w \in \mathbb{R}^d$ and conduct a linear projection. We compute the tied and projected temporal bias $\tilde{B}_{iw} \in \mathbb{R}^d$ as follows:

$$\tilde{B}_{iw} = (E_w^T B_{iw})A \tag{9}$$

where $A \in \mathbb{R}^d$ is a learnable parameter.

Then we compute the sigmoid attention between transformer decoder output H_k and $\tilde{B}_{iw}$ to obtain the $B_{ikw} \in \mathbb{R}^d$ as follows:

$$B_{ikw} = \alpha\sigma(H_k^T C\tilde{B}_{iw})(E_w^T D\tilde{B}_{iw}) \tag{10}$$

where $C, D \in \mathbb{R}^{d\times d}$ are learnable parameters, and α is a tuned hyperparameter.

This temporal bias is added to the output of the Transformer as a bias in the softmax, as described in §3.1 Equation 4. Using this model, we obtain improved example output shown in Table 2.

4 Experiments: Future Abstract Prediction

4.1 Dataset processing

As a concrete example to be experimented with, we conduct experiments to model future abstracts for NLP papers based on previous papers' abstracts. We first collect paper abstracts for each year from ACL anthology website[6] and filter the noisy abstracts such as papers that are not in English. Then we use the years as the year (for other domains such as news, you can use the day or hour as the year) and split the paper abstracts by years and use abstracts from 2003-2019 as training data, the year 2020 as the development data, and the year 2021 as the test data. Table 1 shows the statistics of the dataset. Figure 3 shows the number of abstracts by year for the dataset.

[6]https://aclanthology.org/anthology+abstracts.bib.gz

Dataset Statistics	Train	Dev	Test
# of abstracts	37816	5919	5529
avg. # of sentences per abstract	9.0	6.4	6.5
avg. # of tokens per abstract	225.8	168.5	164.3

Table 1: Data split statistics

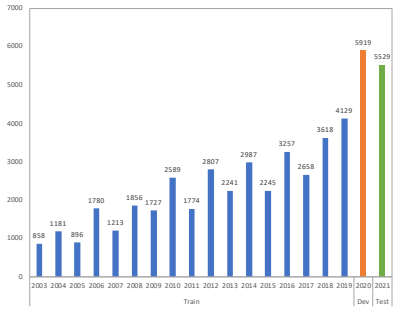

Figure 3: # of abstracts by year

4.2 Models

We use GPT-2 (Radford et al., 2019) as the pre-trained language model in our experiments, although our approach is not restricted to any particular pre-trained language model. We train and evaluate the following models:[7]

- `Baseline` A baseline which fine-tunes GPT-2 on abstracts from all previous years
- `Baseline-n` A baseline which fine-tunes GPT-2 on abstracts from n most recent previous years, since recent years may be more relevant for predicting future years. We evaluated n from 1 to 10, and report the best 2 models ($n = 2$ and $n = 3$).
- `Frequency-NoLSTM` A word frequency model without using an LSTM, instead directly using the previous year's frequency as a bias feature in the model.
- `Frequency` Word Frequency Model (§3.2)
- `Context` Temporal Contextual Model (§3.3)
- `Context`^{2} Doubly Contextual Model (§3.4)

4.3 Hyperparameter Settings

We use the Adam optimizer (Kingma & Ba, 2015). The batch size is set to 2 with gradient accumulation size 2. Between layers, we apply dropout with a probability of 0.1. We fine-tune 10 epochs for each model and do early stopping. The α is set to $1e-3$ or initialized with 1 when automatically learned. Bounds for all hyperparameters are the same as GPT-2. We have several hyperparameter search trials on α which are 1, 1e-1,1e-2, 1e-3, 1e-4, 1e-5. For each model, we have three training and evaluation runs. The method of choosing hyperparameters is based on perplexity scores on the dev set. Fine-tuned RoBERTa Model (Liu et al., 2019) for each year is used to generate temporal word embedding representation. We use beam search decoding with top-k sampling. The beam size is 5, k is 50, and p is 0.92. Since it is topic agnostic, the start token is end of sentence token for GPT-2. All models were trained or evaluated on either one A40 or A6000 GPU. Our implementation is based on Huggingface Transformers (Wolf et al., 2020).

4.4 Automatic Evaluation Metrics

We use three automatic evaluation metrics, which are perplexity (PPL), content perplexity (CPL), and content meteor (CM). Since most of the evolution of ideas in NLP papers is through changes in

[7]Appendix §A gives the model statistics.