Simple and Effective Multi-Token Completion from Masked Language Models

Oren Kalinsky [*]
Amazon
orenk@amazon.com

Guy Kushilevitz [*]
Amazon
guyk@amazon.com

Alex Libov
Amazon
alibov@amazon.com

Yoav Goldberg
Bar Ilan University
yoav.goldberg@gmail.com

Abstract

Pre-trained neural masked language models are often used for predicting a replacement token for a given sequence position, in a cloze-like task. However, this usage is restricted to predicting a single token, from a relatively small pre-trained vocabulary. Recent Sequence2Sequence pre-trained LMs like T5 do allow predicting multi-token completions, but are more expensive to train and run. We show that pre-trained masked language models can be adapted to produce multi-token completions, with only a modest addition to their parameter count. We propose two simple adaptation approaches, trading parameter counts for accuracy. The first method generates multi-token completions from a conditioned RNN. It has a very low parameter count and achieves competitive results. The second method is even simpler: it adds items corresponding to multi-token units to the output prediction matrix. While being higher in parameter count than the RNN method, it also surpasses current state-of-the-art multi-token completion models, including T5-3B, while being significantly more parameter efficient. We demonstrate that our approach is flexible to different vocabularies and domains and can effectively leverage existing pre-trained models available in different domains. Finally, a human evaluation further validates our results and shows that our solution regularly provides valid completions, as well as reasonable correctness for factual-sentence completions.

1 Introduction

Multi-Token-Completion (MTC) is the task of filling masked sentences with a sequence of tokens such that the completed sentence is probable and coherent. e.g., for the masked sentence *"The 46th president of the US, [MASK], was elected in 2020"* a good completion would be any of *"Biden"*, *"Joe Biden"*, *"president Biden"* and more.

While Masked Language Models (MLMs) such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) successfully deal with a simpler variation of this task (single-token completion) these models were pre-trained on a limited vocabulary, not containing multi-word phrases. It is technically possible to complete numerous tokens simultaneously with MLMs by introducing a sequence of [MASK] tokens, but there are no clear methods of conditioning the tokens on each other, and, more importantly, the length of the completion needs to be pre-determined. Expanding MLMs' effective completion vocabulary – the actual vocabulary they are capable of completing well – will help a recent line of work using MLMs to extract knowledge/information from corpora (Jiang et al., 2020; Petroni et al., 2019; Kushilevitz et al., 2020).

Most existing works using MLMs to complete sentences either avoid the problem by limiting the completions to single tokens or use sub-optimal heuristics for MTC (e.g. presetting the length of the sequence and filling one token at a time). A trivial solution is to increase the number of tokens used in the pre-processing tokenization step, and also include multi-word phrases as tokens. However, apart from the longer tokenization time[1], the main problem with this approach is that changing the input level of the model requires adapting the weights of the full MLM to the new input.[2] A recent family of seq2seq pre-trained LMs, the prevalent of which is T5 (Raffel et al., 2020), are trained directly on the MTC task, and indeed perform quite well on it, especially with larger model sizes. However, the seq2seq objective, coupled with very high parameter counts, make such models expensive to train

[*]Authors contributed equally.

[1]Expanding BERT's tokenizer vocabulary from $30K$ to $100K$ phrases leads to a degradation of $\times 30$ in tokenization time. Further details are in Appendix A.

[2]This requires either full pre-training of the MLM or long, end-to-end fine-tuning using a considerable amount of data.

and to run inference on, compared to MLMs.

In this work, we demonstrate how pre-trained MLMs can be *adapted* to produce multi-token completions from (large) fixed vocabularies, using a self-supervised training objective and only a modest parameter count. Specifically, we demonstrate an effective adaptation of pre-trained MLMs to predict, on top of their pre-trained vocabularies, additional $\sim 100K$ noun-phrases (NP-chunks) and entities ranging in length from 1 to 10 tokens. The only requirement is a textual corpus in which each of the desired phrases appears over k times (we used $k = 50$ in this work). The adapted model surpasses the accuracy of T5-3B predictions, while using a fraction of the parameter count and being significantly more efficient to run. Apart from evaluating with automatic measures, we also use human-annotators to explore different aspects of the proposed completions.

After experimenting on the prediction of general-purpose phrases, we also show our methods can be used to predict phrases in a specific domain. This is a significant benefit of the adaptation approach since the completion vocabularies can be tailored to a specific project's needs, using domain-specific pre-trained MLMs, even in domains where huge T5-like models are not available[3].

We use any pre-trained MLM for MTC by extracting the informative representations the MLM uses for completing a single token, and feeding them into a small and simple model that chooses appropriate multi-token completions. We offer two different completion models. Depending on the scenario, both are useful; the first solution, which expands the MLM's decoder matrix, achieves SOTA accuracy. The second solution, using a small iterative generation model, is well suited for large completion-vocabularies while achieving competitive results.

The core reason for the success of our methods, despite being trained on less data and for less time than other solutions, is the fact that the MLM's pre-training is well suited for the MTC task. We provide two types of evidence to support this claim: In section 2 we present an experiment suggesting that MLMs incorporate information regarding multi token phrases. In section 6, we show that pre-training on in-domain data helps our methods perform better. Both of these signals indicate that, as expected, the MLM pre-training captures information required for MTC. Therefore, what is left for our MTC adaptation models is only to extract this information from the MLM, an easier task that does not require a long training with a vast amount of data.

Our main contributions are MTC datasets (general purpose, PubMed-based, and a $3K$ dataset with human labeling), demonstrating that MLMs learn the meaning of Multi Token phrases, and two methods for adapting any pre-trained MLMs for MTC. We publish[4] our datasets, models and code[5].

2 MLMs Learn Multi-Token Phrases

Our work relies on the assumption that MLMs are capable of holding information about multi-token phrases. Despite being a common belief and an essential property for the MLMs to have in order to be as successful as they are, to the best of our knowledge this was not shown explicitly. How do we show that a token-completion MLM is encoding the semantics of multi-token phrases? Looking at completions will fail, because we cannot directly probe for MTC. Instead, we came up with the following experimental design, which measures the encoding of multi-token phrases indirectly by looking at masked positions influenced by them.

We use multi-token phrases that have single token synonyms. We construct a dataset containing quadruples, each quadruple consisting of a multi-token phrase (e.g. *"New York city"*), a single-token synonym (e.g. *"NYC"*), a phrase similar in meaning to the synonyms (e.g. *"Chicago"*) and a random phrase (e.g. *"Dog"*). We collect sentences containing the multi-token phrases, and in each sentence mask out an NP-chunk different from the multi-token phrase (e.g. *"[MASK] is in New York city"*). Next, we replace each multi-token phrase with each of its other corresponding quadruple phrases, forming four similar masked sentences, each containing a different quadruple phrase (e.g. *"[MASK] is in NYC"*).

Finally, we ask an MLM to complete the missing token in each masked sentence, and compare the suggested completions. If the MLM is aware that

[3]For example, an e-commerce company could adapt a pre-trained general purpose MLM to complete multi-token product names, while a biomedical researcher could adapt a pre-trained biomedical MLM such as SciBERT (Beltagy et al., 2019) or BioBERT (Lee et al., 2019) to complete multi-token drug or disease names.

[4]https://registry.opendata.aws/multi-token-completion/

[5]https://github.com/amzn/amazon-multi-token-completion/

the multi-token phrase is semantically similar to the single-token synonym, we expect it to suggest similar completions regardless of which of the two it sees in the sentence. Indeed, the model treats the multi-token phrase similar to the single-token synonym; We use a similarity measure comparing the MLM's suggested completions for the multi-token phrase sentences and each of the other types of sentences. We find that the average similarity between multi-token phrase sentences and single-token synonym sentences is 0.76, while for similar-phrase sentences it is 0.71 and for random-phrase sentences it is 0.63. Our main conclusion from this experiment is that, as expected, MLMs learn semantic meaning of multi-token phrases. Full experiment details (including the similarity measure used) are in Appendix B.

3 Masked Language Modeling Data

In absence of a known dataset for the MTC task, we curate one. We follow (Devlin et al., 2018) and use data from two sources: Wikipedia articles and the Books corpus (Zhu et al., 2015).

Completion Vocabulary. We start by building a vocabulary of phrases. As we aim for general-purpose MTC, and following (Trask et al., 2015) by considering phrases as NP-chunks or entities, we simply focus on phrases that appear frequently in the corpus. We use spacy (Honnibal and Montani, 2017) to extract $64M$ unique NP-chunks and entities. We keep phrases appearing 500 times or more in the corpus, leaving us with $\sim 93K$ phrases. Only 10.2% are single tokens using BERT's cased tokenizer, indicating the importance of MTC. 52.9% of the phrases consist of 2 tokens, 36.9% consist of 3 or more. 53% of the phrases are single-words, 47% span two words or more. Further statistics regarding the number of words and the number of tokens assembling the phrases selected are reported in Table 1.

Masked sentences. We split the corpus using spacy's sentencer. We sample 50 unique sentences containing each vocabulary-phrase. We eliminate recurring sentences and mask out the vocabulary phrase to form masked sentences, using the masked span as the label. We randomly split the data into train (90%), validation (5%) and test (5%) sets.

4 Adapting MLMs for MTC

We propose two simple yet effective solutions, capable of integrating with any pretrained MLM and

#Tokens	%Phrases	#Words	%Phrases
1	$\sim 10.2\%$	1	$\sim 53.3\%$
2	$\sim 52.9\%$	2	$\sim 38.3\%$
3	$\sim 23\%$	3	$\sim 5.4\%$
4	$\sim 9.5\%$	4	$\sim 2.1\%$
5	$\sim 2.8\%$	5	$\sim 0.5\%$
> 5	$\sim 1.6\%$	> 5	$\sim 0.4\%$

Table 1: **Expanded completion vocabulary.** Number of tokens and words assembling the phrases collected.

extending its completion vocabulary to perform MTC. Both solutions utilize the MLM by using the contextual embedding vector of the masked place, which was originally pre-trained to be relevant for completion tasks. Instead of using it to choose the appropriate single token replacement (see Figure 1 (1)) like the MLM, we use it as an input to extension models tuned for completing out of an expanded vocabulary. We train only the relatively small extensions and use the pre-trained MLMs as is, allowing short and effective training.

4.1 Extended-Matrix (EMAT) decoder

Our first extension model is based on extending the decoder matrix of the MLM to include the new multi-token phrases in the completion vocabulary. We assign each new multi-token phrase to an embedding vector, which is added only to the output token-prediction matrix. This is in contrast with simply adding it to the base-model's vocabulary, which results in longer training due to the full model being affected and in increased tokenization time (see Appendix A for tokenization time evaluation). An illustration of the architecture can be found in Figure 1 (2), where we use the MLM to compute the contextual embedding of the masked token and feed it to the extended-matrix decoder. Even though each multi-token phrase is assigned with its own embedding vector, this solution allows us to adapt MLMs trained with a smaller vocabulary (with all mentioned advantages this comes with) to complete phrases that are multi-token phrases in their *original* vocabulary.

During training, we extract the contextual embedding from the masked sentence and only train the extended-matrix decoder yielding a quick training phase. We train the decoder over the train set described in Section 3, expanding the completion vocabulary to $\sim 93K$ phrases. When BERT-base is the base MLM, this approach adds $138M$ parameters, considerably less than models comparable in performance such as T5-3B ($3B$ parameters).

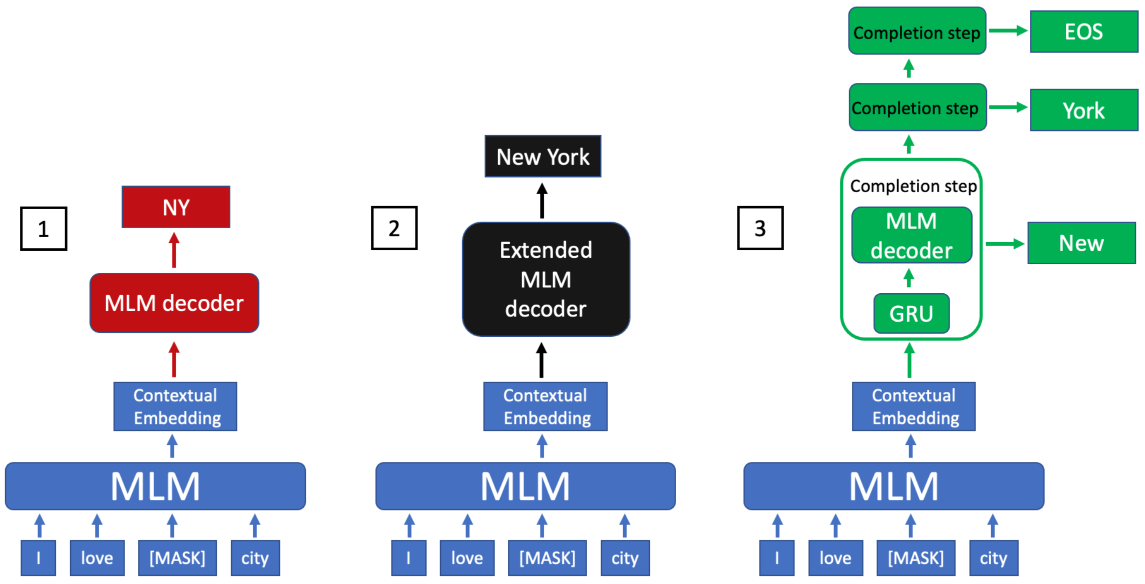

Figure 1: **Architectures.** The MLM body is shared across architectures. It outputs a contextual embedding for the [MASK] token, used by all decoding solutions. The MLM (1) predicts a single token from its vocabulary. EMAT (2) uses an extended decoder to predict a phrase from the extended vocabulary. The Generative extension (3) completes tokens until reaching [EOS]. Some arrows and layers are deducted for readability.

4.2 RNN decoder

Our second model is based on an RNN decoder, specifically a GRU (Cho et al., 2014). We use it to replace the MLM's matrix decoder.

To train our GRU decoder for MTC, similarly to our first solution, we utilize the pretrained MLM by extracting the embedding vector corresponding with the masked token for each sentence. For the GRU-based solution we use this vector as the first hidden vector fed into the GRU. As the first input to the GRU, we wish to provide the context of the multi-token phrase to be generated and thus feed the static embedding of the token *preceding* the [MASK] token. For later steps, the input is the previous token completed by the generative model. The inputs to the GRU are depicted in Figure 1 (3), where the contextual embedding of [MASK] is fed as the first hidden state and the static embedding of 'love' is the first input of the GRU.

Next, at each step we concatenate: a) the output of the GRU; b) the previous vector embedding; and c) the masked token contextual embedding vector from the pretrained MLM, and feed them to a feed forward (FF) layer in order to reduce the dimension. We find that providing these three contexts to the decoder improves its accuracy. The output of the FF is then fed as input to the MLM decoder (For BERT, this is a FF layer followed by an embedding layer and a SoftMax layer) to obtain the token predicted for this step of the autoregressive generation. We train the model to complete the next token of the missing multi-token phrase using a standard back propagation approach with Cross Entropy loss. We utilize the dev set to tune the batch size, number of GRU layers, teacher forcing rate, and learning rate (used parameters can be found in appendix C).

We find that initializing the GRU's decoder embeddings with the same parameters used by the MLM we are plugging into, improves the performance. Also, it helps to pre-train the generative model on a vanilla Language Modeling task before starting MTC training. The idea is to give the fresh extension model some sense of the distribution of the language before the training on the MTC task. An ablation study demonstrating the benefit of these components is shown in Appendix C.

Our generative model consists of only 32M parameters when using BERT as a base model, which are the only trained parameters used for our solution (i.e. the pretrained base MLM is not fine-tuned) and independent of the vocabulary size. During inference, we simply use the MLM to extract the embedding vector corresponding with the masked token and feed it, along with the static embedding of the preceding token, as input to the trained GRU model. Next, we use beam-search to generate sequences until reaching an EOS token.

Model	Coverage	Size	Data size	$a@1$	$a@3$	$a@5$	$a@10$	$a@50$
BERT-cased	13.8%	$110M$	$16G$	$13.5\%/1.8\%$	$21.8\%/3.0\%$	$25.9\%/3.5\%$	$31.2\%/4.3\%$	$43.1\%/5.9\%$
RoBERTa	15.6%	$110M$	$160G$	**19.3%**/3.0%	**30.6%**/4.7%	**36.1%**/5.6%	**43.2%**/6.7%	**59.4%**/9.2%
Naive-MTC-BERT	66.1%	$220M$	$16G$	6.6%/**4.3%**	11.9%/**7.8%**	14.9%/**9.8%**	19.5%/**12.8%**	32.3%/**21.3%**

Table 2: **MLMs performing limited completion.** Coverage is the % of sentences from the test set that the model is capable of completing. MLMs are limited to single token completions. For computational reasons, we allow the Naive-MTC-BERT to generate only single tokens or two-token phrases. Accuracy is reported as x/y; x is the accuracy out of the specific model's limited coverage and y is the accuracy out of the full test set ($coverage * x$).

5 Results

Setup. Measuring success in completion tasks is not trivial since a masked-sentence can have many suitable replacements. In many cases returning the expected phrase in a top-k place, and not necessarily first, is acceptable. Hence, we measure *accuracy@k* (sklearn): the percentage of masked sentences where the label is among the top-k predictions. To better ground our performance expectation from MTC, we first evaluate the accuracy of an easier completion task; Single Token Completion. We go on to evaluate Naive MTC using MLMs. As expected, results are not sufficient. Finally, we evaluate our methods, adapting MLMs for MTC.

5.1 Single Token Completion

Single Token Completion is an easier[6] task than MTC, and MLMs like BERT and RoBERTa are trained directly to perform it. Therefore, results of these MLMs on the single token completion task can be considered as an upper limit for the performance of solutions adapting these models for MTC. We test BERT and RoBERTa on the single token completion task, results of these models on the test set are reported in Table 2.

5.2 Naive MTC with MLMs

A naive way to utilize MLMs for MTC is using a two-step method: given a masked sentence, first predict the number of missing tokens and then duplicate the masked token to the predicted number (e.g. *"The US state [MASK]"* becomes *"The US state [MASK] [MASK]")* and complete them using the MLM. We train a BERT-based classification model predicting the number of missing tokens by assigning a label to the data: for each masked sentence, the label assigned is the number of tokens the masked span splits into using BERT's tokenizer. Due to computational limits of the generation phase, we are only interested in 3 classes: a single token missing, two tokens missing, and 3 or more tokens missing. The class distribution is (10%, 53%, 37%), respectively. We use BERT's default hyper-parameters and train a classification model. This is a task with ambiguous labeling[7], thus we do not expect high accuracy results. The model reaches 64.7% accuracy on the test set.

During inference, we use the model's predictions with a SoftMax function to acquire a probability for a single token mask-replacement and a double token mask-replacement. We utilize these probabilities as follows. For each specific single token, we compute a replacement probability by simply multiplying the probability for any single token replacement by the probability BERT assigns the specific token to replace the mask in the masked sentence. For a double token term we multiply the probability of having a double token replacement by the probability BERT predicts for the specific term, estimated with a standard generating heuristic. After duplicating the mask, we complete the first missing token using the MLM. Then, we replace the first mask with each of the top-100 predicted tokens and complete the second mask. Finally, the probability for each double token term is the product of the probabilities the two tokens assembling it to replace the two missing *[MASK]* tokens.

Results on the test set are reported in Table 2. Even when considering only sentences where the missing phrase consists of one or two tokens, results are not sufficient and call for a different way to use MLMs for MTC. This is likely due to the fact that the first token replacement is computed in an unconditional manner to the second one.

5.3 Multi Token Completion

We use BERT, RoBERTa and SpanBERT as MLMs adapted for MTC with our methods described in section 4. MTC models are capable of completing

[6]For example, BERT's search-space size is $\sim 30K$ while some MTC solutions (e.g. our generation plugin and T5) have an infinite search space.

[7]e.g. for *"The US state [MASK]" "New York"* and *"Texas"* are adequate replacements resulting in different labels.

Model	Size	Data Size	Inf-time	$a@1$	$a@3$	$a@5$	$a@10$	$a@50$
ILM (GPT-2)	**124M**	$1G/41G$	120ms	1.3%	2.7%	3.7%	5.3%	10.8%
T5-base	$220M$	$700G$	224ms	5.2%	8.3%	9.7%	11.4%	15.6%
T5-3B	$3B$	$700G$	802ms	11.6%	18.5%	21.5%	25.4%	NA
BERT-Adapted-RNN	**32M**$/142M$	**1G/17G**	44ms	9.24%	14.64%	17.39%	21.32%	31.28%
BERT-Adapted-EMAT	$138M/248M$	**1G/17G**	**15ms**	**12.64%**	**20.48%**	**24.63%**	**30.65%**	**45.85%**
ROBERTA-Adapted-RNN	$48M/158M$	$1G/161G$	59ms	8.14%	12.99%	15.51%	19.07%	28.40%
ROBERTA-Adapted-EMAT	$161M/271M$	$160G/161G$	16ms	11.59%	19.06%	23.18%	29.27%	45.79%
SPAN-BERT-Adapted-RNN	**32M**/142M	**1G/17G**	46ms	8.29%	12.82%	15.07%	18.34%	26.334%
SPAN-BERT-Adapted-EMAT	$138M/248M$	**1G/17G**	**15ms**	6.91%	11.6%	14.12%	17.93%	27.45%

Table 3: **MTC results.** Models are described in Section 5.3. Adapted models are trained with our solutions as described in Section 4. Size is the number of parameters; Data size is the amount of data seen during training. For adapted models we report both of these numbers excluding/including the base MLM. Inf-time is the average inference time of a single sentence, measured using an Nvidia T4 GPU. T5-3B is too large for a beam of size > 10.

a phrase of any length hence have a coverage of 100% on the test set. Apart from our methods, we also report on several baselines.

ILM (Donahue et al., 2020) is a GPT-2 based framework for infilling, a task similar to MTC. The LM is shown a sentence with missing places and is trained to generate text probable of replacing the missing parts. A crucial observation is that GPT-2 is an LM and not an MLM. Therefore, ILM's pretraining seems less appropriate for completing relatively short phrases (a task similar to the MLM objective), and more appropriate for longer generation tasks (a task more similar to the LM objective). We fine-tune ILM on our dataset for a single epoch, taking ~ 3.5 days using an Nvidia T4 GPU.

T5(Raffel et al., 2020) is a transformer model pretrained on the MTC task. It is a strong baseline reaching SOTA on many NLP tasks. We report on two versions, T5-base and T5-3B, both trained on a dataset 1-2 order of magnitudes larger than ours.

Results and Discussion. Results are reported in Table 3. Our extended matrix (EMAT) solution performs best, even when compared to the huge T5-3B. The RNN plugin is slightly inferior, but still performs better than existing solutions comparable in size. Note that even though the size of the EMAT plugin is $\sim \times 4$ of the size of the RNN plugin, inference time for the tested vocabulary is shorter because of GPU optimizations and the fact that the RNN model may run several times ($\sim$ 3 in average on the test set) for each completion. However, increasing the completion vocabulary size will not affect the GRU size but will increase the size of the matrix extension, making the RNN plugin a better fit for large completion vocabularies[8].

6 Domain Specific MTC

We investigate MTC in a specific domain.

Datasets. We chose the Biology Domain, due to the availability of data and models. Using PubMed (pubmed) abstracts as a corpus, we extract two MTC datasets. Similarly to section 3, we first construct the completion vocabularies.

Key-phrase vocabulary: we use $\sim 20K$ MeSH vocabulary[9] phrases appearing frequently as key-phrases in pubmed papers. This assures us phrases from the Biology domain. We discard phrases appearing less than 50 times as NP-chunks or entities in the corpus, leaving us with $\sim 12.5K$ phrases.

Frequent-phrases vocabulary: Similarly to section 3, we extract a vocabulary of phrases appearing more than 500 times as NP-chunks or entities in the corpus ($\sim 144K$ phrases). To make sure we are considering mostly phrases from the Biology domain, we discard phrases appearing in our general purpose vocabulary (described in section 3), leaving us with $\sim 118K$ phrases.

Finally, for both vocabularies we extract 50 sentences containing each of the vocabulary phrases, mask the phrase out in each sentence and split the data into train, development and test sets.

Base Models. To evaluate the impact that the domain-specific pre-training has on the performance of our methods, we test our adaptation methods on top of models pretrained on different datasets. *SciBERT* (Beltagy et al., 2019) is a BERT-like model trained on the Semantic Scholar data - data from the scientific domain, closer to the biology domain than the general purpose BERT. *BioBERT* (Lee et al., 2019) is a BERT-like model trained on PubMed abstracts.

[8]For a completion vocabulary with $1M$ phrases and BERT as the MLM, the size of the RNN plugin remains $32M$, while the size of the EMAT plugin grows to $769M$.

[9]MeSH (Medical Subject Headings) is NLM's controlled vocabulary of biomedical terms used to describe the subject of each journal article in MEDLINE.

Model	Dataset	$a@1$	$a@3$	$a@5$	$a@10$	$a@50$
T5-base	*key-phrases*	3.6%	5.2%	5.8%	6.7%	8.4%
T5-3B	*key-phrases*	10.3%	15.1%	16.9%	19.5%	NA
BERT-Adapted-EMAT	*key-phrases*	12.9%	20.7%	24.9%	31.1%	48.8%
SciBERT-Adapted-EMAT	*key-phrases*	13.4%	21.2%	25.2%	30.8%	44.9%
BioBERT-Adapted-EMAT	*key-phrases*	**16.3%**	**26.9%**	**32.1%**	**39.5%**	**57.2%**
T5-base	*NP-chunks and Entities*	3.9%	5.7%	6.6%	7.7%	10.5%
T5-3B	*NP-chunks and Entities*	**8.2%**	12.8%	15.0%	18.1%	NA
BERT-Adapted-EMAT	*NP-chunks and Entities*	7.1%	11.8%	14.4%	18.5%	30.5%
SciBERT-Adapted-EMAT	*NP-chunks and Entities*	**8.2%**	**14.2%**	**17.5%**	**22.5%**	**35.4%**
BioBERT-Adapted-EMAT	*NP-chunks and Entities*	8.0%	13.5%	16.4%	20.7%	31.77%

Table 4: **MTC on the pubmed test sets**. T5-3B is too large for a beam of size > 10.

Results are reported in table 4. For brevity, we compare our Extended Matrix method, tuned on three base models (BERT, SciBERT, BioBERT), to the two primary baselines: T5-base and T5-3B. We report on the two PubMed datasets we curated.

Conclusions. First, our method outperforms the strong T5-3B baseline in the domain-specific scenario as well, showing MLMs can be effectively adapted for domain-specific MTC. Second, the success of *BioBERT* and *SciBert* suggests that the pretraining of the MLM is in fact utilized by our methods (i.e. there is no catastrophic forgetting). Last, the *key-phrases* dataset is easier than the *frequent NP-chunks and Entities* one. This is likely due to the *key-phrases* dataset mostly containing phrases that are more frequent and significant.

7 Human Evaluation

Completion tasks are difficult to evaluate since a sentence can have many different valid completion options including, but not limited to, the original masked span. We use *accuracy@k* to deal with this issue, but in some use-cases only the first completion is important. Thus, we define a manual task to more accurately evaluate sentence completions.

Human-annotators are presented with the masked sentence, the original masked span and the *first* completion suggested by each of the methods we evaluate (full annotation task is in Appendix D). Annotators were not aware which model provided each completion, and the completions were presented in a random order for each sentence. The sentences are divided between three expert English-speaking annotators. Prior to performing the annotations, the annotators met for a calibration session, each annotating the same 50 sentences and discussing results until reaching high agreement. We sample 750 sentences from the test set described in section 3. Each expert annotator is assigned with 250 sentences and the completions of each of the four methods, amounting to 1000 samples each.

First, we ask annotators whether the completions are grammatically correct and make sense in the context of the sentence, *regardless of whether they are factually correct* (denoted as a "Valid" completions). For example, any year or location is a valid completion for *"Jules Verne was born in [MASK]"*.

In some cases, valid completions can be general and uninformative. For example, in the sentence *"On January 1, 1998, [MASK] was released publicly online as SGI freeware."* the original span is *"Blender"*. While any SGI freeware name would make sense, the word *"it"* (suggested by T5-3B) is also a valid completion. This happens with general words like *"he"*, *"she"*, or *"person"* but can also occur with entities, e.g. in the sentence *"Jules Verne was born in [MASK]"*, the original masked span is *"Nantes"* and a suggested completion might be *"Europe"*. We ask annotators to flag these cases, where the suggested completion is more general (less specific) than the original span.

Factual Correctness of completions is also important, since some works use completion methods for knowledge extraction (Petroni et al., 2019; Jiang et al., 2020). To evaluate this, we first ask annotators to mark whether the masked span is a part of a specific fact. For example, *"MLK, (born [MASK])"* should be marked as a fact, while *"he died aged [MASK]"* shouldn't. We find that 50.93% of the masked sentences are part of a fact[10]. For the factual sentences, we ask the annotators to label whether the proposed completion is factually correct. If they do not know, annotators are instructed to use the web to verify the correctness.

Results are reported in Table 5. As expected, the actual valid-completion percentage of the

[10]Recall that the evaluation data originates from both Wikipedia and books.

Model	Valid %	Valid Specific%	Correct %	Correct Specific %
ILM	52.2%	47.8%	9.4%	7.5%
T5-base	66.0%	49.0%	29.8%	18.5%
T5-3B	81.8%	66.8%	**40.8%***	30.1%
BERT-Adapted-EMAT	**84.1%**	**80.4%***	32.9%	**31.6%**

Table 5: **Human evaluation results.** Valid completions are ones that are grammatically correct and make sense. Specific completions are completions that where not annotated to being more general than the original masked span. Correct is the percentage of sentences labeled as facts that are valid completions and are also factually correct. * marks that the difference between best and second best is statistically significant.

first completions is much higher than the *accuracy@1* measured. An important observation is that while checking for valid completions using human-evaluation is obviously a more accurate measure than *accuracy@k*, the two are correlated. Models that perform well in one measure, do so also in the other. This means *accuracy@k* is a good proxy for the actual valid-completion measure. Specifically, our method performs impressively when annotating for validness as well, slightly better than the much larger T5-3B. Finally, it seems general purpose models like T5 tend to complete general phrases more than our method. This is likely due to the fact that during our MTC training, the model sees the same amount of examples for each phrase. Other methods see more examples of general phrases, since these are more common in the language.

As for factual correctness; while T5-3B completes more facts correctly than our methods, when eliminating cases where the completion is general[11] our method performs better. This is important since general completions are not as interesting and can be more easily acquired. It can be especially crucial for knowledge extraction methods using these models. For example, a method trying to extract presidents with the sentence *"US presidents such as [MASK]"* would benefit from completions like *"Obama"* or *"Trump"* and not *"the president"* or *"this person"* which are correct, but uninformative.

8 Related Work

Transformer-based LMs and MLMs (Peters et al., 2018; Devlin et al., 2018) have revolutionized NLP in the past couple of years. While most of the impact has been achieved using these pretrained models as a source of meaningful contextual embeddings, recent works are using these models for the task they were pretrained for: Masked Language Modeling (Petroni et al., 2019; Kushilevitz et al., 2020; Lazar et al., 2021; Shani et al., 2021; Jiang et al., 2020).

While most MLMs are capable of completing multiple missing tokens simultaneously, they do so in an unconditional manner, yielding unsatisfying results. Therefore, some works using MLMs for Masked Language Modeling simply restrict themselves to single token completions (Petroni et al., 2019) while others use heuristics in order to generate multi token completions (Lazar et al., 2021; Jiang et al., 2020).

Some attempts towards making MLMs predict multi token units better by changing the masking technique during training include masking spans instead of tokens (Joshi et al., 2020) and masking whole words (Cui et al., 2019), but these methods still complete in an unconditioned manner during inference. XL-net (Yang et al., 2019) avoids interdependence between masked tokens using a Permutation Language Modeling pretraining objective.

Similarly to our solution, extending the decoder matrix with n-gram vocabulary items is done also by Xiao et al. (2020), but for a different purpose: they use them as an auxiliary signal during training, in order to improve BERT's single-token pretraining. They mask n-grams, and predict a specific n-gram assigned id *together with the single tokens assembling the n-gram*, as to improve the single-token embeddings. The BERT model is pretrained end to end and the n-gram embeddings are discarded after pretraining and are not part of their final model used during fine-tuning and inference. While we share their extension of the decoder matrix, the MTC task adds additional requirements: to allow for easy MTC vocabulary support, we aim for a short MTC training and inference, building on the preexisting pretrained MLMs and avoiding

[11]For example, for "Garson accepted the role, winning [MASK]." original span is "the Academy Award for Best Actress". T5 completes "an academy award"- factually correct, but more general than the original span. Our method tries to pinpoint a specific award - a harder task. It completes "the Academy Award for Best Picture" which is factually wrong.

the need to pretrain from scratch. In addition, our solution does not distinguish between multi-token and single token-phrases during training since both are part of the completion vocabulary.

In Donahue et al. (2020), the ILM framework is introduced. This is a framework for infilling, a task similar to MTC. ILM is a GPT-2 (Radford et al., 2018) based solution, meaning the pretrained model used is a LM and not an MLM, requiring it to adapt to a different task. Finally, T5 (Raffel et al., 2020) was pre-trained for the MTC task using a vast amount of data. It performs well but requires a lot of training data, time and memory, does not utilize existing MLMs and is not available in many domains and languages.

9 Conclusions

We show MLMs can be adapted for multi token completion even though they were trained for single token completion. We presented two simple but effective solutions that leverage the pretrained MLMs and offer quick adaptations to new vocabularies. The two solutions are trading-off performance and size: 1) an extended matrix decoder offering SOTA accuracy but size-dependent on the completion vocabulary; 2) an RNN decoder with slightly lower accuracy but size independent of the vocabulary size. We also demonstrate the flexibility of our approach to different vocabularies and domains by evaluating it on the PubMed dataset and showing that leveraging domain-pretrained MLMs offers significant accuracy improvement. Finally, we validate our results by conducting a human evaluation to account for valid completions that are not measured using our automatic metric. It shows that our extended matrix solution provides valid completions in 84% of the times, and can also correctly handle facts in 30% of the times, comparatively to the much larger T5-3B.

10 Limitations

The main limitation of our work is that it requires a fixed and pre-determined completion vocabulary. We acknowledge that this is a burden, and in some cases such a vocabulary might not be available. We believe a solution for adapting MLMs for MTC without such a prerequisite is feasible, and this is a goal for future work.

11 Ethical Considerations

Human Evaluation. In terms of fair-pay, the payment to the expert annotators was above the minimum wage in the US. Consent was given from the annotators to use their annotations and release them as part of this research. The annotators were not aware which solution generated each completion and the completions were presented in a random order as to avoid bias based on the order. To achieve high quality results, the annotators had a calibration session as to better understand the guidelines and requirements described in Appendix D.

Environmental. Compared to the massively large language models such as T5-3B, our models are lightweight and can be run on smaller more energy efficient hardware such as a CPU. In addition, this becomes more apparent when considering the superior inference-latency of our solutions. Since energy is composed of both time and power-consumption, our lightweight models should waste significantly less energy during inference.

Biases and Exclusion. The proposed models depend on a fixed and pre-determined vocabulary of potential multi-token completions, and the choice of this set in itself may result in omissions, exclusions under-representation of some groups or concepts, and over-representation of others. Care should be taken to select a set that alleviate such biases to the extent possible. Also after the selection of the set, the algorithm does not guarantee balanced, fair or unbiased selections of candidate completions. Users should be aware of this when designing algorithms whose predictions may influence certain groups.

12 Acknowledgements

The work of Yoav Goldberg was supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 802774 (iEXTRACT).

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3613–3618. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese BERT. *CoRR*, abs/1906.08101.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Lee Raymond Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2492–2501. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020. X-FACTR: multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5943–5959. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Guy Kushilevitz, Shaul Markovitch, and Yoav Goldberg. 2020. A two-stage masked LM method for term set expansion. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6829–6835. Association for Computational Linguistics.

Koren Lazar, Benny Saret, Asaf Yehudai, Wayne Horowitz, Nathan Wasserman, and Gabriel Stanovsky. 2021. Filling the gaps in ancient akkadian texts: A masked language modelling approach. *CoRR*, abs/2109.04513.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

pubmed. pubmed.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners. *none*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Chen Shani, Nadav Borenstein, and Dafna Shahaf. 2021. How did this get funded?! automatically identifying quirky scientific achievements. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 14–28. Association for Computational Linguistics.

sklearn. accuracy at k.

T. Sørenson. 1948. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. Biologiske skrifter. I kommission hos E. Munksgaard.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *CoRR*, abs/1511.06388.

Dongling Xiao, Yu-Kun Li, Han Zhang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gram: Pre-training with explicitly n-gram masked language modeling for natural language understanding. *CoRR*, abs/2010.12148.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

Multi	Single	Similar	Random
new york city	nyc	chicago	disco
fish tank	aquarium	zoo	lena
every week	weekly	monthly	fruit
extort	blackmail	scam	cat
barman	bartender	waitress	mckenzie

Table 6: Experiment data sample. **Multi** is the multi-token phrase, **Single** is the single-token synonym, **Similar** is a term close in the sense2vec space and **Random** is a random single-token word.

A Tokenizing an expanded vocabulary

We experiment to see how tokenizing time is affected by larger vocabularies. We find that including multi word phrases as tokens actually has a worse effect then just adding more tokens, since the tokenizer cannot assume a word is the maximal span for each token. We sample 10K random sentences from Wikipedia and use huggingface's[12] implementation for BERT's standard cased tokenizer. When using the original vocabulary (of size $\sim 30K$) the tokenization takes 4.23 seconds. When expanding the vocabulary to our collected vocabulary (of size $\sim 100K$, including multi word phrases) the tokenization time jumps to 119.26 seconds using the same machine.

B MLMs Learn Multi-Token Phrases

In section 2, we report an experiment illustrating that MLMs are capable of treating multi-token phrases properly. The purpose and outline of the experiment are described in 2, further details are provided in this Appendix. BERT-base is the MLM used.

B.1 Data collection

We curate a dataset of multi-token phrases that have single-token synonyms. We start from Wordnet (Miller, 1995) synonyms, keep only synonyms where one phrase is a single-token and the other is a multi-token phrase using BERT's tokenizer[13], and manually select synonyms which are interchangeable. We collect 100 such synonyms. To each pair of synonyms we add a similar phrase chosen as the phrase closest to the single-token synonym in the sense2vec[14] vector-space which is manually verified as not a synonym of the pair, and not slang or an inappropriate phrase[15]. Finally, to each triplet we add a random single-token word forming a quadruple. A sample of the dataset is shown in Table 6.

B.2 Experimental setup

For each quadruple in our dataset, we conduct the following experiment: We collect k sentences containing the multi-token phrase from Wikipedia. For example, given the quadruple *("fish tank", "aquarium", "zoo", "lena")*, the multi-token phrase is *"fish tank"* and one such sentence is *"nemo is placed in a fish tank in a dentist's office."*. We use each of the collected sentences to compute a similarity score between the phrases as follows.

Masked sentence per quadruple term. We first form a masked sentence for the multi-token synonym, by masking out a random NP-chunk which is not the multi-token phrase (e.g. *"[MASK] is placed in a fish tank in a dentist's office."*). Then, we form a masked sentence for each of the other quadruple terms by replacing the multi-token phrase with it (e.g. for the phrase *"aquarium"* we form the sentence *"[MASK] is placed in a aquarium in a dentist's office."*)

Similarity between masked sentences. We use the MLM to compute similarity between masked sentences. Following (Kushilevitz et al., 2020), for two masked sentences (m_i, m_j) we query the MLM to complete the mask in each and compare the predicted completions using the Sørenson-Dice coefficient (Dice, 1945; Sørenson, 1948):

$$sim(m_i, m_j) = \\ |top_q(MLM(m_i)) \cap top_q(MLM(m_j))|/q$$

Where $MLM(m)$ is the list of tokens proposed by the MLM to complete the mask in the masked-sentence m, ranked by their probability. $top_q(MLM(m_i))$ is the top q tokens in the list (q being a parameter). An example for the similarity measure process for a single sentence is shown in Table 7.

[12]https://huggingface.co/transformers/model_doc/bert.html#berttokenizer. We used the transformers package version 4.12.3.

[13]Multi-token phrases are not necessarily multi-word phrases. Uncommon words are also split to multiple tokens.

[14]Sense2vec (Trask et al., 2015) is a twist on the word2vec algorithm.

[15]Sense2vec is trained on data from Reddit, which yields phrases that are slang or inappropriate. These will probably not be significantly found in BERT's training data (Wikipedia and Books) and are therefore filtered out for this experiment.

Phrase type	Phrase	Masked sentence	MLM-suggestions	Intersection	Similarity
Multi-token synonym	fish tank	"a glass fish tank is sufficient for keeping [MASK]."	1.'**fish**', 2.'**water**', 3.'**ducks**', 4.'**trout**',..., 8.'**salmon**', ... , 13.'**eels**',..., 17.'**turtles**',...	50	1
Single-token synonym	aquarium	"a glass aquarium is sufficient for keeping [MASK]."	1.'**animals**', 2.'**fish**', 3.specimens', ,..., 8.'**turtles**', 9.'**ducks**', ..., 15.'**water**',..., 31.'**eels**',...	33	0.66
Similar phrase	zoo	"a glass zoo is sufficient for keeping [MASK]."	1.'**animals**', 2.'**birds**', 3.cats', ,..., 8.'**ducks**', 9.'pigeons', ..., 30.'goats',..., 37.'lions',...	20	0.4
Random token	lena	"a glass lena is sufficient for keeping [MASK]."	1.'warm', 2.'dry', 3.**balance**', 4.'**water**',..., 7.'**it**', ..., 20.'safe',..., 46.'records',...	17	0.34

Table 7: **Similarity measure example.** Original sentence found in the corpus is *"a glass fish tank is sufficient for keeping tarantulas."*. MLM-suggestions shown are a sample of the top 50 (we use $q = 50$) suggestions for mask completions using BERT. The bold suggestions are ones that appear in the top-50 suggestions for the multi-token synonym sentence. Intersection is the size of the intersection between the top-50 tokens suggested for the sentence and the top 50 tokens suggested for the multi-token synonym sentence. Similarity is $intersection/q$.

hyperparameter	RNN-based method	EMAT method
Batch size	128	128
Learning rate	$1e^{-3}$	$1e^{-3}$
Epochs	10	2
GRU layers	2	-
Teacher forcing	0.5	-
Dropout	0.2	-

Table 8: **Hyperparameters.**

Similarity between terms. We define similarity between two terms as the average similarity across all pairs of masked sentences containing them.

B.3 Experiment results

For each quadruple, we compute the similarity between the multi-token phrase and the other quadruple terms. We use $k = 100$ (number of sentences for each quadruple) and $q = 50$ (number of top tokens considered for the similarity measure). For 82 out of the 100 quadruples in the dataset, the single-token synonym is the most similar to the multi-token phrase. 15 times the most similar is the similar phrase and only 3 times the most similar is the random term, showing the effectiveness of the similarity measure we use. The multi-token phrase is most similar to the single-token synonym: the average similarity measured between them is 0.766 while for the similar phrase it's 0.715 and for the random phrase it's 0.632. Results show the MLM treats multi-token phrases similarly to their single-token synonyms. This corroborates the assumption that the MLM is capable of storing information about multi-token phrases.

Version	$a@1$	$a@5$	$a@50$
lm_pretrain_unshared	7.79%	15.14%	28.10%
no_pretrain_shared	8.99%	16.67%	29.11%
lm_pretrain_shared	**9.31%**	**17.15%**	**30.27%**

Table 9: Ablation study. The results shown use BERT as the Base MLM.

C Model Details

In Section 4, we briefly describe our adaptation solutions. Herein, we describe the exact hyperparameters used and an ablation study of the RNN-based method. Table 8 presents the hyperparameters that were selected after tuning on the dev set. The hyperparameters had a larger impact on the RNN-based solution than the extended matrix (extended-matrix) solution.

In addition, we test to check the effectiveness of two components in our RNN-based solution. The First is pre-training the GRU on a vanilla Language Modeling task, we do this in order to give the model some sense of understanding of the language distribution before training it on the MTC task. The second component is sharing the embedding layers between the original MLM and the added completion GRU. As seen in Table 9 both of these are shown to help.

D Manual Annotation Instructions

In this task, you are given a sentence with a missing phrase (marked as ___), a correct completion (which is not necessarily the only possible correct completion) and a proposed completion. The completed sentence is the sentence that is formed by planting the proposed completion in the location of the missing phrase. Please answer the following questions:

1. Is the completed sentence grammatically correct and does it make sense? In this question, ignore the factual correctness of the completion. For example, in the sentence *"Barack Obama was born in ___"*, any place or date is a valid completion.

2. Is the proposed completion more general than the given correct completion? For example, the word *"he"* is more general than a specific name. The phrase *"North America"* is more general than the phrase *"New York"*. The phrase *"Los Angeles"* is not more general than the phrase *"New York"*.

3. Is the missing phrase in the context of the sentence a part of a specific fact (geographical, physical, mathematical, etc.)? For example, in the sentence *"Barack Obama was born in ___"* the missing phrase is a part of a fact. In the sentence "*He was born in ___"* the missing phrase is not a part of a specific fact because *"He"* may refer to many different people.

4. If the answer to question 3 is yes, is the completed sentence factually correct? If you are not sure, please use the web to verify.