Nonparametric Masked Language Modeling

Sewon Min[1,2] **Weijia Shi**[1,2] **Mike Lewis**[2] **Xilun Chen**[2]
Wen-tau Yih[2] **Hannaneh Hajishirzi**[1,3] **Luke Zettlemoyer**[1,2]
[1]University of Washington [2]Meta AI [3]Allen Institute for AI
{sewon,swj0419,hannaneh,lsz}@cs.washington.edu
{mikelewis,xilun,scottyih}@meta.com

Abstract

Existing language models (LMs) predict tokens with a softmax over a finite vocabulary, which can make it difficult to predict rare tokens or phrases. We introduce NPM, the first nonparametric masked language model that replaces this softmax with a nonparametric distribution over every *phrase* in a reference corpus. NPM fills in the [MASK] solely from retrieving a token from a text corpus. We show that NPM can be efficiently trained with a contrastive objective and an in-batch approximation to full corpus retrieval. Zero-shot evaluation on 16 tasks including classification, fact probing and question answering demonstrates that NPM outperforms significantly larger parametric models, either with or without a retrieve-and-generate approach. It is particularly better at dealing with rare patterns (word senses or facts) and predicting rare or nearly unseen words (e.g., non-Latin script). We release the model and code at github.com/facebookresearch/NPM.

1 Introduction

Current large language models, despite their wide use and impressive performance, are expensive to scale, difficult to update, and struggle with long-tail knowledge and patterns (Kandpal et al., 2022). Recent work follows a retrieve-and-generate approach to partially address these issues (Lewis et al., 2020; Izacard et al., 2022); however, their final predictions are still made by a parametric model. In particular, they still include a softmax over a finite vocabulary, which limits expressivity (Yang et al., 2018; Pappas et al., 2020) and can make them reluctant to predict rare or unseen tokens (e.g., *Thessaloniki* in Figure 1).

In this paper, we introduce NPM, the first **N**on**P**arametric **M**asked Language Model that predicts tokens solely based on a nonparametric distribution over *phrases* in a text corpus (Figure 1). NPM consists of an *encoder* that maps the text into a fixed-sized vector, and a *reference corpus* from which NPM retrieves a phrase and fills in the [MASK]. It, crucially, does not have a softmax over a fixed vocabulary, but instead has a *fully nonparametric* distribution over phrases. This is in contrast to a recent body of work that incorporates nonparametric components in a parametric model (Borgeaud et al., 2022; Izacard et al., 2022; Zhong et al., 2022b).

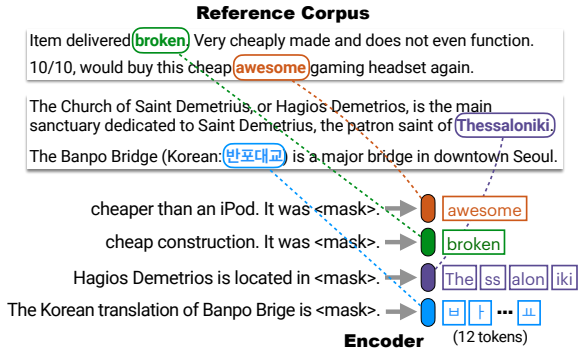

Figure 1: An illustration of NPM. The *encoder* maps a masked sentence into a dense vector, and retrieves the nearest phrase from a *reference corpus*. NPM can fill in the [MASK] with multiple tokens, e.g., *Thessaloniki* (4 BPE tokens) and unseen words, e.g., 반포대교 (12 BPE tokens).

Training such a nonparametric model introduces two key challenges: (1) full corpus retrieval during training is expensive, and (2) learning to predict an arbitrary length phrase without a decoder is non-trivial. We address the first challenge by using in-batch approximations to full corpus retrieval (Wu et al., 2020; Zhong et al., 2022b), and the second by extending span masking (Joshi et al., 2020) and a phrase-level contrastive objective (Oord et al., 2018; Lee et al., 2021).

We perform zero-shot evaluation on 16 tasks including classification, fact probing and question answering. They include temporal shift and word-level translation tasks that highlight the need to predict new facts or rare phrases. We compare with

a range of competitive baselines including encoder-only (Liu et al., 2019), encoder-decoder (Raffel et al., 2020), and decoder-only models (Zhang et al., 2022; Brown et al., 2020). We also compare with a retrieve-and-generate approach that feeds a concatenation of the input and passages to parametric models using off-the-shelf retrieval. Results show that NPM is significantly more parameter-efficient, outperforming up to 500x larger parametric models and up to 37x larger retrieve-and-generate models. It is particularly good at (1) predicting rare words (e.g., an entity split into multiple BPE tokens such as *Thessaloniki*) and (2) disambiguating word senses (e.g., *cheap* may indicate *inexpensive* or *of very poor quality*; Figure 1). Finally, our evaluation on an entity translation task demonstrates that NPM can predict a word consisting of characters that are extremely rare if not unseen (e.g., non-Latin script; Figure 1).

In summary, our contributions are as follows.

1. We introduce NPM, the first nonparametric masked language model that fills in the `[MASK]` solely from a phrase-level nonparametric distribution over a corpus.
2. We introduce a novel training scheme to train NPM on unlabeled data. We completely remove the softmax over the output vocabulary, enabling an effectively unbounded output space by predicting any n-gram.
3. Zero-shot evaluation on 16 downstream tasks shows that NPM outperforms significantly larger parametric models, are better on rare patterns, scale well, can be efficiently updated at test time, and can predict extremely rare if not unseen tokens (e.g., words in non Latin script).

2 Related Work

Language Models (LMs). Large LMs trained on a vast amount of text are shown to perform a wide range of downstream tasks in a zero-shot manner by converting a task into a cloze format (Radford et al., 2019; Brown et al., 2020). This is possible because a variety of knowledge is encoded in the parameters of the models. Recent work has scaled parametric LMs by adding more parameters (Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2022) which can be very expensive in practice. Moreover, such models struggle with predicting rare words or entities, and cannot be updated over time.

There has been a recent body of work that incorporates the nonparametric component with a parametric LM. We distinguish (1) work that concatenates retrieved text to the input and trains the model with a standard LM objective (Borgeaud et al. (2022); Izacard et al. (2022); so-called retrieve-and-generate approaches) from (2) work that retrieves tokens from a large text corpus to estimate a probability distribution that is interpolated with the output distribution from a standard LM (Khandelwal et al. (2020); Yogatama et al. (2021); Zhong et al. (2022b); Lan et al. (2023); so-called kNN models). Our work is closely related to such a line of work and can be seen as an extreme version of the kNN approach with no interpolation. However, our work is the first that models a *fully* nonparametric distribution by entirely removing the softmax over a finite vocabulary. This offers a range of new functionalities, such as modeling a distribution over phrases, or predicting rare or unseen words.

Bottleneck in softmax. Most if not all language models use a softmax function that gives a categorical probability distribution over a finite vocabulary. Yang et al. (2018) showed that this softmax is a low-rank approximation of a high-rank output space, making the model less expressive. Pappas et al. (2020) discussed that a fixed output vocabulary makes language models resistant to adaptation to new domains and tasks. We share the motivation with such prior work and propose to use a nonparametric output space to address these issues. Moreover, although not explicitly explored in this paper, our work that completely removes the softmax over the vocabulary can make training more efficient, especially when the vocabulary is large (e.g., multilingual models (Conneau et al., 2020)).

Nonparametric models. In nonparametric models, the data distribution is not defined by a fixed set of parameters, but is rather a function of the available data (Siegel, 1957; Hollander et al., 2013). Having complexity that grows as the data grows, they are differentiated from parametric models whose complexity is bounded as a priori. Freeman et al. (2002) noted that the term nonparametric does not imply that they have no parameters, but rather that the number and nature of the *effective* parameters are flexible and can depend on the data.

Recent work in NLP has explored nonparametric inference without training (Khandelwal et al., 2020; He et al., 2021; Xu et al., 2022), or trained the nonparametric model on the labeled data for a specific downstream task (Seo et al., 2018, 2019; Lee et al., 2021). In contrast, our work trains a fully

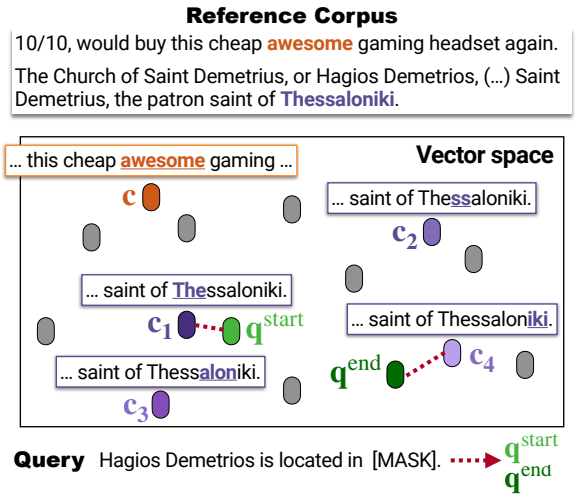

Figure 2: Inference of NPM (Section 3.1). Each token in the reference corpus $\mathcal{C}$ is mapped into a dense vector space. At test time, a query is represented as two vectors, $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, each in the same vector space. We use a nearest neighbor search to retrieve the start and the end of the phrase using $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, respectively.

nonparametric language model without the labeled data and performs a range of tasks zero-shot.

3 Method

We introduce **NPM**, the first **N**on**P**arametric **M**asked Language Model. NPM consists of an encoder and a reference corpus, and models a nonparametric distribution over a reference corpus (Figure 1). The key idea is to map all the phrases in the corpus into a dense vector space using the encoder and, when given a query with a `[MASK]` at inference, use the encoder to locate the nearest phrase from the corpus and fill in the `[MASK]`.

Encoder-only models are competitive representation models (Patel et al., 2022), outperforming the other two classes of models in classification tasks (Section 5.4). However, existing encoder-only models are unable to make a prediction whose number of tokens is unknown, making their use cases limited without fine-tuning. NPM addresses this issue, since it can fill in the `[MASK]` with an arbitrary number of tokens by retrieving a *phrase*.

We first describe inference of NPM assuming a learned encoder (Section 3.1), and then describe how we train the encoder to map the text into a good vector space (Section 3.2).

3.1 NPM: Inference

Overview. The encoder maps every distinct *phrase* in a reference corpus $\mathcal{C}$ into a dense vector space. At test time, the encoder maps the masked query into the same vector space and retrieves phrases from $\mathcal{C}$ to fill in the `[MASK]`. Here, $\mathcal{C}$ does not have to be the same as the training corpus, and can be replaced or scaled at test time without re-training the encoder.

In practice, there is a significant number of phrases in the corpus, and it is expensive to index all of them. We therefore use a technique from Lee et al. (2021) that represents a phrase with *token* representations of the start and the end of the phrase. In this approach, we index representations of each distinct token in $\mathcal{C}$, and then at test time, use a k nearest neighbor search for the start and the end of the phrase, separately. Consider Figure 2 as an example. We represent a query with two vectors, $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$. We then use each to retrieve the start and the end of the plausible phrases—in this case, $\mathbf{c}_1$ and $\mathbf{c}_4$, which are the start and the end of ***Thessaloniki***, respectively.

Method. Formally, let $\mathcal{C} = \{c_1, \cdots, c_N\}$ be a reference corpus with N tokens. We first map each token c_i into a contextualized, h-dimensional vector $\mathbf{c}_i \in \mathbb{R}^h$ by feeding the text into the encoder and take the vector that corresponds to each token: $\mathbf{c}_1 ... \mathbf{c}_N = \text{Encoder}(c_1 ... c_N)$.

At inference time, NPM is given a query whose t-th token is masked: $q_1 ... q_{t-1}$, `[MASK]`, $q_{t+1} ... q_L$. We replace `[MASK]` with two special tokens $\texttt{[MASK}_\text{s}\texttt{]}\,\texttt{[MASK}_\text{e}\texttt{]}$ and feed it into the encoder to obtain a list of h-dimensional vectors:

$$\mathbf{q}_1 ... \mathbf{q}_{L+1} = \text{Encoder}(q_1 ... q_{t-1}, \texttt{[MASK}_\text{s}\texttt{]}, \texttt{[MASK}_\text{e}\texttt{]}, q_{t+1} ... q_L).$$

We then take the vector corresponding to $\texttt{[MASK}_\text{s}\texttt{]}$ and $\texttt{[MASK}_\text{e}\texttt{]}$ as $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, respectively.[1]

$$\mathbf{q}^{\text{start}} = \mathbf{q}_t, \mathbf{q}^{\text{end}} = \mathbf{q}_{t+1}.$$

We then make a prediction via:

$$\underset{v^* \in \mathcal{V}^*}{\text{argmax}} \sum_{i \le j} \mathbb{I}[v^* = c_{i:j}] \Big(\exp(\text{sim}(\mathbf{q}^{\text{start}}, \mathbf{c}_i)) + \exp(\text{sim}(\mathbf{q}^{\text{end}}, \mathbf{c}_j)) \Big),$$

where $\mathcal{V}^*$ is a set of possible n-grams defined by the vocabulary $\mathcal{V}$ and sim is a pre-defined similarity function that maps a pair of vectors into a scalar

[1] This allows obtaining two vectors without encoding the query twice, e.g., unlike Lee et al. (2021)

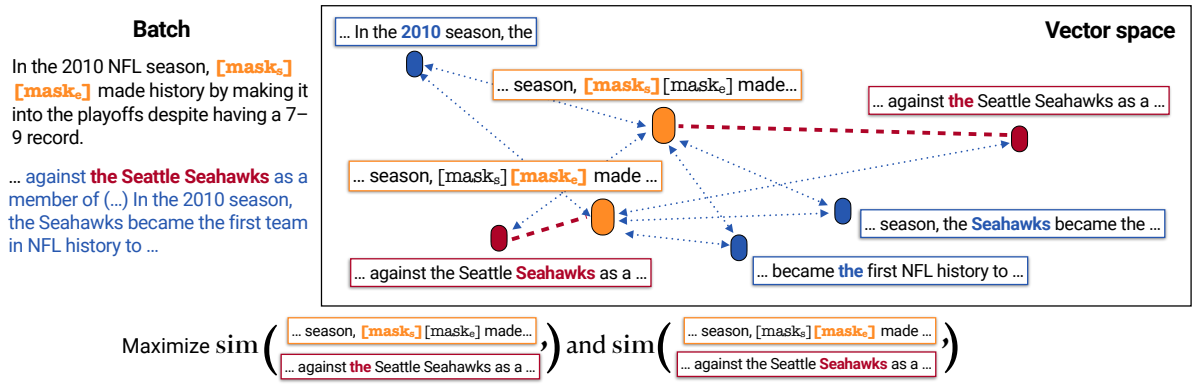

Figure 3: Training of NPM. (Section 3.2). [MASK$_s$] [MASK$_e$] indicates the masked *span* whose original phrase is *the Seattle Seahawks*. We maximize the similarity scores between ... [MASK$_s$] [MASK$_e$] ... and ...the Seattle Seahawks... , and between ... [MASK$_s$] [MASK$_e$] ... and ...the Seattle Seahawks... .

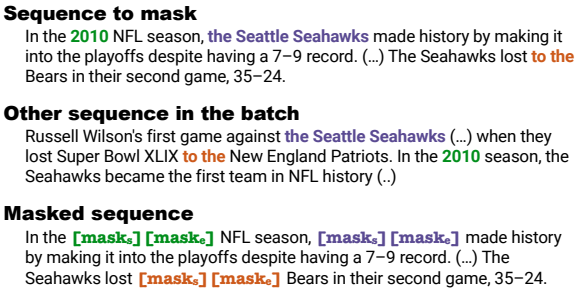

Figure 4: Our span masking (Section 3.2.1). For simplicity, this figure assumes two sequences in the batch. Spans to mask out are chosen based on whether there is any co-occurring spans in other sequences in the batch. Then, each span is replaced with [MASK$_s$] [MASK$_e$].

value. In practice, iterating over N tokens is infeasible. We thus use an approximation using a fast nearest neighbor search for the start and the end separately. Details are provided in Appendix A.1.

Similarity function. The choice of similarity function can be flexible. We follow Zhong et al. (2022b) in using a scaled inner product $\text{sim}(\mathbf{h}_1, \mathbf{h}_2) = \frac{\mathbf{h}_1 \cdot \mathbf{h}_2}{\sqrt{h}}$, where h is a dimension of the token vectors.

3.2 NPM: Training

NPM is trained on unlabeled text data. We describe the masking strategy first (Section 3.2.1), and then the training objective (Section 3.2.2).

3.2.1 Masking

We extend span masking (Joshi et al., 2020), which masks spans (consecutive tokens) whose length is sampled from a geometric distribution. Our span masking differs from Joshi et al. (2020) in two ways. First, we mask spans if they co-occur in the other sequences in the batch to guarantee in-batch positives during training (Section 3.2.2). For instance, masked spans in Figure 4 are '*2010*', '*the Seattle Seahawks*' and '*to the*' all of which are found in the other sequences. Second, instead of replacing each token in the span with a [MASK], we replace the whole span with two special tokens [MASK$_s$] [MASK$_e$]. For instance, each of '*2010*', '*the Seattle Seahawks*' and '*to the*' is replaced with [MASK$_s$] [MASK$_e$]. This is to obtain the start and the end vectors for each span as we do at inference.

3.2.2 Training Objective

Key idea. We illustrate an example in Figure 3. The masked span is '*the Seattle Seahawks*', thus the model should retrieve a phrase '*the Seattle Seahawks*' from other sequences in the reference corpus when it is given a query like this at test time. Specifically, we should encourage the [MASK$_s$] vector to be closer to ...the Seattle Seahawks... and the [MASK$_e$] vector to be closer to ...the Seattle Seahawks... , while being distant from other tokens. We train the model to do so by approximating the full corpus as the other sequences in the batch. Concretely, we train the model to retrieve the start and the end of the span '*the Seattle Seahawks*' from other sequences in the same batch. Note that our masking strategy ensures that every masked span has a co-occurring span in the batch (Section 3.2.1).

Obtaining vector representations. Consider the i-th sequence in the batch that consists of L tokens, $x^i = x^i_1 ... x^i_L$. We denote $\hat{x}^i = \hat{x}^i_1 ... \hat{x}^i_L$ as a consequence of span masking over x^i. Both x^i and $\hat{x}^i$ are fed into the encoder, and each token is mapped

into an h-dimensional vector:[2]

$$\begin{aligned}\mathbf{x}_1^i \cdots \mathbf{x}_L^i &= \text{Encoder}(x_1^i \cdots x_L^i),\\ \hat{\mathbf{x}}_1^i \cdots \hat{\mathbf{x}}_L^i &= \text{Encoder}(\hat{x}_1^i \cdots \hat{x}_L^i).\end{aligned}$$

Training objective. We consider a masked span in x_i, represented with $\texttt{[MASK}_\text{s}\texttt{][MASK}_\text{e}\texttt{]}$, denoted as $\hat{x}_t^i, \hat{x}_{t+1}^i$. We then denote g_t^i as the original n-gram that were replaced by $\hat{x}_t^i, \hat{x}_{t+1}^i$.

We now define the objective for this masked span, and the final objective is summed over all masked spans. The training objective for this masked span is defined as

$$-\left(\log\frac{\sum_{\mathbf{y}\in\mathcal{Y}_\text{s}^+(g_t^i)}\exp(\text{sim}(\hat{\mathbf{x}}_t^i,\mathbf{y}))}{\sum_{\mathbf{y}\in\mathcal{Y}_\text{s}^+(g_t^i)\cup\mathcal{Y}_\text{s}^-(g_t^i)}\exp(\text{sim}(\hat{\mathbf{x}}_t^i,\mathbf{y}))} + \log\frac{\sum_{\mathbf{y}\in\mathcal{Y}_\text{e}^+(g_t^i)}\exp(\text{sim}(\hat{\mathbf{x}}_{t+1}^i,\mathbf{y}))}{\sum_{\mathbf{y}\in\mathcal{Y}_\text{e}^+(g_t^i)\cup\mathcal{Y}_\text{e}^-(g_t^i)}\exp(\text{sim}(\hat{\mathbf{x}}_{t+1}^i,\mathbf{y}))}\right).$$

Here, $\text{sim}(\cdot,\cdot)$ is a similarity function defined in Section 3.1, and $\mathcal{Y}_\text{s}^+(g_t^i)$, $\mathcal{Y}_\text{s}^-(g_t^i)$, $\mathcal{Y}_\text{e}^+(g_t^i)$ and $\mathcal{Y}_\text{e}^-(g_t^i)$ are *start positives*, *start negatives*, *end positives* and *end negatives* of g_t^i, respectively, which are defined in the next paragraph. This objective follows a phrase-level contrastive learning objectives in prior work (Lee et al., 2021; Ram et al., 2021; Deng et al., 2021; Kulkarni et al., 2022) with an extension that allows *multiple* positives.

In-batch positives and negatives. The start positives and the end positives are the start and the end of the spans to be retrieved. The start negatives and the end negatives are tokens that are not the start positives and not the end positives, respectively. More formally:

$$\begin{aligned}\mathcal{Y}_\text{s}^+(g_t^i) &= \{x_m^j | g_t^i = x_m^j \ldots x_{m+|g_t^i|-1}^j \ \ \& \ \ i \neq j\},\\ \mathcal{Y}_\text{s}^-(g_t^i) &= \{x_m^j | g_t^i \neq x_m^j \ldots x_{m+|g_t^i|-1}^j \ \ \& \ \ i \neq j\},\\ \mathcal{Y}_\text{e}^+(g_t^i) &= \{x_m^j | g_t^i = x_{m-|g_t^i|+1}^j \ldots x_m^j \ \ \& \ \ i \neq j\},\\ \mathcal{Y}_\text{e}^-(g_t^i) &= \{x_m^j | g_t^i \neq x_{m-|g_t^i|+1}^j \ldots x_m^j \ \ \& \ \ i \neq j\}.\end{aligned}$$

Here, $|g_t^i|$ indicates the length of the span g_t^i.

4 Training Details

Training data. We use English Wikipedia (August 2019) and an English portion of CC-News (Mackenzie et al. (2020), February 2019) for training, which contains 13B tokens in total. The data is segmented into sequences, each with up to 256 tokens.

[2]The unmasked sequence and the masked sequence may have different lengths before padding, but we pad them to have the same length.

Training. We use the model architecture and initial weights of RoBERTa large (Liu et al., 2019), consisting of 354M parameters. Training is done for 100,000 steps, using thirty-two 32GB GPUs. One batch consists of 512 sequences (131,072 tokens). We use an Adam optimizer (Kingma and Ba, 2014) with a learning rate of 3×10^{-5}, weight decay of 0.01 and $4,000$ steps of warm-up.

Batching. The choice of batching is important in in-batch approximations, as it determines the quality of positives and negatives. For instance, Zhong et al. (2022b) uses BM25 to ensure the sequences in the same batch are likely to share the same topic. With a pretraining corpus with billions of tokens, it can be significantly expensive to build a BM25 index. Therefore, we instead construct the batch by grouping sequences from the same document and assigning them to the same batch.[3] This trick ensures that (a) positives (spans that share the string) are likely to share the context, reducing false positives, and (b) negatives are those that the model is likely to be confused with, thus training against them helps the model better identify positives. During training, we gather all sequences from multiple GPUs to increase the size of the effective batch and make in-batch approximation more effective.

5 Experiments: Closed-set Tasks

We perform zero-shot evaluation on closed-set tasks where a small set of candidates is given.

5.1 Evaluation Datasets

We include nine classification datasets that are known for not necessarily requiring factual knowledge: AGNews (Zhang et al., 2015), Yahoo (Zhang et al., 2015), Subj (Pang and Lee, 2004), SST-2 (Socher et al., 2013), MR (Pang and Lee, 2004), Rotten Tomatoes (RT), CR (Hu and Liu, 2004), Amazon polarity (Amz, McAuley and Leskovec (2013)) and RTE (Dagan et al., 2005). The tasks range from topic classification and sentiment analysis to subjectivity classification and textual entailment. Statistics are provided in Appendix B.

5.2 Baselines

We compare with the encoder-only, the decoder-only and the encoder-decoder models with various sizes (354M to 175B parameters). We include RoBERTa (Liu et al., 2019) as the encoder-only,

[3]Documents that are not long enough to construct a batch are grouped with each other.

Model	# Params	AGN	Yahoo	Subj	SST-2	MR	RT	CR	Amz	RTE	Avg
Baselines (encoder-only)											
RoBERTa (Gao et al., 2021)	1.0x	-	-	51.4	83.6	80.8	-	79.5	-	51.3	-
RoBERTa	1.0x	71.3	41.4	67.6	84.5	81.7	81.1	80.4	83.5	57.4	72.1
Baselines (encoder-decoder)											
T5	2.2x	72.0	51.3	54.9	57.5	57.7	59.1	56.4	59.3	55.6	58.2
T5 3B	8.5x	**80.5**	53.6	54.8	59.6	58.6	57.3	53.7	57.0	58.5	59.3
Baselines (decoder-only)											
GPT-2 (Shi et al., 2022)	2.2x	67.4	49.7	60.8	55.3	54.6	53.0	66.2	57.6	53.1	57.5
+ PMI (Shi et al., 2022)	2.2x	65.1	48.8	62.5	76.5	74.6	74.1	82.8	76.2	54.2	68.3
GPT-2 kNN† (Shi et al., 2022)	2.2x	29.8	37.0	50.0	47.1	49.9	49.1	69.3	57.4	54.1	49.3
GPT-2 kNN-LM† (Shi et al., 2022)	2.2x	78.8	51.0	62.5	84.2	78.2	80.6	84.3	85.7	55.6	73.4
GPT-3 (Holtzman et al., 2021)	500x	75.4	53.1	66.4	63.6	57.4	57.0	53.8	59.4	56.0	60.2
+ PMI (Holtzman et al., 2021)	500x	74.7	54.7	64.0	71.4	76.3	75.5	70.0	75.0	**64.3**	69.5
Ours (encoder-only, nonparametric)											
NPM†	1.0x	74.5	53.9	**75.5**	**87.2**	**83.7**	**86.0**	81.2	83.4	61.7	**76.4**
Full fine-tuning (reference)											
RoBERTa (Gao et al., 2021)	1.0x	-	-	97.0	95.0	90.8	-	89.4	-	80.9	-

Table 1: Zero-shot results on closed-set tasks. *# Params* indicates the relative number of model parameters compared to RoBERTa large (354M). RoBERTa, T5 and GPT-2 are their *large* variants unless specified otherwise; GPT-3 is from *Davinci, non-instruct*. Numbers with citations are taken from the corresponding papers. As a reference, we provide results of fine-tuning on the full training dataset in the last row. † indicates a reference corpus is used. **NPM significantly outperforms larger parameters models.**

T5 (Raffel et al., 2020) as the encoder-decoder, and GPT-2/3 (Radford et al., 2019; Brown et al., 2020) as the decoder-only model. For the decoder-only models, we additionally apply PMI (Holtzman et al., 2021) for better calibration of the model output. We also compare with Shi et al. (2022) who use kNN inference using GPT-2 with PMI. In particular, (1) GPT-2 kNN uses kNN inference without training, and (2) GPT-2 kNN-LM interpolates distributions from GPT-2 and GPT-2 kNN.

5.3 Setup

We use the templates and verbalizers from Shi et al. (2022) for all models. When available, we use fuzzy verbalizers from Shi et al. (2022). We use a domain-specific reference corpus: a union of the English Wikipedia and CC News for AGN, Yahoo and RTE, a subjectivity corpus for Subj, and a review corpus for sentiment classification datasets. Their sizes vary from 15M tokens to 126M tokens. Details are in Appendix B. Fast similarity search is done using FAISS (Johnson et al., 2019) with the HNSW index. We use $k = 4096$ for inference.

5.4 Results

NPM outperforms baselines in the zero-shot setting (Table 1). We discuss the results in detail below.

Comparison between baselines. Among parametric models, RoBERTa achieves the best performance, outperforming larger models including GPT-3. This is perhaps surprising, and is likely because bidirectionality of the encoder-only model plays a vital role, as claimed in Patel et al. (2022). The kNN-LM approach from Shi et al. (2022), which incorporates the nonparametric component to the parametric model, outperforms all other baselines. Nonetheless, solely relying on retrieval (kNN) performs poorly with GPT-2, suggesting that using kNN at inference only is limited.

Figure 5: Predictions from RoBERTa (baseline) and NPM. The bottom indicates the context NPM retrieves to fill in `[MASK]`. Note that the fuzzy verbalizer maps *broken* to `Negative` and *awesome* to `Positive`.

Baselines versus NPM. NPM significantly outperforms all baselines, achieving consistently competitive performance over all datasets. This indicates that, even for tasks that do not explicitly require external knowledge, nonparametric models are very competitive.

Qualitative analysis. Figure 5 depicts predictions from RoBERTa and NPM on a sentiment

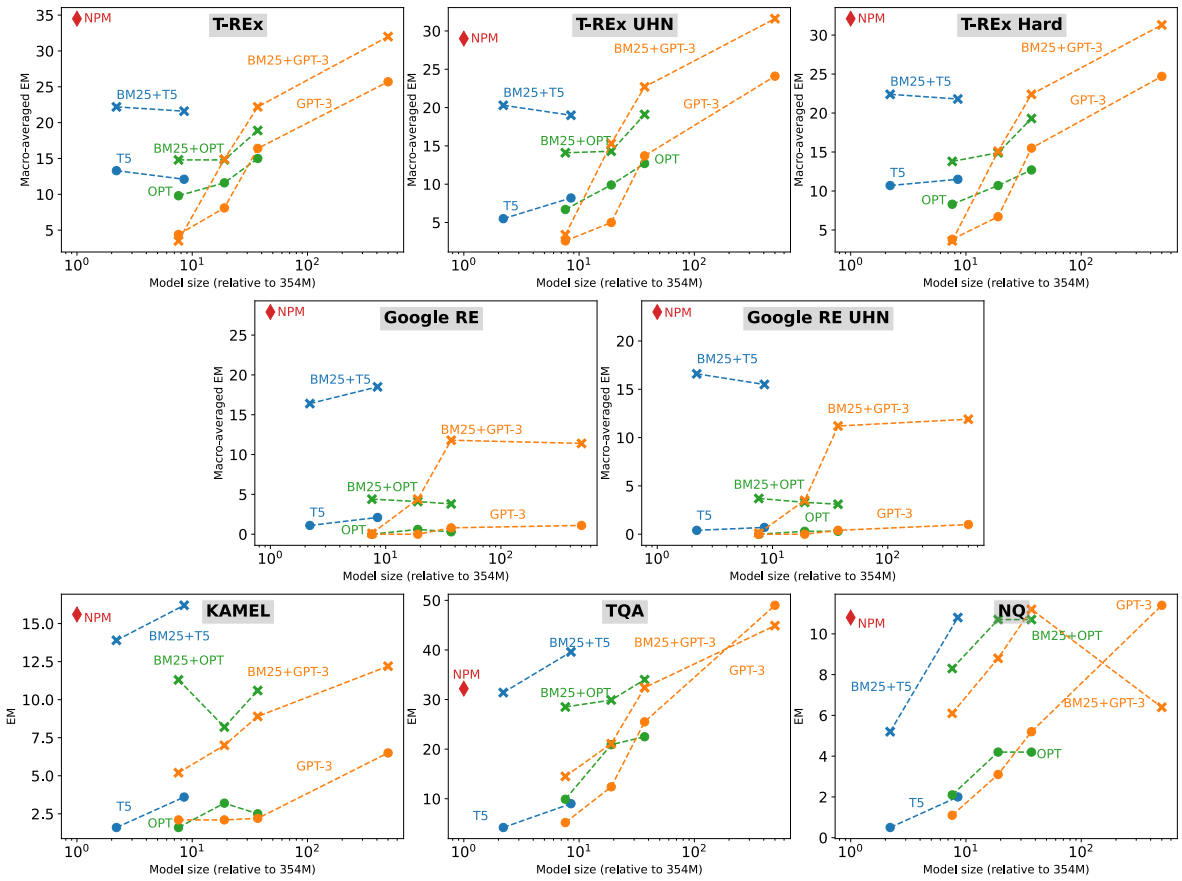

Figure 6: **Zero-shot results on knowledge tasks.** The x-axis indicates the relative number of model parameters in log scale compared to RoBERTa large (354M). NPM outperforms significantly larger parameters models, either with or without BM25. See Table 8 in Appendix C for the raw numbers.

analysis task. The first example uses *cheap* to indicate *inexpensive*, and the second example uses *cheap* to indicate *of very poor quality*. RoBERTa predicts `Positive` to both, while NPM makes correct predictions by retrieving the context that uses *cheap* in the same context as the input.

We also find that representations from NPM lead to better word sense disambiguation. For instance, RoBERTa assigns a high similarity score between *cheap* (*inexpensive*) and *cheap* (*of very poor quality*). On the other hand, NPM successfully assigns a low similarity score between *cheap* and *cheap*, even though their surface forms are the same.

6 Experiments: Open-set Tasks

We include zero-shot evaluation on open-set tasks whose answer can be any arbitrary-length string.

6.1 Evaluation Datasets

We evaluate on seven datasets: T-REx and Google-RE from LAMA (Petroni et al., 2019), KAMEL (Kalo and Fichtel, 2022), Natural Questions (NQ, Kwiatkowski et al. (2019)), TriviaQA (TQA, Joshi et al. (2017)), TempLAMA$^{22}_{19}$ and an entity translation task. In particular, TempLAMA requires probing knowledge with temporal updates, motivated by Dhingra et al. (2022) and Jang et al. (2022). The entity translation task involves a translation of an entity from English to other, non-Latin languages, requiring the model to predict extremely rare (if not unseen) characters. See Appendix B for details and statistics of all datasets.

6.2 Baselines

We compare with T5 (Raffel et al., 2020) as the encoder-decoder, and GPT-3 (Brown et al., 2020) and OPT (Zhang et al., 2022) as the decoder-only models. The encoder-only models are not applicable for open-set tasks since the number of tokens to predict is unknown.

Prior work found that a "retrieve-and-generate" approach that concatenates the input and passages from an off-the-shelf retrieval system is often helpful in knowledge-dependent tasks (Kandpal et al., 2022). We add them as baselines, using up to five passages from BM25 (Robertson et al., 2009).

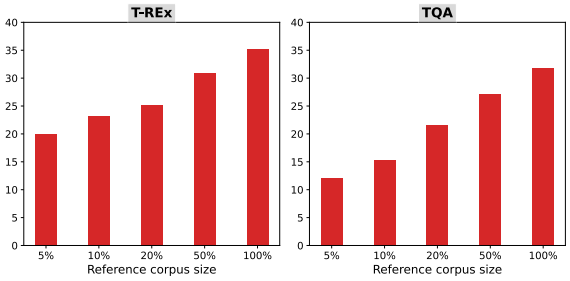

Figure 7: Ablation on the size of the reference corpus, from 41M tokens (5%) to 810M tokens (100%). There is a strong correlation between the size of the corpus and downstream performance.

6.3 Setup

For all datasets, we report Exact Match (EM). The LAMA test data is biased toward frequent entities because they are filtered to only include answers that are single tokens based on BERT (Devlin et al., 2019). Since we do not want our evaluation to be biased toward overly frequent entities, we report a micro-averaged accuracy over the data whose answers are 1, 2, 3 and 4+ grams, respectively. Other datasets do not have such filtering, therefore we report average EM.

As a reference corpus, we use the English Wikipedia from 08/01/2019, consisting of 810M tokens. For TempLAMA$^{22}_{19}$, we use the English Wikipedia from 08/01/2022, consisting of 858M tokens.

For NPM, we find combining with sparse retrieval significantly helps, likely because dense retrieval and sparse retrieval capture complementary features (Karpukhin et al., 2020; Seo et al., 2019). In particular, we reduce the search space to the top 3 passages based on BM25 and perform dense search as done in Kassner and Schütze (2020).

6.4 Results

Figure 6 show results on five knowledge tasks.

First, performance of parametric models largely depends on the number of parameters, as it has been claimed in much of prior work (Brown et al., 2020; Kandpal et al., 2022). The retrieve-and-generate approach that combines parametric models with BM25 significantly improves performance.

NPM outperforms or is on par with significantly larger baselines across all datasets. It substantially outperforms all models on two LAMA datasets, including 500x larger GPT-3 either with or without BM25. On KML, TQA and NQ, NPM consistently outperforms 37x larger models with or without BM25. This is impressive given that NPM is not trained on data with questions.

Model	#Params	Unchanged	Changed	AVG
Baselines				
T5	2.2x	1.9	0.4	1.1
T5 3B	8.5x	1.8	0.4	1.1
OPT 6.7B	19x	2.5	1.0	1.7
OPT 13B	37x	4.9	2.1	3.5
BM25 + T5	2.2x	13.7→14.9	3.0→**20.1**	17.5
BM25 + T5 3B	8.5x	11.9→12.0	2.2→17.8	14.9
BM25 + OPT 6.7B	19x	10.2→8.2	1.7→11.3	9.7
BM25 + OPT 13B	37x	14.8→14.4	2.8→16.6	15.5
Ours				
NPM	1.0x	18.9→**19.5**	2.9→17.5	**18.5**

Table 2: Results on TempLAMA$^{22}_{19}$, on an unchanged set, a changed set, and a macro-average over two, respectively. xx→xx indicates performance when using the outdated and the updated Wikipedia, respectively.

It is also worth noting that sparse retrieval is critical in NPM, e.g., without sparse retrieval, performance on LAMA-TREx drops from 34.5 to 16.1. We think this is because (1) sparse retrieval and dense retrieval capture complementary features, and (2) the removal of approximation in search improves search quality. We think future work can explore completely removing sparse retrieval, as has been done in Lee et al. (2021) to improve Seo et al. (2019).

Impact of the reference corpus size. Figure 7 reports the impact of the size of the reference corpus, from 41M tokens (5%) to 810M tokens (100%). Performance of NPM is highly correlated with the size of the reference corpus, strongly suggesting that using a larger reference corpus is important.

Results on temporal knowledge tasks. Table 2 reports results on TempLAMA. NPM retains its performance on the unchanged set (18.9 →19.5) and successfully updates its answers on the changed set (2.9 → 17.5). Its performance is significantly better than the performance of parametric models with up to 13B parameters, and is on par with a larger model with the retrieve-and-generate approach, which also successfully updates its answer by leveraging the updated corpus. This is in agreement with prior work that shows the model with a nonparametric component adapts to temporal updates by replacing the reference corpus at test time (Izacard et al., 2022). Nonetheless, the retrieve-and-generate approach is still significantly worse than NPM when the target entities are rare, which we show in the next paragraph.

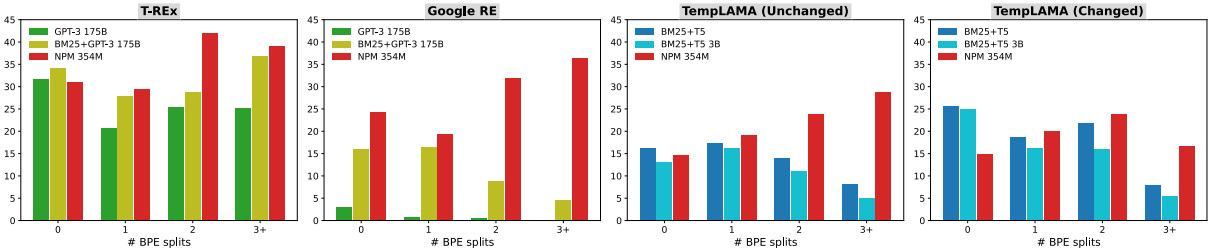

Figure 8: Performance on LAMA and TempLAMA tasks, broken down based on the number of BPE splits of the target entity, which is an indication of rarity of the entities (L:Frequent→R:Rare). NPM outperforms GPT-3 or T5 more significantly when the target entities are rare.

Model	#Params	#L	w/o BM25	w/ BM25
Baselines, English-only				
T5	2.2x		0.2	1.9
T5 3B	8.5x		0.5	4.4
OPT 6.7B	19x		0.4	22.3
OPT 13B	37x		1.0	24.6
Ours, English-only				
NPM	1.0x		**52.4**	
References, Multilingual				
mT5	3.4x	101	1.3	19.0
mT5 XL	11x	101	4.1	**56.6**
BLOOM 3B	8.5x	46	0.0	17.4
BLOOM 7.1B	20x	46	0.1	26.0

Table 3: Results on the entity translation task. See Table 10 in Appendix C for per-language results. *#L* indicates the number of languages multilingual models are trained on. **Bold** and **Bold** indicate the best among monolingual models and the best including multilingual models, respectively. NPM significantly outperforms all existing monolingual models, and approaches or outperforms larger multilingual models.

Performance on rare entities. We break down the instances on LAMA and TempLAMA based on the number of BPE splits of the target entity, e.g., *Thessaloniki* is one word that is split into 4 BPE tokens, thus the number of splits is 3. Since BPE splits a word if they are rare, the number of BPE splits indicates the rarity of the entity. We compare NPM with GPT-3 and BM25+GPT-3 on LAMA, and BM25+T5 (770M and 3B) on TempLAMA, the two most competitive baselines on each dataset.

Figure 8 reports results. On LAMA, NPM outperforms GPT-3 fairly consistently, with larger gains as the number of BPE splits increases. On TempLAMA, while BM25+T5 is competitive on frequent entities with zero BPE split, it consistently lags behind NPM with ≥ 1 BPE splits. This suggests that NPM is particularly good at addressing rare entities, compared to not only parametric models without retrieval but also the retrieve-and-generate approach.

Results in Entity Translation. Results on the entity translation task are shown in Table 3 (per-language results are reported in Table 10 of Appendix C). T5 and OPT struggle to perform the task, both with and without BM25 retrieval. In contrast, NPM performs well across all languages.

In order to better calibrate performance of NPM, we provide reference performance of models that are purposely trained on the multilingual data—mT5 (Xue et al., 2021) and BLOOM (Scao et al., 2022). NPM outperforms 3.4x larger mT5 and 20x larger BLOOM, and approaches 11x larger mT5, even though it is trained on English. We think strong cross-lingual transferability of NPM is likely because it can retrieve a phrase based on its surrounding context, even if it has not seen the exact word during training.

7 Conclusion

We introduced NPM, a nonparametric masked language model that replaces a softmax over the output vocabulary with a nonparametric distribution over a reference corpus. NPM can be efficiently trained using a contrastive objective and an in-batch approximation to a full corpus. Zero-shot evaluation on 16 tasks shows that NPM outperforms significantly larger parametric models. NPM is particularly good at rare patterns (word senses or facts), scaling and updating at test time, and predicting extremely rare if not unseen characters.

Limitation

Scaling through the inference corpus. The size of the reference corpus is an additional dimension for model scale in nonparametric models. In this paper, we scale the corpus up to nearly 1B tokens, which is still smaller than the training data of very large language models (Brown et al., 2020; Rae et al., 2021). We think future work can scale it fur-

ther using tools such as Distributed FAISS (Johnson et al., 2019) or ScaNN (Guo et al., 2020).

Significant memory usage. Using NPM saves GPU compute and memory compared to using models with more parameters. However, NPM requires more RAM and disk memory due to embeddings of a reference corpus. For instance, the largest corpus in our experiments (full English Wikipedia) requires 70GB of RAM and 1.4TB of disk memory. Future work can build more efficient NPM as done in prior work in nearest neighbor search (Jegou et al., 2010; Norouzi et al., 2012; Ge et al., 2014; Izacard et al., 2020; Yamada et al., 2021).

Exploration of larger vocabulary. Large vocabulary is known to lead performance gains (Conneau et al., 2020) but is bounded in memory costs. Previous work explored more efficient softmax approximations (Morin and Bengio, 2005; Chen et al., 2016; Grave et al., 2017). Our nonparametric training offers an alternative by removing the softmax over the vocabulary. With the RoBERTa architecture, increasing the vocab size by 2x makes the baseline training 50% more memory expensive, but does not increase the memory in training NPM. However, this paper does not include more systematic evaluation on the effect of large vocabulary. Future work can explore training NPM with a significantly larger vocabulary to further boost performance.

Extension for generation. Our paper evaluates NPM only on prediction tasks. It is currently nontrivial to use NPM for generation, since it is the encoder-only model. Future work can explore autoregressive generation as done in Patel et al. (2022) or use NPM for editing (Schick et al., 2022; Gao et al., 2022).

Extension to few-shot learning and fine-tuning. Our paper focuses on zero-shot evaluation only. Future work can extend NPM to a few-shot learning setup. In fact, fine-tuning NPM is significantly easier than fine-tuning larger models such as T5, OPT and GPT-3 which we compare NPM with, and can be explored in future work.

Better cross-lingual transfer. Our work explored cross-lingual transfer in a limited setup where the model is trained on monolingual data. We think future work can train multilingual NPM, and explore more comprehensive cross-lingual evaluation. In fact, nonparametric training may alleviate the burden of collecting large-scale multilingual corpora since it makes the model less sensitive to the language coverage in the training data, and may lead to significantly better cross-lingual transfer, as we demonstrate in the entity translation task.

Model	#Params	FS	SP	Acc	#Q/sec
RoBERTa	1.0x			67.6	36.36
NPM‡	1.0x	✓		75.5	7.63
OPT 2.7B	7.6x			2.1	0.71
OPT 2.7B + BM25‡	7.6x		✓	8.3	0.28
OPT 6.7B	19x			4.2	0.18
OPT 6.7B + BM25‡	19x		✓	10.7	0.12
NPM‡	1.0x		✓	10.8	4.52

Table 4: Inference speed measured on Subj with $|\mathcal{C}| = 15$M (the first block) and NQ with $|\mathcal{C}| = 810$M (the second block). A single GPU used (Quadro GP100). ‡ indicates the corpus is used. 'FS' and 'SP' indicate that a FAISS index is used and a sparse index (+ exact inner product search in case of NPM) is used, respectively. NPM is slower than the same-sized parametric model, but is faster than larger models (either with or without retrieval) while outperforming or matching performance.

Limitation in speed. We find that search makes inference considerably slower than the counterpart without search. We think that (1) search can significantly be faster with better engineering (we use the default hyperparameters of the FAISS index with no tuning) or better index, and (2) the speed of NPM is still on par with the speed of significantly larger parametric models that NPM outperforms (see Table 4). Moreover, while not explored in this work, there has been work that improves inference speed (He et al., 2021; Alon et al., 2022) that can be applied to NPM. We leave improving inference speed to future work.

Acknowledgements

We thank Ari Holtzman, Eric Wallace, Iz Beltagy, Jinhyuk Lee, Jungsoo Park, Mark Johnson, Noah Smith, Ofir Press, Patrick Lewis, Xiang Deng, Xinxi Lyu, Zexuan Zhong, UW-NLP members and anonymous reviewers for discussion and comments on the paper. This research was supported by NSF IIS-2044660, ONR N00014-18-1-2826, ONR MURI N00014- 18-1-2670, an Allen Distinguished Award and gifts from AI2. SM is supported by a J.P. Morgan fellowship.

References

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic

language modeling with automaton-augmented retrieval. In *Proceedings of the International Conference of Machine Learning*.

Mikel Artetxe, Jingfei Du, Naman Goyal, Luke Zettlemoyer, and Ves Stoyanov. 2022. On the role of bidirectionality in language model pre-training. In *Proceedings of Empirical Methods in Natural Language Processing*.

Akari Asai, Jungo Kasai, Jonathan H. Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021. XOR QA: Cross-lingual open-retrieval question answering. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Bogdan Babych and Anthony F. Hartley. 2003. Improving machine translation quality with automatic named entity recognition. *Proceedings of the International EAMT workshop on MT and other Language Technology Tools, Improving MT through other Language Technology Tools Resources and Tools for Building MT*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the International Conference of Machine Learning*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of Advances in Neural Information Processing Systems*.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the Association for Computational Linguistics*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the Association for Computational Linguistics*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*.

Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. ReasonBERT: Pre-trained to reason with distant supervision. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the International Conference on Language Resources and Evaluation*.

William T Freeman, Thouis R Jones, and Egon C Pasztor. 2002. Example-based super-resolution. *IEEE Computer graphics and Applications*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2022. Attributed text generation via post-hoc research and revision. *arXiv preprint arXiv:2210.08726*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the Association for Computational Linguistics*.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. In *Proceedings of the International Conference of Machine Learning*.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the International Conference of Machine Learning*.

Ahmed Hassan, Haytham Fahmy, and Hany Hassan. 2007. Improving named entity translation by exploiting comparable and parallel corpora. In *Proceedings of the International Workshop on Acquisition and Management of Multilingual Lexicons*.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Myles Hollander, Douglas A Wolfe, and Eric Chicken. 2013. *Nonparametric statistical methods*. John Wiley & Sons.

Ari Holtzman, Peter West, Vered Schwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of Empirical Methods in Natural Language Processing*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Knowledge Discovery and Data Mining*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Sebastian Riedel, and Edouard Grave. 2020. A memory efficient baseline for open domain question answering. *arXiv preprint arXiv:2012.15156*.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*.

Jan-Christoph Kalo and Leandra Fichtel. 2022. Kamel: Knowledge analysis with multitoken entities in language models. In *Proceedings of the Conference on Automated Knowledge Base Construction*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large language models struggle to learn long-tail knowledge. *arXiv preprint arXiv:2211.08411*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

Nora Kassner and Hinrich Schütze. 2020. BERT-kNN: Adfding a kNN search component to pretrained language models for better QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the International Conference on Learning Representations*.

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

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning rich representation of keyphrases from text. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*.

Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. 2023. Copy is all you need. In *Proceedings of the International Conference on Learning Representations*.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the Association for Computational Linguistics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the Association for Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of Advances in Neural Information Processing Systems*.

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. *arXiv preprint arXiv:1907.11692*.

Joel Mackenzie, Rodger Benham, Matthias Petri, Johanne R Trippas, J Shane Culpepper, and Alistair Moffat. 2020. Cc-news-en: A large english news corpus. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the ACM conference on Recommender systems*.

Robert C Moore. 2003. Learning translations of named-entity phrases from parallel corpora. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International workshop on artificial intelligence and statistics*.

Mohammad Norouzi, Ali Punjani, and David J Fleet. 2012. Fast search in hamming space with multi-index hashing. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3108–3115. IEEE.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics*.

Nikolaos Pappas, Phoebe Mulcaire, and Noah A. Smith. 2020. Grounded compositional outputs for adaptive language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1252–1267, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems*.

Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. 2022. Bidirectional language models are also few-shot learners. *arXiv preprint arXiv:2209.14500*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of Empirical Methods in Natural Language Processing*.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *arXiv preprint arXiv:1911.03681*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

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. *Journal of Machine Learning Research*.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Proceedings of the Association for Computational Linguistics*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*.

Minjoon Seo, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of Empirical Methods in Natural Language Processing*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the Association for Computational Linguistics*.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Sidney Siegel. 1957. Nonparametric statistics. *The American Statistician*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Empirical Methods in Natural Language Processing*.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of the International Semantic Web Conference*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of Empirical Methods in Natural Language Processing: System Demonstrations*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of Empirical Methods in Natural Language Processing*.

Frank F. Xu, Junxian He, Graham Neubig, and Vincent Josua Hellendoorn. 2022. Capturing structural locality in non-parametric language models. In *Proceedings of the International Conference on Learning Representations*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient passage retrieval with hashing for open-domain question answering. In *Proceedings of the Association for Computational Linguistics*.

Jinghui Yan, Jiajun Zhang, JinAn Xu, and Chengqing Zong. 2018. The impact of named entity translation for neural machine translation. In *Proceedings of the China Workshop on Machine Translation*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. In *Proceedings of the International Conference on Learning Representations*.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Proceedings of the Association for Computational Linguistics*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of Advances in Neural Information Processing Systems*.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the International Conference of Machine Learning*.

Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. 2022a. Describing differences between text distributions with natural language. In *Proceedings of the International Conference of Machine Learning*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022b. Training language models with memory augmentation. In *Proceedings of Empirical Methods in Natural Language Processing*.

A Model Details

A.1 Details of NPM

Approximation at inference. Given $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, we take the top k tokens with the highest similarity scores with each of them, and compute scores over spans composed by these tokens. Let $c^*_{i:j}$ be a span in $\mathcal{C}$ from the i-th token to the j-th token, and $\text{E}(c) \in \mathbf{R}^h$ be a vector corresponding to a token $c \in \mathcal{C}$. We find the top k tokens for the start and the end:

$$c_{\text{s}_1}, c_{\text{s}_2}, \cdots, c_{\text{s}_k} = \underset{c \in \mathcal{C}}{\text{argTopk}}\, \text{sim}(\mathbf{q}^{\text{start}}, \text{E(c)}),$$
$$c_{\text{e}_1}, c_{\text{e}_2}, \cdots, c_{\text{e}_k} = \underset{c \in \mathcal{C}}{\text{argTopk}}\, \text{sim}(\mathbf{q}^{\text{end}}, \text{E(c)})$$

using a fast nearest neighbor search. We then define a set of candidate phrases $\tilde{\mathcal{C}}^*$ as:

$$\left(\bigcup_{i=1}^{k} \bigcup_{j=1}^{l_{\max}} c^*_{\text{s}_i:\text{s}_i+j-1} \right) \cup \left(\bigcup_{i=1}^{k} \bigcup_{j=1}^{l_{\max}} c^*_{\text{e}_i-j+1:\text{e}_i} \right),$$

and predict:

$$\underset{v^* \in \mathcal{V}^*}{\text{argmax}} \sum_{c^* \in \tilde{\mathcal{C}}^*} \mathbb{I}[v^* = c^*] \text{expsim}(\mathbf{q}, \text{E}(c^*)),$$

where $\text{E}(c^*) \in \mathbf{R}^{2h}$ is a vector corresponding to c^*, and $\mathcal{V}^*$ is a set of any possible n-grams defined by the vocabulary $\mathcal{V}$.

A.2 Training Details

All implementation was done with PyTorch (Paszke et al., 2019), PyTorch Lightning[4] and Huggingface Transformers (Wolf et al., 2020).

Masking. We use a masking ratio of 15% for all models, following the standard in prior work (Devlin et al., 2019; Liu et al., 2019; Joshi et al., 2020). We implement masking as follows: (1) we first identify all possible candidate spans (spans that positives are found from other sequences in the batch), (2) sample the length of spans to mask from a geometric distribution with a hyperparameter $p = 0.5$, and (3) mask the spans with respect to the sampled length until the masking budget has been spent. We do not mask more than 128 spans from one sequence, and do not mask the span if the same span has been masked for more than ten times within the batch in order to prevent repeatedly masking overly frequent spans.

[4] https://github.com/Lightning-AI/lightning

For `[MASK`$_\text{s}$`]` and `[MASK`$_\text{e}$`]`, we use the `[MASK]` vocab from the RoBERTa tokenizer. Note that it is not necessary to use different tokens for `[MASK`$_\text{s}$`]` and `[MASK`$_\text{e}$`]` since the Transformer can handle positional information.

A.3 A special case: NPM SINGLE

Along with NPM, we introduce **NPM SINGLE**, which outputs a nonparametric distribution over every single *token* in $\mathcal{C}$, instead of a *phrase*. To some extent, NPM is a strict generalization of NPM SINGLE, and NPM SINGLE still has a problem that existing encoder-only models have, e.g., can only fill in the `[MASK]` with a single token. We however think NPM SINGLE can be useful for some applications, e.g., for fine-tuning, as existing encoder-only models are used for.

Inference. Given a reference corpus $\mathcal{C} = \{c_1, \cdots, c_N\}$, we construct N number of h-dimensional vectors $\mathbf{c}_1, \cdots, \mathbf{c}_N \in \mathbb{R}^h$ by feeding the text into the encoder. At inference time, given a query whose t-th token is `[MASK]`, we feed it into the encoder:

$$\mathbf{q}_1 ... \mathbf{q}_L = \text{Encoder}(q_1 ... q_{t-1}, \texttt{[MASK]}, q_{t+1} ... q_L).$$

We take $\mathbf{q}_t$ as a vector that represents the `[MASK]` token in the query. Finally, the prediction is made by aggregating the similarity scores to the tokens in $\mathcal{C}$:

$$\underset{v \in \mathcal{V}}{\text{argmax}} \sum_{c \in \mathcal{C}} \mathbb{I}[c = v] \exp(\text{sim}(\mathbf{q}_t, \text{E}(c))),$$

where $\text{E}(c) \in \mathbf{R}^h$ is a vector corresponding to c, and $\mathcal{V}$ is the vocabulary set.

In practice, since computing scores over all tokens in $\mathcal{C}$ is infeasible, an approximation is made by computing scores for the top k nearest neighbors only, and treating other tokens to have a similarity score of $-\text{Inf}$. More precisely:

$$c^1, c^2, \cdots, c^k = \underset{c \in \mathcal{C}}{\text{argTopk}}\, \text{sim}(\mathbf{q}_t, \text{E}(c))$$

are obtained by using an index (e.g., FAISS (Johnson et al., 2019)), and the following is returned as a prediction:

$$\underset{v \in \mathcal{V}}{\text{argmax}} \sum_{1 \le i \le k} \mathbb{I}[c^i = v] \exp(\text{sim}(\mathbf{q}_t, \text{E}(c^i))).$$

Training. Let $x_1^i ... x_L^i$ be the i-th sequence in the batch, whose subset is replaced with `[MASK]` and converted to $\hat{x}_1^i ... \hat{x}_L^i$. Both the unmasked sequence and the masked sequence are fed into the encoder, and each token is mapped into an h-dimensional vector:

$$\begin{aligned} \mathbf{x}_1^i \cdots \mathbf{x}_L^i &= \text{Encoder}(x_1^i \cdots x_L^i), \\ \hat{\mathbf{x}}_1^i \cdots \hat{\mathbf{x}}_L^i &= \text{Encoder}(\hat{x}_1^i \cdots \hat{x}_L^i). \end{aligned}$$

The training objective is then defined as:

$$\sum_{t=1}^{L} \mathbb{I}[\hat{x}_t = \texttt{[MASK]}] l(x_t^i, \hat{x}_t^i),$$

where $l(x_t^i, \hat{x}_t^i)$ is

$$-\log \frac{\sum_{\mathbf{y} \in \mathcal{Y}^+(x_t^i)} \exp(\text{sim}(\hat{\mathbf{x}}_t^i, \mathbf{y}))}{\sum_{\mathbf{y} \in \mathcal{Y}^+(x_t^i) \cup \mathcal{Y}^-(x_t^i)} \exp(\text{sim}(\hat{\mathbf{x}}_t^i, \mathbf{y}))}.$$

Here, $\text{sim}(\cdot, \cdot)$ is a similarity function defined in Section 3.1, and $\mathcal{Y}^+(x_t^i)$ and $\mathcal{Y}^-(x_t^i)$ are *positives* and *negatives* of x_t^i—tokens from *other* sequences in the batch that share and do not the vocab, respectively.

$$\begin{aligned} \mathcal{Y}^+(x_t^i) &= \left\{ x_m^j | x_t^i = x_m^j \text{ and } i \neq j \right\}, \\ \mathcal{Y}^-(x_t^i) &= \left\{ x_m^j | x_t^i \neq x_m^j \text{ and } i \neq j \right\}. \end{aligned}$$

A.4 Inference on closed-set tasks

When applying NPM and NPM SINGLE on closed-setk tasks, we closely follow Shi et al. (2022) who adapts kNN-LM for zero-shot inference on classification tasks. We assume a fuzzy verbalizer: $f : \mathcal{Y} \rightarrow \tilde{\mathcal{V}}$, where $\mathcal{Y}$ is a set of labels in the task and $\tilde{\mathcal{V}} \in \mathcal{V}$ is a subset of the vocabulary $\mathcal{V}$. The fuzzy verbalizer maps a label to a set of tokens that express the label, e.g., in a sentiment classification task, $f(\texttt{Positive})$ includes *awesome* or *great*, and $f(\texttt{Negative})$ includes *terrible* or *broken*.

NPM SINGLE is given a query vector $\mathbf{q} \in \mathbb{R}^h$ and predicts:

$$\underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{c \in \mathcal{C}} \mathbb{I}[c \in f(y)] \exp\left(\frac{\text{sim}(\mathbf{q}, \text{E}(c))}{\tau}\right),$$

where $\text{E}(c) \in \mathbf{R}^h$ is a vector corresponding to c, and τ is a hyperparameter.

NPM is given a query vector $\mathbf{q} \in \mathbb{R}^{2h}$ and predicts:

$$\underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{c^* \in \mathcal{C}^*} \mathbb{I}[c^* \in f(y)] \exp\left(\frac{\text{sim}(\mathbf{q}, \text{E}(c^*))}{\tau}\right),$$

where $\text{E}(c^*) \in \mathbf{R}^{2h}$ is a vector corresponding to c^*. Note that this is essentially equivalent to

$$\underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{c \in \mathcal{C}} \mathbb{I}[c \in f(y)] \exp\Bigg(\frac{\text{sim}(\mathbf{q}^{\text{start}}, \text{E}(c))}{\tau} + \frac{\text{sim}(\mathbf{q}^{\text{end}}, \text{E}(c))}{\tau} \Bigg).$$

We use $\tau = 5.0$ for both NPM SINGLE and NPM.

B Evaluation Details

Table 5 reports statistics and templates on each downstream task, and Table 6 reports statistics of the retrieval corpus used in experiments.

For closed-set tasks, we use templates and verbalizers provided by Shi et al. (2022) for most datasets, except two datasets. For RTE, we use the template from Artetxe et al. (2022). For Subj, we write our own template, motivated by Zhong et al. (2022a) that found Subj is mainly about differentiating a review and a summary. For open-set tasks, we use templates provided by the original authors, except NQ and TQA for which we use the templates from GPT-3 (Brown et al., 2020). Due to limited computation resource, we subsample the data to include up to 3,000 examples, following the standard from prior work (Zhao et al., 2021; Shi et al., 2022). For closed-set tasks, we use exactly the same set of data as Shi et al. (2022), and for open-set tasks, we use the same script to subsample the data. For LAMA T-REx and Google RE, we subsample up to 1,000 examples for each of 1, 2, 3 and 4+ grams. For the entity translation task, we subsample up to 1,000 examples per language.

The following is a more detailed description of open-set tasks used in Section 6.

LAMA (Petroni et al., 2019) is a factual probing benchmark that is designed to quantify the amount of factual knowledge in the model. It requires the model to predict the object given a subject-relation tuple in a cloze format. We use two versions of LAMA (Petroni et al., 2019): (1) LAMA T-REx, derived from Elsahar et al. (2018) and (2) LAMA Google-RE, derived from the Google-RE corpus.[5] For each version, we additionally consider the UHN (UnHelpfulNames) subset (Poerner et al., 2019)) where instances whose subject strongly hints the object by names (e.g., `Apple Watch`

[5] https://code.google.com/archive/p/relation-extraction-corpus

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}_s\rvert$	# labels	Example
Closed-set tasks				
AGN	120,000	3,000	4	Indiana defends its NCAA mens's soccer title by edging UC Santa Barbara in penalty kicks. The text topic is about `[MASK]`. (`[MASK]`={politics, sports, business, technology})
Yahoo	60,000	3,000	10	Company for cemecal at espaniea? Answer: Can you give us more info? The text topic is about `[MASK]`. (`[MASK]`={society, science, health, education, computer, sports, business, entertainment, family, politics})
Subj	2,000	2,000	2	He tells mitchell that he is now in debt. This is a `[MASK]`. (`[MASK]`={review, summary})
SST-2	2,210	2,210	2	It was `[MASK]`. (`[MASK]`={great, terrible})
MR	2,000	2,000	2	Simplistic, silly and tedious. It was `[MASK]`. (`[MASK]`={great, terrible})
RT	1,066	1,066	2	weird. rewarding. It was `[MASK]`. (`[MASK]`={great, terrible})
CR	2,000	2,000	2	I am very pleased so far. It was `[MASK]`. (`[MASK]`={great, terrible})
Amz	400,000	3,000	2	It was `[MASK]`. (`[MASK]`={great, terrible})
RTE	277	277	2	Most commercial logwood is grown in Honduras, right? `[MASK]`, plants are grown in water or in substance other than soil. (`[MASK]`={Yes, No})
Open-set tasks				
LAMA T-REx	34,039	2,983	-	AVCDH is owned by `[MASK]`.
LAMA Google RE	5,200	1,856	-	Joshua Mathiot died in `[MASK]`.
KAMEL	46,800	3,000	-	What is followed by So-Lo? Answer: `[MASK]`.
NQ	3,610	3,000	-	who sang i ran all the way home? The answer is: `[MASK]`.
TQA	11,313	3,000	-	Who wrote the opera Carmen? The answer is: `[MASK]`.
TempLAMA$^{22}_{19}$				
- changed	3,360	3,000	-	Contributor Covenant is developed by `[MASK]`.
- unchanged	3,360	3,000	-	Atari 8-bit family is developed by `[MASK]`.
Entity translation	10,452	6,622	-	The Korean translation of Banpo Bridge is: `[MASK]`.

Table 5: Statistics of downstream datasets. $\lvert\mathcal{D}\rvert$ and $\lvert\mathcal{D}_s\rvert$ indicate the number of test examples on the original data and the subsampled data, respectively. See Appendix B for details.

Corpus name	Source	$\lvert\mathcal{C}\rvert$	Datasets used
En-Wiki+CCNews	Subset of En-Wiki 08/01/2019 and CCNews	126M	AGN, Yahoo, RTE
Subjectivity corpus	Raw IMDB	15M	Subj
Review corpus	Amazon and IMDB	62M	SST-2, MR, RT, CR, Amz
En-Wiki 2019	En-Wiki 08/01/2019	810M	All open-set tasks
En-Wiki 2022	En-Wiki 08/01/2022	858M	TempLAMA$^{22}_{19}$

Table 6: Statistics of the retrieval corpus. $\lvert\mathcal{C}\rvert$ indicates the number of tokens in the corpus.

and `Apple`) are excluded. We also consider the hard subset of T-Rex from Zhong et al. (2021).

Note that Petroni et al. (2019) only include triples whose object is one token based on BERT (Devlin et al., 2019); however, with a different pretrained model like RoBERTa, entities could be multiple BPE tokens. Entities that are splitted into multiple BPE tokens are more rare entities.

KAMEL (Kalo and Fichtel, 2022) is another factual probing task as LAMA but with a few key differences to make it more general and broad: (1) it includes a broader coverage of triples, (2) it removes the constraint that the object is one token based on BERT, (3) it includes objects with literal values, and (4) it has a question answering format.

Natural Questions (NQ, Kwiatkowski et al. (2019)) and **TriviaQA** (TQA, Joshi et al. (2017)) are two welll-studied open-domain question answering datasets. We use the open-version of NQ (Lee et al., 2019) and TQA where the question is the only input and the model should use its knowledge to answer the question.

TempLAMA$^{22}_{19}$ is a task that requires probing knowledge with temporal updates. The task is first introduced by Dhingra et al. (2022) and Jang et al. (2022); however, we could not use either of existing data as their time split do not match our training. We therefore create the data by using a script provided by Dhingra et al. (2022) but using the 2019 and the 2022 dumps. We take Wikipedia triples whose relations are available for a template from either Petroni et al. (2019) or Dhingra et al. (2022). We then include triples whose object entities differ between the 2019 dump and the 2022 dump (due to the entity being updated), or only appear in the 2022 dump (due to the subject or the relation being

ISO Code	Language	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}_s\rvert$
zh	Chinese	3,199	1,000
ar	Arabic	2,013	1,000
el	Greek	1,618	1,000
iw	Hebrew	841	841
ru	Russian	758	758
jp	Japanese	471	471
hi	Hindi	427	427
ko	Korean	418	418
pl	Polish	177	177
tr	Turkish	150	150
cs	Czech	109	109
ta	Tamil	80	80
th	Thai	74	74
mn	Mongolian	64	64
ml	Malayalam	53	53
TOTAL		10,452	6,622

Table 7: Statistics of the entity translation benchmark. Languages are sorted based on their availabilities.

added) to the *changed* set. Otherwise, triples are included in the *unchanged* set. We additionally find that many triples are overly difficult because the fact is extremely niche and not really known. We thus filter the data to only include facts that appear in Wikipedia. Specifically, we include triples if the subject has a corresponding Wikipedia page and the object entity appears in that Wikipedia page.

Entity translation requires translating an entity from English to other languages that are not Latin based. While this is mainly to evaluate if the model can generate rare or unseen characters that are not in English, the entity translation task itself is a vital and challenging task in real applications such as machine translation (Babych and Hartley, 2003; Yan et al., 2018) and cross-lingual question answering (Clark et al., 2020; Asai et al., 2021). It is often beyond a series of simple translations of each word, or spelling out its pronunciation (Moore, 2003; Hassan et al., 2007; Sun et al., 2017). For instance, the Korean translation of *Banpo Bridge* in Figure 1 (반포대교) is not the concatenation of the translations of *Banpo* and *Bridge* (반포 다리).

We first identify a list of 15 non-Latin languages: Arabic (ar), Czech (cs), Greek (el), Hindi (hi), Hebrew (iw), Japanese (jp), Korean (ko), Malayalam (ml), Mongolian (mn), Polish (pl), Russian (ru), Tamil (ta), Thai (th), Turkish (tr), and Chinese (zh). We then implement heuristics to identify entities and their translations from English Wikipedia. Specifically, we parse the first paragraph of each Wikipedia article and pair the found translation with a topic entity of the article. For instance, a Korean translation of `Banpo Bridge` is found from the first sentence of `https://en.wikipedia.org/wiki/Banpo_Bridge`. Per-language statistics are reported in Table 7.

C Additional Results

Full results on knowledge tasks. Table 8 reports full results on five knowledge tasks. See Figure 6 for an illustration, and Section 6.4 for discussion.

Comparison to few-shot GPT-3. Table 9 compares zero-shot NPM SINGLE and NPM with zero- and four-shot GPT-3. Our zero-shot models outperform 500x larger zero-shot GPT-3 and 7.6x larger 4-shot GPT-3, but lag behind 4-shot GPT-3 that is 19x or larger. We think future work can explore extending our models to a few-shot setup.

Additional qualitative results. Figure 9 depicts predictions from RoBERTa and NPM in topic classification, choosing a label between four candidates: *health*, *computer*, *travel* and *politics*. All three examples contain the word *torch*, but with different meanings, e.g., an infectious diseases, a tool, and a computer library. RoBERTa predicts *health* for all of them, while NPM predicts *health*, *travel* and *computer*, which are all correct predictions.

As in Figure 5, we find that representations from NPM enable better word sense disambiguation: the pairwise similarities between between different meanings of *torch* are significantly lower than the pairwise similarities between other tokens that share the meaning.

Entity translation given an oracle passage. We evaluate models on the entity translation task where an oracle passage—a passage that is guaranteed to contain the translation information—is provided to the model. Baselines prepend oracle passages to the input, as it does with the retrieve-and-generate approach. NPM uses oracle passages to restrict the search space.

Table 11 reports results. While performance overall increases compared to when the oracle passage is not provided, the overall comparison between models does not change from Table 10: (1) all monolingual models significantly suffer, except for a couple of languages that are derived from Latin; (2) NPM significantly outperforms all monolingual models; (3) NPM even outperforms 3.4x larger mT5 and 20x larger BLOOM, and approaches 11x larger mT5.

Model	#Params	$\mathcal{C}$	T-REx			Google RE		KML	TQA	NQ
			All	UHN	Hard	All	UHN			
Baselines (encoder-decoder)										
T5	2.2x		13.3	5.5	10.7	1.1	0.4	1.6	4.2	0.5
T5 3B	8.5x		12.1	8.2	11.5	2.1	0.7	3.6	9.0	2.0
BM25 + T5	2.2x	✓	22.2	20.3	22.4	16.4	16.6	13.9	31.4	5.2
BM25 + T5 3B	8.5x	✓	21.6	19.0	21.8	18.5	15.5	**16.2**	39.6	10.8
Baselines (decoder-only)										
OPT 2.7B	7.6x		9.8	6.7	8.3	0.0	0.0	1.6	9.9	2.1
GPT-3 2.7B	7.6x		4.4	2.6	3.8	0.0	0.0	2.1	5.2	1.1
OPT 6.7B	19x		11.6	9.9	10.7	0.6	0.3	3.2	20.9	4.2
GPT-3 6.7B	19x		8.1	5.0	6.7	0.0	0.0	2.1	12.4	3.1
OPT 13B	37x		15.0	12.7	12.7	0.3	0.3	2.5	22.5	4.2
GPT-3 13B	37x		16.4	13.7	15.5	0.8	0.4	2.2	25.5	5.2
GPT-3 175B	500x		25.7	24.1	24.7	1.1	1.0	6.5	**49.0**	**11.4**
BM25 + OPT 2.7B	7.6x	✓	14.8	14.1	13.8	4.4	3.7	11.3	28.5	8.3
BM25 + GPT-3 2.7B	7.6x	✓	3.5	3.4	3.6	0.1	0.1	5.2	14.5	6.1
BM25 + OPT 6.7B	19x	✓	14.8	14.3	14.9	4.1	3.3	8.2	29.9	10.7
BM25 + GPT-3 6.7B	19x	✓	14.9	15.3	15.1	4.4	3.5	7.0	21.1	8.8
BM25 + OPT 13B	37x	✓	18.9	19.1	19.3	3.8	3.1	10.6	34.0	10.7
BM25 + GPT-3 13B	37x	✓	22.2	22.7	22.4	11.8	11.2	8.9	32.4	11.2
BM25 + GPT-3 175B	500x	✓	32.0	**31.6**	31.3	11.4	11.9	12.2	44.9	6.4
Ours (encoder-only, nonparametric)										
NPM	1.0x	✓	**34.5**	29.0	**32.1**	**27.9**	**23.0**	15.6	32.2	10.8

Table 8: Results on open-set tasks (numbers used in Figure 6). *# Params* indicates the relative number of model parameters compared to RoBERTa large (354M), and $\mathcal{C}$ indicates whether a text corpus is used. For LAMA (T-REx and Google RE), the macro-averaged EM over 1, 2, 3 and 4+ grams are reported. All models are zero-shot. NPM significantly outperforms larger parameters models, either with and without a retrieval-and-generate approach that uses BM25.

Model	#Params	AGN		SST-2	
		0-shot	4-shot	0-shot	4-shot
Baselines (Parametric)					
RoBERTa	x1.0	71.3	-	84.5	-
GPT-3 2.7B (Zhao et al., 2021)	x7.6	44.7	43.3	57.2	59.1
+ CC (Zhao et al., 2021)	x7.6	63.2	71.1	71.4	79.9
GPT-3 2.7B (Holtzman et al., 2021)	x7.6	69.0	-	53.8	88.1
+ PMI (Holtzman et al., 2021)	x7.6	67.9	-	72.3	87.7
GPT-3 6.7B (Holtzman et al., 2021)	x19	64.2	-	54.5	92.9
+ PMI (Holtzman et al., 2021)	x19	57.4	-	80.0	79.8
GPT-3 13B (Holtzman et al., 2021)	x37	69.8	-	69.0	85.4
+ PMI (Holtzman et al., 2021)	x37	70.3	-	81.0	86.9
GPT-3 175B (Zhao et al., 2021)	x500	43.9	61.0	71.6	93.6
+ CC (Zhao et al., 2021)	x500	73.9	85.9	75.8	94.3
GPT-3 175B (Holtzman et al., 2021)	x500	75.4	-	63.6	89.9
+ PMI (Holtzman et al., 2021)	x500	74.7	-	71.4	95.5
Ours (Nonparametric)					
NPM SINGLE	x1.0	74.2	-	86.8	-
NPM	x1.0	74.5	-	87.2	-

Table 9: Comparison to GPT-3 on AG News and SST-2. *# Params* indicates the relative number of model parameters compared to RoBERTa large (354M). All GPT-3 numbers are taken from previous work. k-shot indicates that the model performs in-context learning with k labeled examples with no gradient updates. We report on SST-2 and AGN, because they are all datasets shared between our paper and previous papers that report GPT-3 results (Zhao et al., 2021; Holtzman et al., 2021). Our zero-shot models outperform 500x larger zero-shot GPT-3 and 7.6x larger 4-shot GPT-3, but lag behind 4-shot GPT-3 that is 19x or larger.

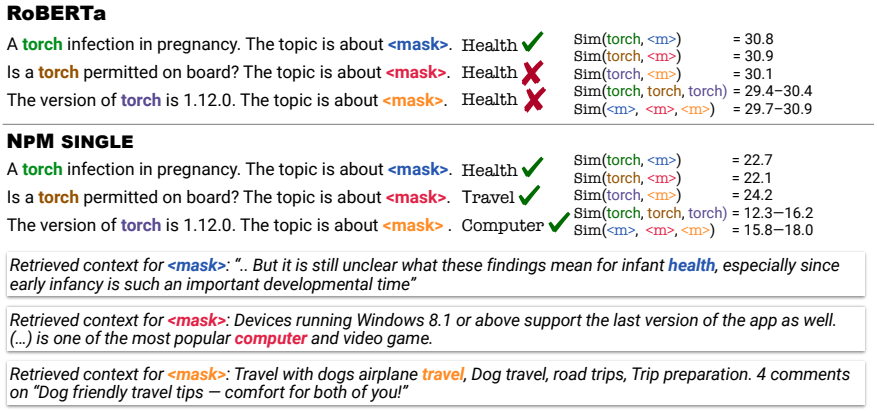

Figure 9: Predictions from RoBERTa (baseline) and NPM on a topic classification task (classes={*health, computer, travel, politics*}). The bottom indicates the context NPM retrieves to fill in `[MASK]`. On the right, we indicate the token-wise similarity scores. NPM assigns significantly lower scores to the token pairs with distinct meanings than to the token pairs with the similar meaning, e.g., *torch* (*a disease*) and *torch* (*a tool*).

Model	#Params	#L	ar	cs	el	hi	iw	jp	ko	ml	mn	pl	ru	ta	th	tr	zh	AVG
Baselines, English-only																		
T5	2.2x		0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	1.1	0.9	0.0	0.0	0.0	0.0	0.2
T5 3B	8.5x		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.6	1.3	0.0	0.0	0.0	0.0	0.5
OPT 6.7B	19x		0.0	0.0	0.3	0.0	0.0	0.0	3.1	0.0	0.0	0.0	2.9	0.0	0.0	0.0	0.0	0.4
OPT 13B	37x		1.5	0.0	1.2	0.7	0.0	0.0	1.4	0.0	0.0	1.1	7.4	0.0	0.0	1.3	0.1	1.0
BM25 + T5	2.2x		0.0	5.5	0.3	0.2	0.5	0.0	0.2	1.9	0.0	6.8	0.8	1.2	0.0	11.3	0.0	1.9
BM25 + T5 3B	8.5x		0.0	12.8	0.1	0.7	0.2	0.8	0.0	0.0	1.6	28.8	1.7	0.0	0.0	20.0	0.0	4.4
BM25 + OPT 6.7B	19x		26.4	**54.1**	15.5	11.2	11.8	14.4	19.6	5.7	3.1	**47.5**	52.5	6.2	12.2	32.0	22.7	22.3
BM25 + OPT 13B	37x		17.3	51.4	24.9	15.5	27.8	12.3	22.0	11.3	7.8	45.8	48.2	8.8	18.9	**34.0**	23.3	24.6
Ours, English-only																		
NPM	1.0x		**51.9**	33.0	**60.9**	**63.2**	**63.7**	**59.0**	**60.5**	**50.9**	**46.9**	33.3	**61.2**	**51.2**	**60.8**	32.7	**56.9**	**52.4**
References, Multilingual																		
mT5	3.4x	101	0.3	1.8	1.5	0.0	0.4	1.9	0.7	0.0	0.0	1.1	4.6	2.5	1.4	3.3	0.7	1.3
mT5 XL	11x	101	4.4	3.7	4.9	6.8	0.7	2.3	4.1	1.9	4.7	5.6	8.0	5.0	0.0	6.7	2.8	4.1
BLOOM 3B	8.5x	46	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0
BLOOM 7.1B	20x	46	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.1
BM25 + mT5	3.4x	101	12.4	22.9	21.6	9.8	12.5	28.9	19.1	11.3	18.8	15.8	16.0	17.5	28.4	16.7	33.4	19.0
BM25 + mT5 XL	11x	101	**64.4**	**64.2**	54.3	**65.6**	62.7	55.4	**69.4**	43.4	**62.5**	52.0	53.7	37.5	50.0	**48.7**	**65.0**	**56.6**
BM25 + BLOOM 3B	8.5x	46	24.2	25.7	1.7	13.3	15.1	18.5	17.9	5.7	6.2	21.5	11.1	10.0	27.0	18.0	44.5	17.4
BM25 + BLOOM 7.1B	20x	46	19.0	49.5	11.4	20.8	8.1	30.1	25.4	5.7	6.2	**54.2**	29.0	6.2	37.8	33.3	53.7	26.0

Table 10: Results on the entity translation task. *#L* indicates the number of languages multilingual models are trained on. **Bold** and **Bold** indicate the best among monolingual models and the best including multilingual models, respectively. NPM significantly outperforms all existing monolingual models, and approaches or outperforms larger multilingual models.

Model	#Params	#L	ar	cs	el	hi	iw	jp	ko	ml	mn	pl	ru	ta	th	tr	zh	AVG
Baselines, English-only																		
T5	2.2x		0.0	13.8	0.7	0.9	0.6	1.1	0.5	3.8	1.6	15.8	1.3	7.5	0.0	16.7	0.4	4.0
T5 3B	8.5x		0.2	21.1	1.0	0.7	0.7	2.3	1.2	3.8	4.7	37.3	2.9	8.8	1.4	30.7	0.4	7.3
OPT 6.7B	19x		24.4	56.9	22.9	15.5	19.7	19.1	32.5	24.5	3.1	**56.5**	60.9	22.5	23.0	46.0	30.2	30.5
OPT 13B	37x		20.7	**62.4**	22.7	15.7	30.9	17.6	36.1	18.9	15.6	**56.5**	52.2	22.5	35.1	**48.7**	40.0	33.0
Ours, English-only																		
NPM	1.0x		**70.3**	44.0	**76.8**	**74.0**	**82.4**	**71.3**	**73.2**	**58.5**	**59.4**	45.2	**71.5**	**68.8**	**66.2**	45.3	**74.5**	**65.4**
References, Multilingual																		
mT5	3.4x	101	19.4	25.7	30.8	19.0	20.6	33.8	28.2	28.3	40.6	18.6	23.1	30.0	29.7	26.7	37.4	27.5
mT5 XL	11x	101	**83.2**	**76.1**	69.6	**81.5**	77.4	68.2	**85.2**	49.1	**67.2**	**65.5**	62.7	51.2	**68.9**	**64.0**	**79.0**	**69.9**
BLOOM 3B	8.5x	46	51.2	27.5	3.1	30.2	34.1	34.0	30.9	11.3	7.8	28.2	23.0	17.5	37.8	22.0	70.1	28.6
BLOOM 7.1B	20x	46	29.6	43.1	12.0	27.6	12.2	32.5	30.9	9.4	15.6	59.3	38.1	13.8	43.2	32.0	65.5	31.0

Table 11: Results on the entity translation task given an oracle passage. *#L* indicates the number of languages multilingual models are trained on. **Bold** and **Bold** indicate the best excluding multilingual models and the best including multilingual models, respectively.