Retrieval-based LMs

Yufang Liu

East China Normal University

@ukhndlwl

Generalization through Memorization: Nearest Neighbor Language Models

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, Mike Lewis

Stanford University, Facebook AI Research

ICLR 2020

Language Models

Lots of text is very easily available, so we train models on large amounts of data.

But improving LM performance or scaling to larger datasets, by training bigger and bigger models with billions of parameters, requires massive amounts of GPU compute..☹

Language Models

Instead, can explicitly memorizing data make LMs generalize better without the added cost of training?

Language Models

Nearest Neighbor Language Models

kNN-LM: Intuition

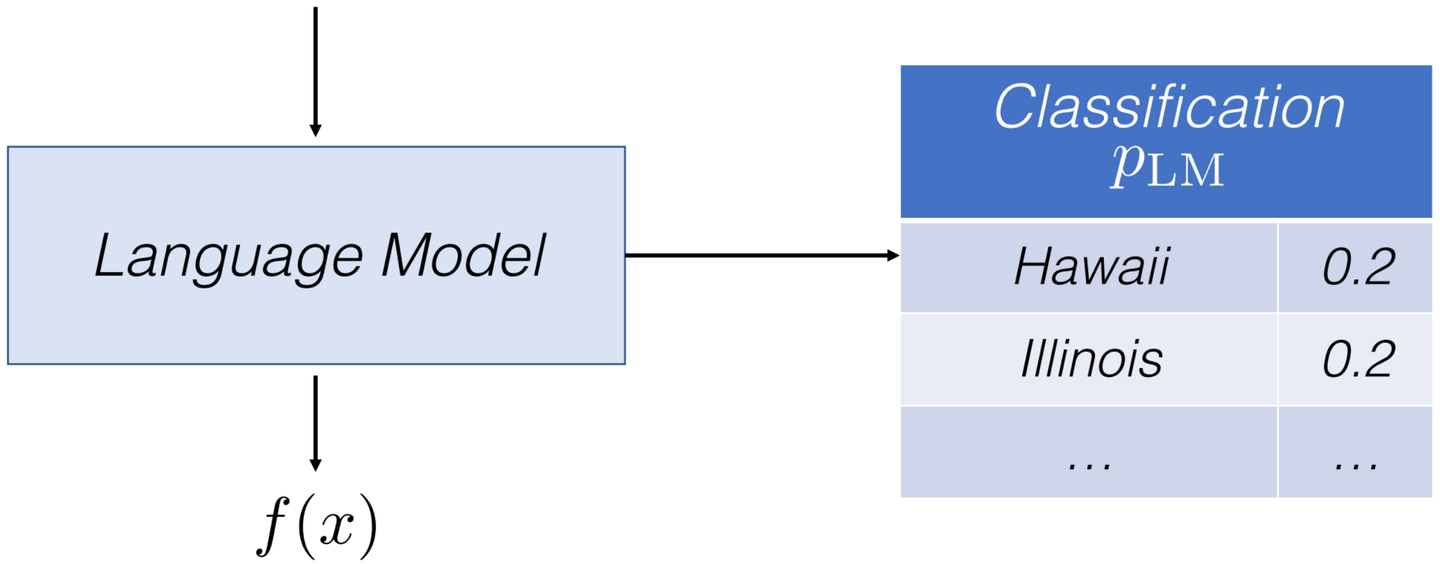
Test Context: Obama's birthplace is _???_

Previously Seen Contexts	*Targets*
Obama was senator for	*Illinois*
Barack is married to	*Michelle*
Obama was born in	*Hawaii*
…	…
Obama is a native of	*Hawaii*

Methods

Given a new test context…

$x =$ *Obama's birthplace is* _____



Classification p_{LM}	
Hawaii	*0.2*
Illinois	*0.2*
…	…

Methods

Given a new test context…

$x =$ *Obama's birthplace is* _____

$q = f(x) =$ 

Nearest Neighbors Datastore

Methods

Given a new test context…

$x =$ *Obama's birthplace is* _____

$q = f(x) =$ 

Keys	*Values*
f(Obama was senator for) *f(Obama was born in)* …	*Illinois* *Hawaii* …

Methods

Constructing the datastore

Training Contexts c_i	*Representations* $k_i = f(c_i)$	*Targets* v_i
Obama was senator for		*Illinois*
Barack is married to		*Michelle*
Obama was born in		*Hawaii*
…	…	…
Obama is a native of		*Hawaii*

Inference

The k-nearest neighbors for $q = f(x)$

Representations $k_i = f(c_i)$	*Targets* v_i	*Distances* $d_i = d(q, k_i)$
	*Illinois*	*4*
	*Michelle*	*100*
	*Hawaii*	*3*
…	…	…
	*Hawaii*	*5*

Inference

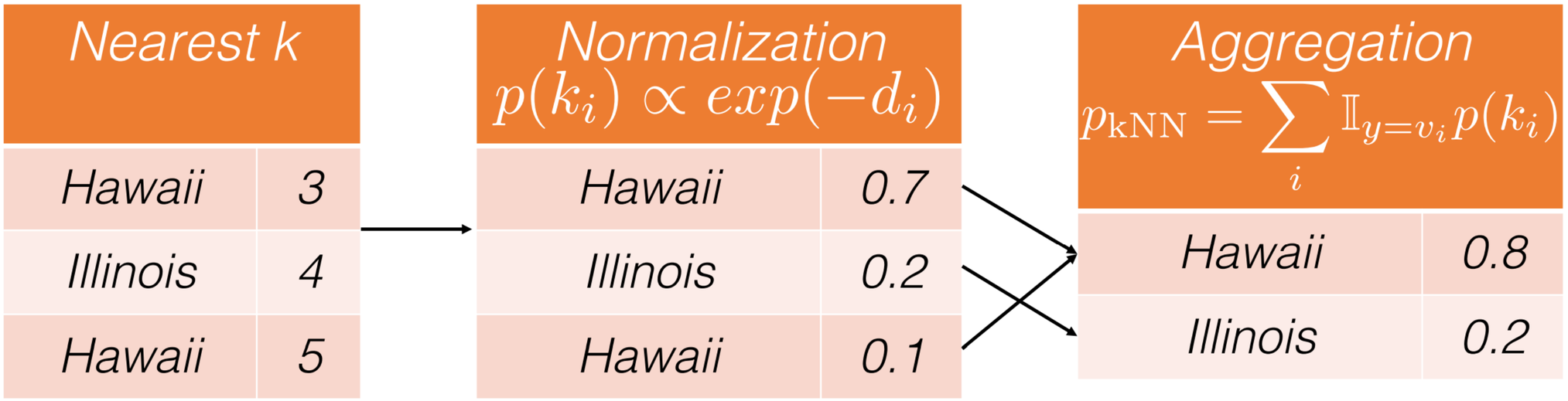
The k-nearest neighbors for $q = f(x)$

Targets v_i	*Distances* $d_i = d(q, k_i)$
Illinois	*4*
Michelle	*100*
Hawaii	*3*
…	…
Hawaii	*5*

Nearest k	
Hawaii	*3*
Illinois	*4*
Hawaii	*5*

Inference

The kNN distribution



Inference

Given a new test context…

$x =$ *Obama's birthplace is* _____

Language Model	
Hawaii	*0.2*
Illinois	*0.2*
…	…

k-Nearest Neighbors	
Hawaii	*0.8*
Illinois	*0.2*

→

kNN-LM $(1-\lambda)\ p_{\mathrm{LM}} + \lambda\ p_{\mathrm{kNN}}$	
Hawaii	*0.6*
Illinois	*0.2*
…	…

Experiments

- keys: 1024-dimensional in last layer (FFN input), quantized to 64-bytes
- FAISS index: 1M randomly sampled keys to learn 4096 cluster centroids
- k = 1024 neighbors, the index looks up 32 cluster centroids

Experiments

Model	Perplexity (↓) Dev	Test	# Trainable Params
Baevski & Auli (2019)	17.96	18.65	247M
+Transformer-XL (Dai et al., 2019)	-	18.30	257M
+Phrase Induction (Luo et al., 2019)	-	17.40	257M
Base LM (Baevski & Auli, 2019)	17.96	18.65	247M
+kNN-LM	**16.06**	**16.12**	247M
+Continuous Cache (Grave et al., 2017c)	17.67	18.27	247M
+kNN-LM + Continuous Cache	**15.81**	**15.79**	247M

Table 1: Performance on WIKITEXT-103. The kNN-LM substantially outperforms existing work. Gains are additive with the related but orthogonal continuous cache, allowing us to improve the base model by almost 3 perplexity points with no additional training. We report the median of three random seeds.

Explicitly memorizing the training data helps generalization.

Experiments

Training Data	Datastore	Perplexity (↓) Dev	Test
WIKI-3B	-	16.11	15.17
WIKI-100M	-	20.99	19.59
WIKI-100M	WIKI-3B	14.61	13.73

Table 3: Experimental results on WIKI-3B. The model trained on 100M tokens is augmented with a datastore that contains about 3B training examples, outperforming the vanilla LM trained on the entire WIKI-3B training set.

LMs can scale to larger text collections without the added cost of training, by simply adding the data to the datastore.

Experiments

Training Data	Datastore	Perplexity (↓) Dev	Test
WIKI-3B	-	37.13	34.84
BOOKS	-	14.75	11.89
WIKI-3B	BOOKS	24.85	20.47

Table 4: Domain adaptation experiments, with results on BOOKS. Adding an in-domain datastore to a Wikipedia-trained model improves results by 23 points, approaching in-domain training.

A single LM can adapt to multiple domains without the in-domain training, by adding domain-specific data to the datastore.

Experiments

- key function

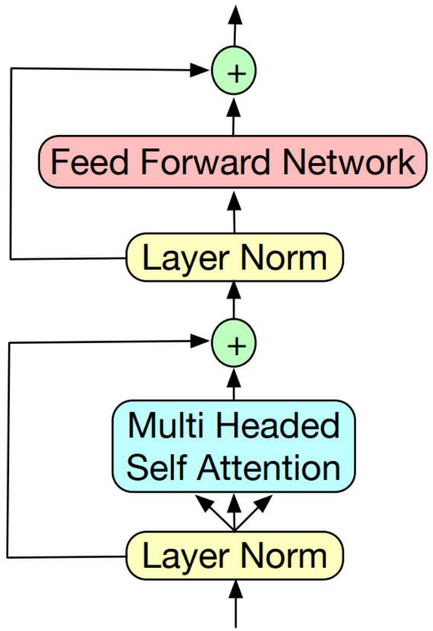


Figure 3: Transformer LM layer.

Key Type	Dev ppl. (↓)
No datastore	17.96
Model output	17.07
Model output layer normalized	17.01
FFN input after layer norm	**16.06**
FFN input before layer norm	17.06
MHSA input after layer norm	16.76
MHSA input before layer norm	17.14

Table 5: WIKITEXT-103 validation results using different states from the final layer of the LM as the representation function $f(\cdot)$ for keys and queries. We retrieve k=1024 neighbors and λ is tuned for each.

why kNN-LM improves performance

kNN-LM is most helpful typically contain rare patterns. Examples include factual knowledge, names, and near-duplicate sentences from the training set

Test Context ($p_{\text{kNN}} = 0.998$, $p_{\text{LM}} = 0.124$)	**Test Target**
it was organised by New Zealand international player Joseph Warbrick, promoted by civil servant Thomas Eyton, and managed by James Scott, a publican. The Natives were the first New Zealand team to perform a haka, and also the first to wear all black. They played 107 rugby matches during the tour, as well as a small number of Victorian Rules football and association football matches in Australia. Having made a significant impact on the...	development

Training Set Context	**Training Set Target**	**Context Probability**
As the captain and instigator of the 1888-89 Natives – the first New Zealand team to tour the British Isles – Warbrick had a lasting impact on the..	development	0.998
promoted to a new first grade competition which started in 1900. Glebe immediately made a big impact on the..	district	0.00012
centuries, few were as large as other players managed. However, others contend that his impact on the..	game	0.000034
Nearly every game in the main series has either an anime or manga adaptation, or both. The series has had a significant impact on the..	development	0.00000092

Figure 6: Example where the kNN model has much higher confidence in the correct target than the LM. Although there are other training set examples with similar local n-gram matches, the nearest neighbour search is highly confident of specific and very relevant context.

why kNN-LM improves performance

- Implicit vs Explicit Memory
- Interpolating the memorizing LM improves validation perplexity by just 0.1 – compared to 1.9 from kNN-LM

kNN-LM allows the model to memorize the training data while retaining an effective similarity function.

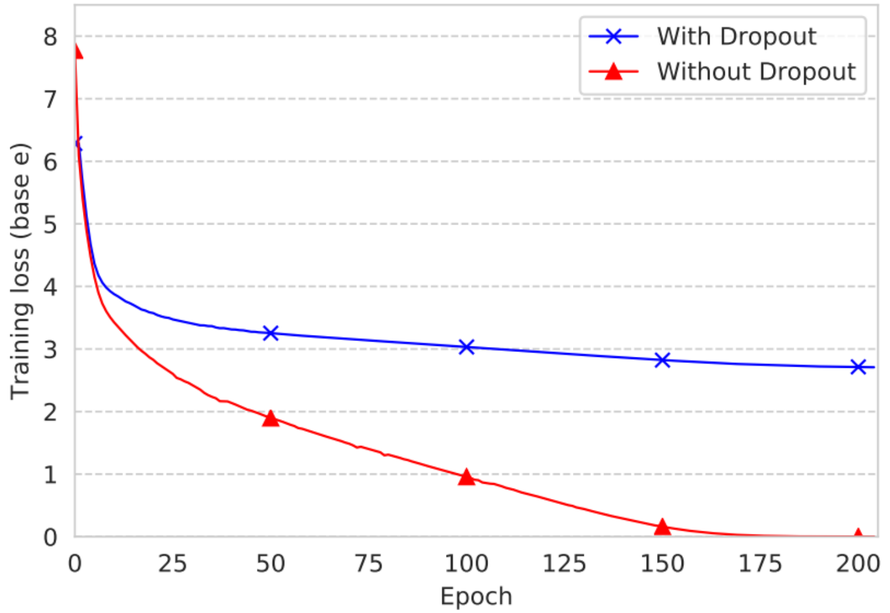


Figure 8: Training curves for the Transformer LM with and without dropout. Turning off dropout allows the training loss to go to 0, indicating that the model has sufficient capacity to memorize the training data.

Conclusion

- Explicitly memorizing the training data helps generalization, scalability and adaptation.

- Speed is slow ! (two orders of magnitude slower than base model in machine translation task)

Carnegie Mellon University
Language Technologies Institute

Neuro-Symbolic Language Modeling with Retrieval Automaton

Uri Alon
Language Technologies Institute
Carnegie Mellon University


Frank F. Xu
CMU


Junxian He
CMU

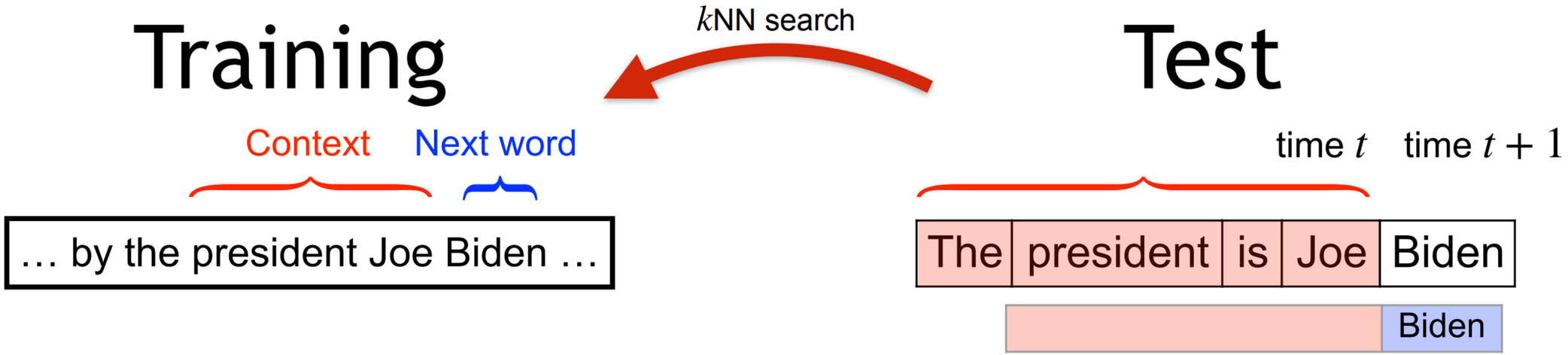

Sudipta Sengupta
Amazon AWS


Dan Roth
AWS AI Labs


Graham Neubig
CMU

ICML 2022

Background: K-Nearest Neighbor Language Model (kNN-LM)
(Khandelwal et al., ICLR'2020)



K-nearest neighbor search: for **every generated token**

time $\left(k\text{NN search}\right)$ >> time (forward pass)

If we performed **kNN search** to retrieve "Joe",
can we save the search when predicting "Biden"?

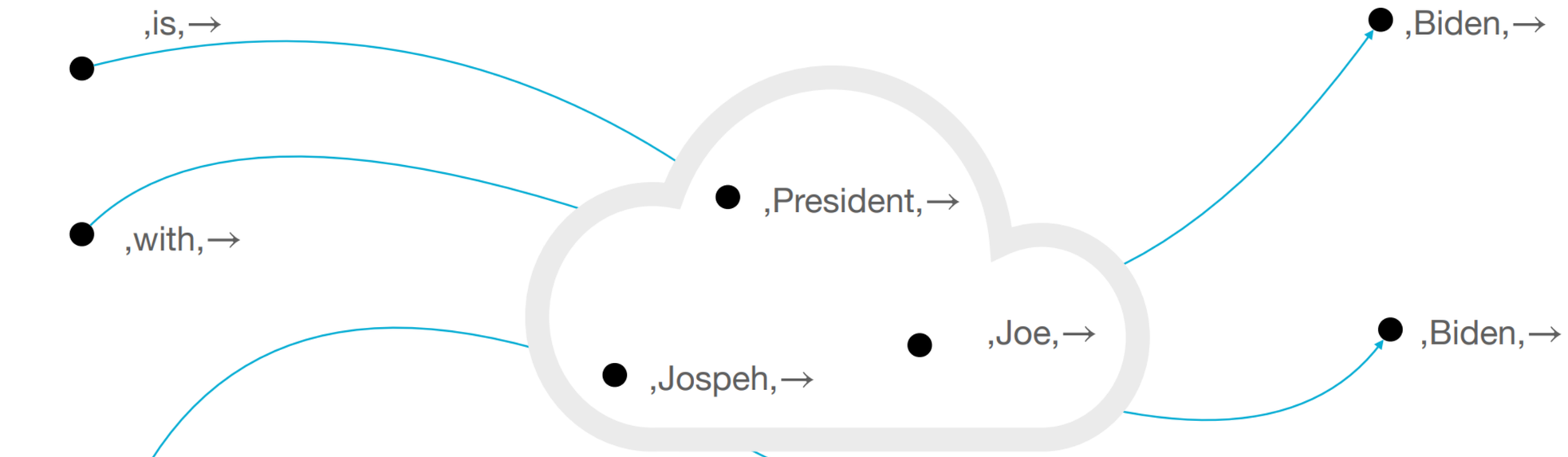
Adding Pointers Between Datastore Entries

Training

… by the president Joe Biden …

We still need to perform k**NN search** once, but in the following time steps, we can just follow pointers instead!

Clustering Entries with Close Keys



Cluster such entries, and share their outgoing pointers

👍 Capture n-grams that **were unseen at training time**

👍 Longer pointer traversal, without backing up to kNN search

Constructing the Automaton

- Pointers $f(c_i) \approx f\left(c^{(t)}\right)$ $c^{(t)} = \left(w^{(1)}, ..., w^{(t-1)}\right)$

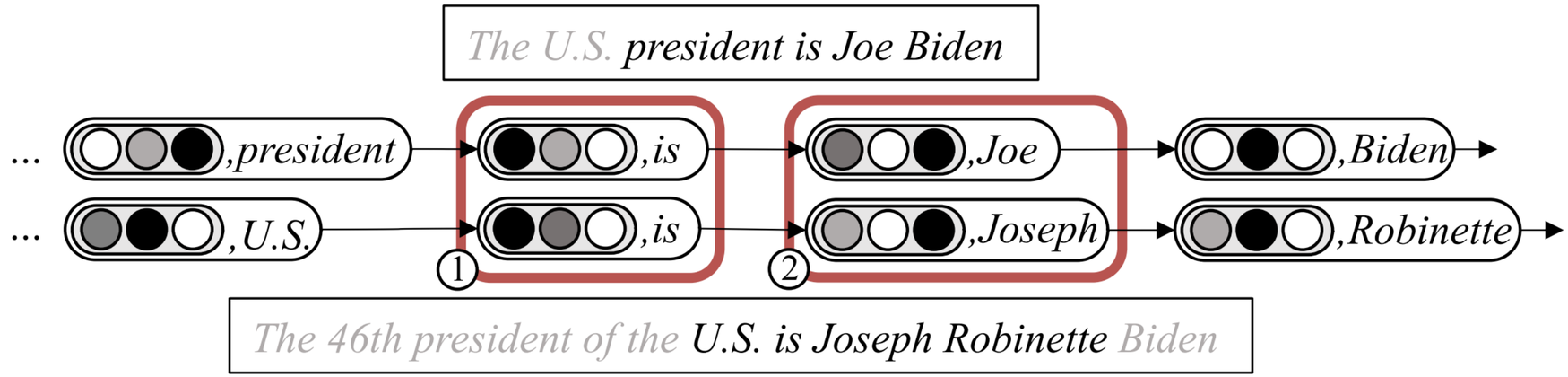
$$(f(c_i), w_i) \Rightarrow (f(c_i), w_i, p_i) \in (\mathcal{K}, \mathcal{V}, \mathcal{P})$$

- States
  - generalize to unseen phrases
  - cluster entries having close keys

- Transition States
  - allowable next states -> having the same value

Constructing the Automaton

- Pointers
- States
- Transition States



Traversal of the Automaton

- continue traversing
  - the number of new states is greater than or equal to a threshold τ
  - if not, add the remaining states we obtained

- Transition Weights

$$\phi(q, c, w) = \sum_{(k_i, w_i, \cdot) \in \pi^{-1}(q)} \mathbb{1}_{w=w_i} \exp\left(-dist\left(f(c), k_i\right)\right)$$

$$p_{\text{auto}}(w \mid c, \mathcal{S}) \propto \sum\nolimits_{q \in \mathcal{S}} \phi(q, c, w)$$

$$p(w \mid c, \mathcal{S}) = \lambda p_{\text{auto}}(w \mid c, \mathcal{S}) + (1-\lambda)\, p_{LM}(w \mid c)$$

Methods

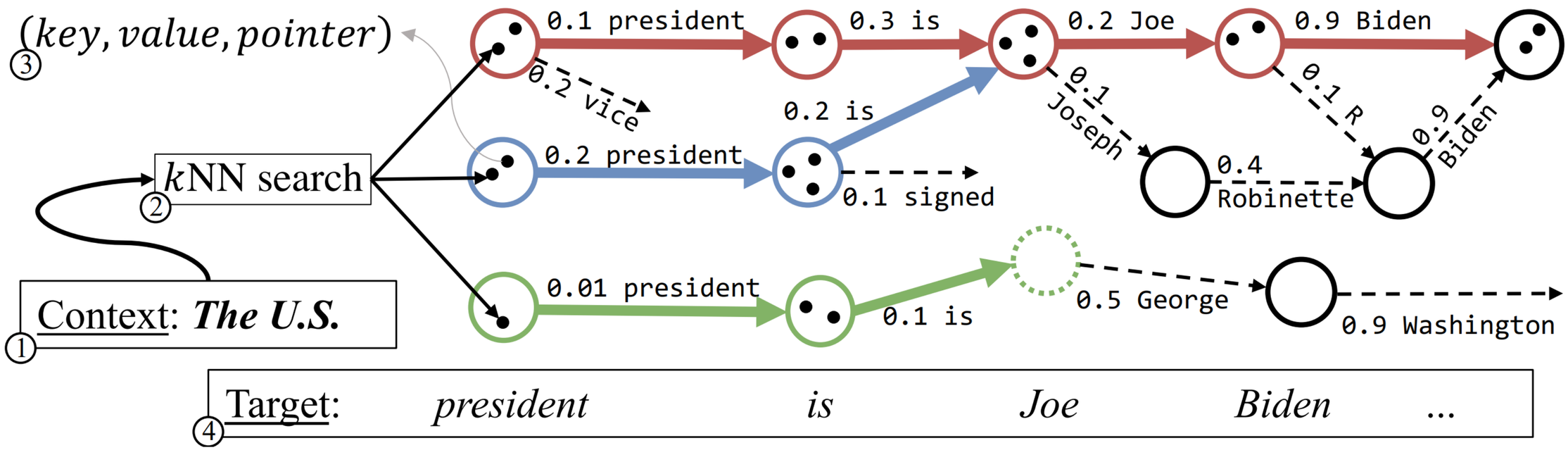


Figure 1: An illustration of RETOMATON. Given a context ① ("*The U.S.*"), a k-nearest neighbor search ② returns the nearest datastore entries. Every datastore entry (• in the figure) is a 3-tuple of $(key, value, pointer)$ ③, where the key is the LM's hidden state and the $value$ is the target token as in Khandelwal et al. (2020); the $pointer$ points to the datastore entry that appears next in the corpus. Close datastore entries are clustered together, and form an automaton state (○,○,○). The pointers of the clustered entries form the state's possible transitions. At inference time, the model decodes ④ while performing multiple parallel traversals (→, →, →) on the automaton to find useful datastore entries, instead of performing a full kNN search. Dashed arrows (⇢) denote allowed automaton transitions that were not taken during the current decoding.

In-Domain Datastore

- saves 81% of the kNN searches to achieve same results
- "w/o clustering" still saves 60%
- performs best when FoSS = 0
- pointers are helpful when FoSS<0.4, clustering allows longer sequences of consecutive steps

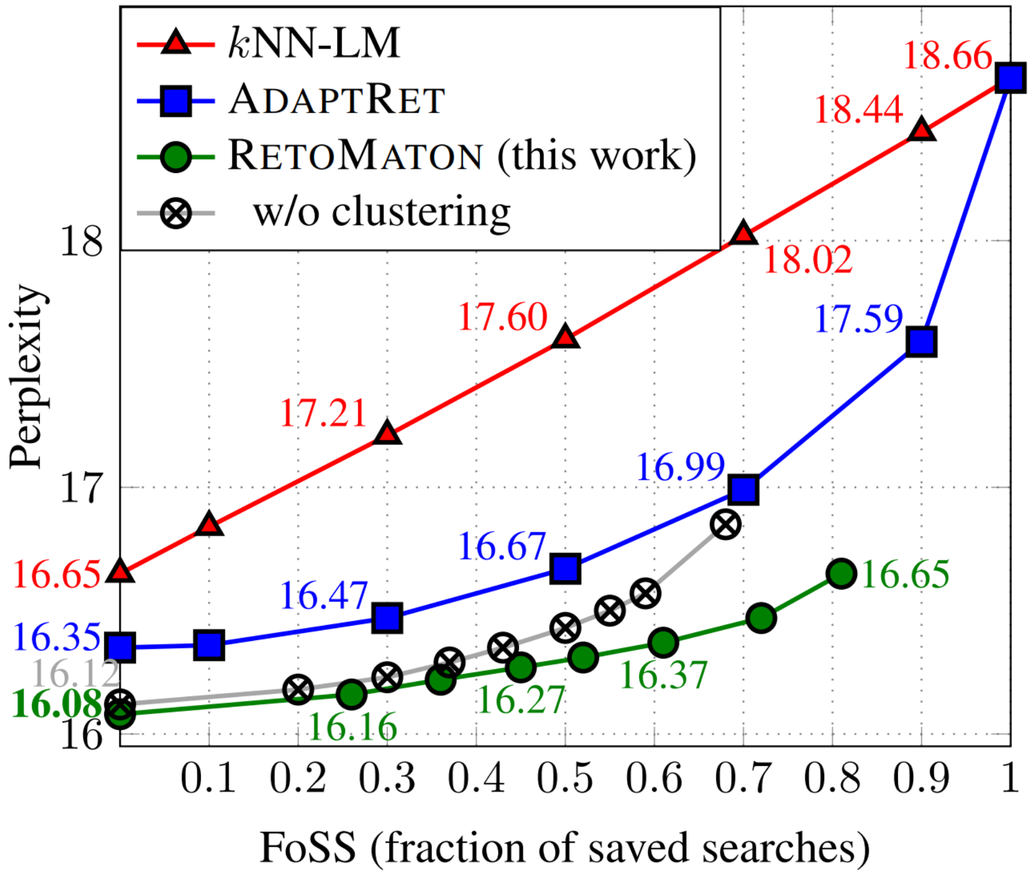


Figure 3: Experiments on WIKITEXT-103, where the datastore is created from the same training set that the base LM was trained on. RETOMATON reduces perplexity across all FoSS values, and even reduces perplexity when FoSS=0.

ADAPTRET trains a light MLP to predict at each time step whether the base LM is "confident enough" to use its output only, or should it perform a kNN search.

Domain Adaptation

Train on WMT News Crawl;

Test+build datastore on Law

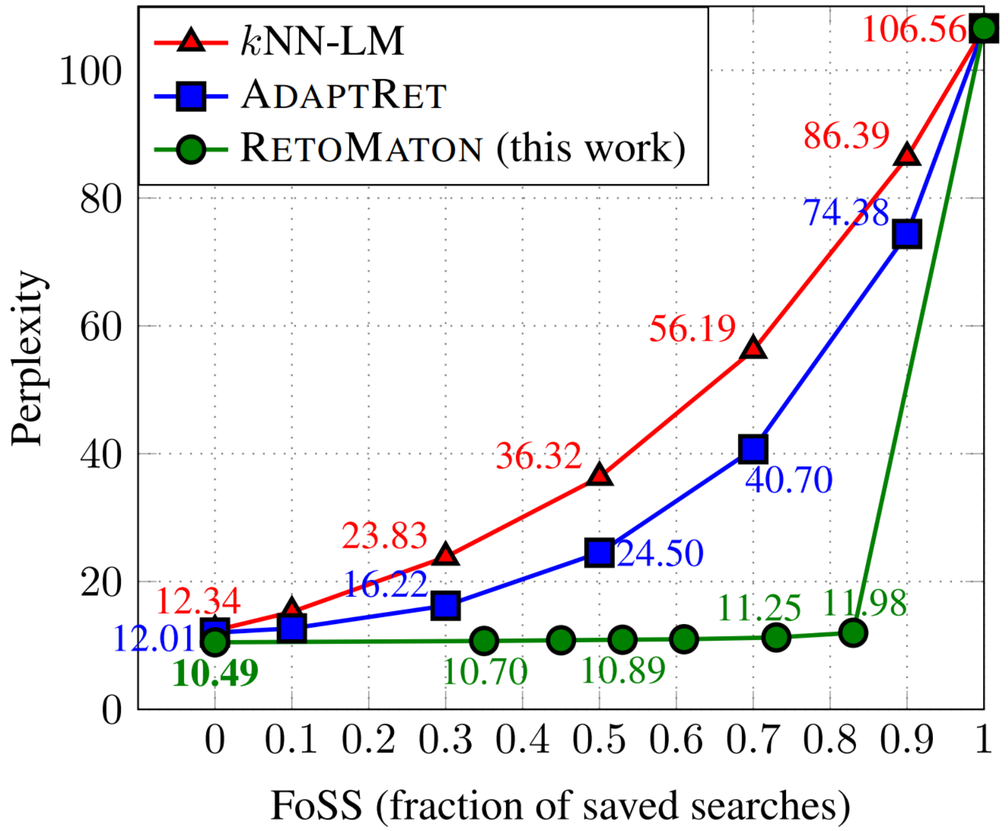


Figure 4: Experiments for domain adaptation, where the datastore is constructed from Law-MT.

Clustering Granularity

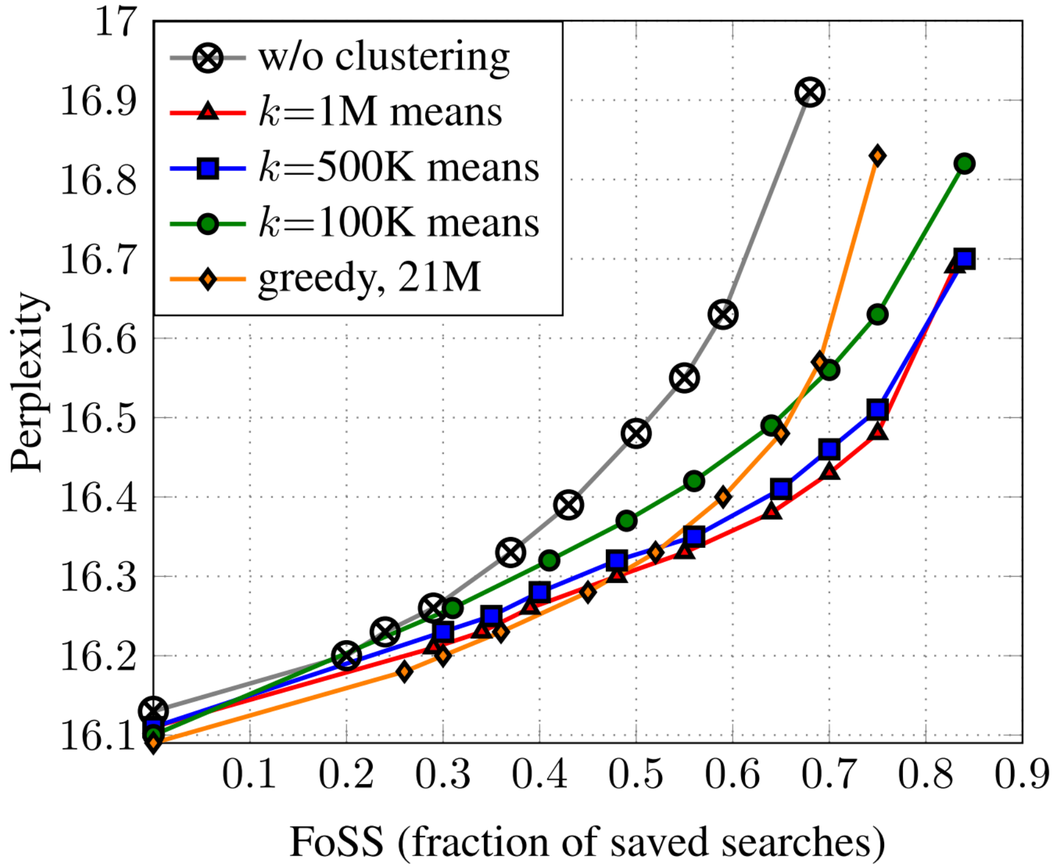


Figure 5: Analysis of the number of clusters on the validation set of WIKITEXT-103. Additional clustering runs can be found in Appendix D (Figure 9).

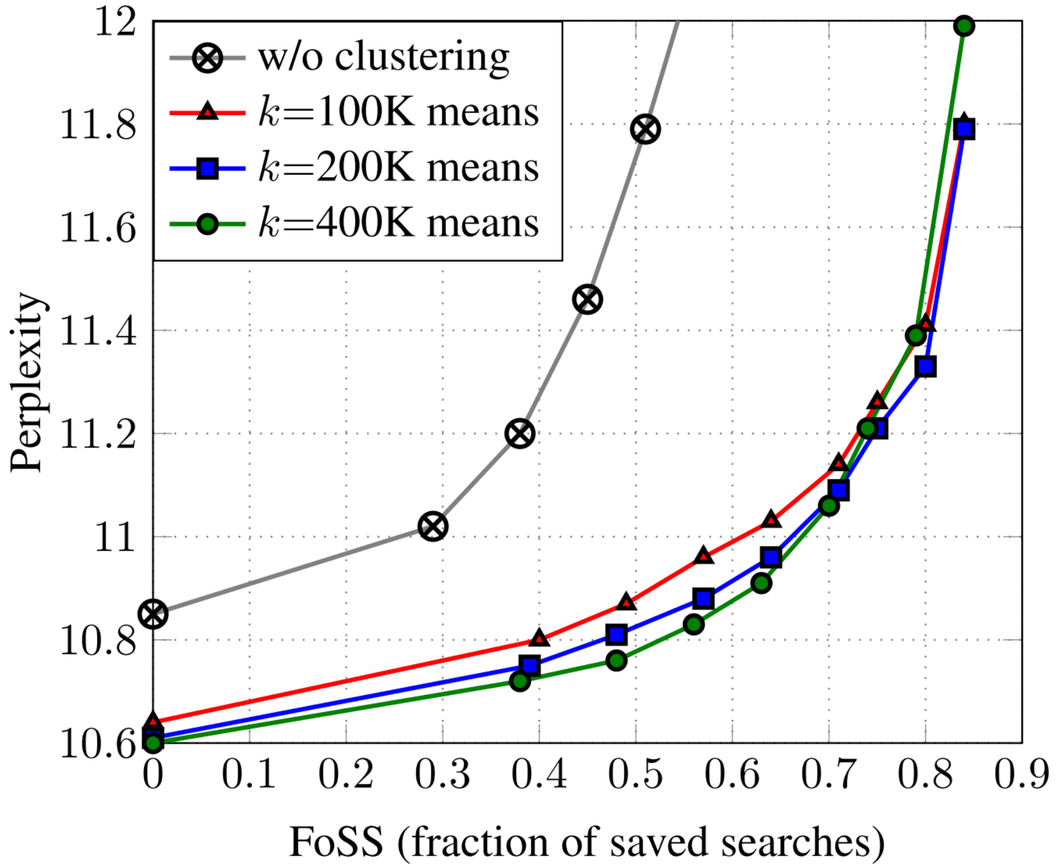


Figure 6: Analysis of the number of clusters on the validation set of Law-MT. A larger version of this figure can be found in Appendix D (Figure 10).

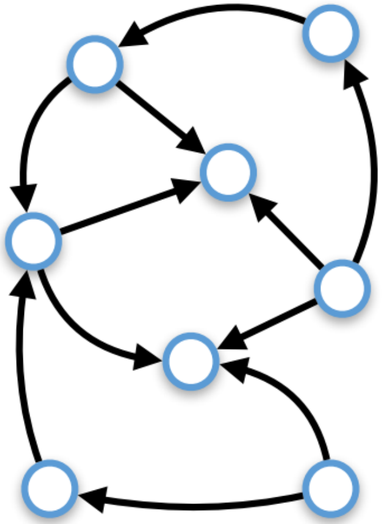
Qualitative Analysis

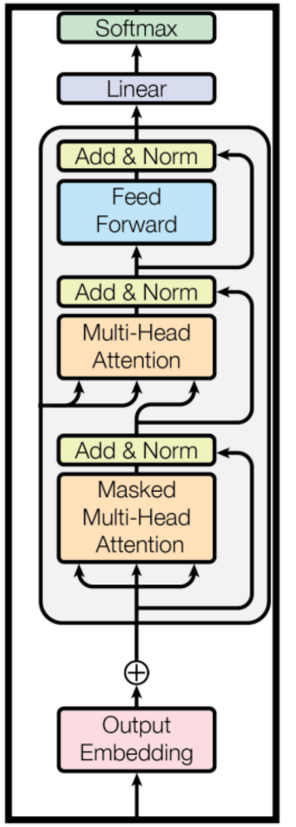
length=3	length=6	length=10
, and the	*the Streets Have No Name "*	*. As a result , there was not a single*
, but the	*Department of Transportation (MDOT)"*	*and some were moved to new locations . Before its*
roughly bounded by	*In the United States ,*	*but it was not until the following day that a*
in the first	*the end of the song ,*	*the end of the Second World War was completed in*
a number of	*not occur until 27 May 1915*	*to lack of evidence ; however , the decision was*
, when the	*fired the shots that caused the*	*the Streets Have No Name ' is more like the*

Table 2: Some of the sequences from the WIKITEXT-103 validation set that our automaton captured without performing kNN search. We selected length=10 sequences that did not appear in the training data.

RetoMaton

- Synergy between a **symbolic** automaton and a **neural** LM
  - Saving **pointers** between training entries
  - **Clustering** of entries into automaton states
  - ***Dynamic*** transition scores
- Lower perplexity than the base LM, while saving up to **83%** of the kNN searches compared to kNN-LM
- The creation of the automaton is **unsupervised**
  - Constructed from the original training data
  - Another domain





Please visit our poster session!

http://urialon.ml
ualon@cs.cmu.edu

https://github.com/neulab/retomaton
https://github.com/neulab/knn-transformers

Training Language Models with Memory Augmentation

Zexuan Zhong[†] **Tao Lei**[*] **Danqi Chen**[†]
[†]Princeton University
{zzhong, danqic}@cs.princeton.edu, taole@google.com

EMNLP 2022

Motivation

- introduce memory in test time -> suboptimal training of LMs
- training LMs with memory augmentation

Training with In-batch Memories

- the next-token probability distribution

$$P(w \mid c) \propto \exp(E_w^\top f_\theta(c)) + \sum_{(c_j, x_j) \in \mathcal{M}_{\text{train}} : x_j = w} \exp(\text{sim}(g_\theta(c), g_\theta(c_j))).$$

g_θ input of the final FFN layer

$$\text{sim}(q, k) = \frac{q \cdot k}{\sqrt{d}}$$

- minimize the negative log-likelihood of next token

Memory Augmentation

- Local memory

Goal: reduce the discrepancy between training and testing

$$\mathcal{M}_{\text{local}}(c_t) = \{(c_j, x_j)\}_{1 \le j \le t-1}. \qquad c_t = x_1, \ldots, x_{t-1}$$

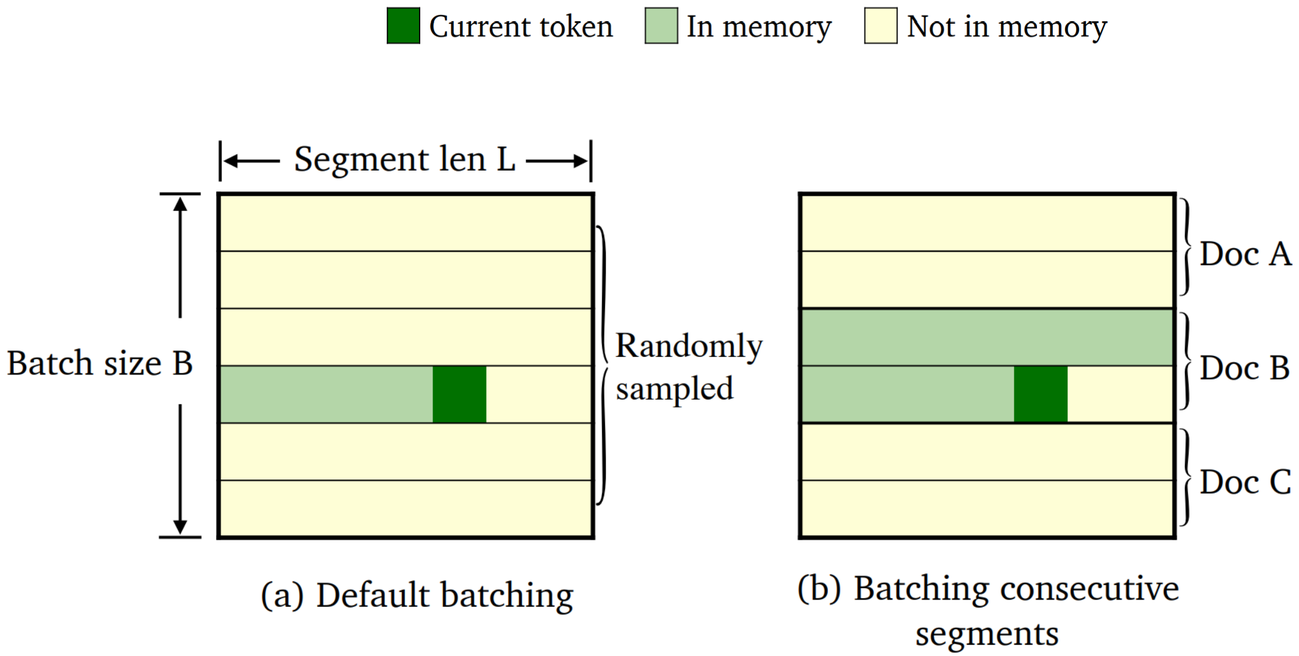
- Long-term memory

$$\mathcal{M}_{\text{long}}(c_t^{(i)}) = \{(c_j^{(k)}, x_j^{(k)})\}_{1 \le k < i, 1 \le j \le L}. \qquad \underbrace{s^{(1)}, \ldots, s^{(T)}}_{\text{document}} \qquad \underbrace{s^{(i)} = \{c_1^{(i)}, \ldots, c_L^{(i)}\}}_{\text{segment}}$$

- External memory

$$\mathcal{M}_{\text{ext}} = \{(c_j, x_j) \in \mathcal{D}\}. \qquad |\mathcal{M}_{\text{ext}}| \text{ is large, e.g., } 10^8$$

Adaption to Different Memories



(a) Default batching

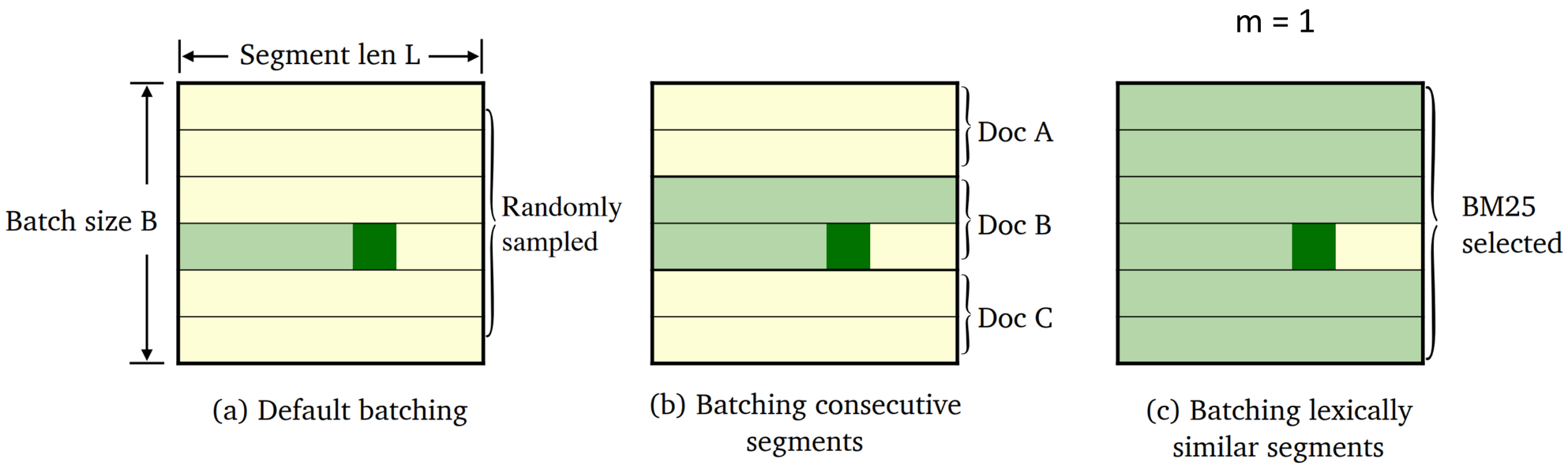
(b) Batching consecutive segments

m > 1

	Training Memory	**Testing Memory**
vanilla LM	None	None
cont. cache	None	$\mathcal{M}_{\text{local}}$ or $\mathcal{M}_{\text{long}}$
kNN-LM	None	$\mathcal{M}_{\text{ext}}$
TRIMELM	$\mathcal{M}_{\text{local}}$	$\mathcal{M}_{\text{local}}$
TRIMELM$_{\text{long}}$	§4.2	$\mathcal{M}_{\text{local}}$, $\mathcal{M}_{\text{long}}$
TRIMELM$_{\text{ext}}$	§4.3	$\mathcal{M}_{\text{local}}$, $\mathcal{M}_{\text{long}}$, $\mathcal{M}_{\text{ext}}$

Adaption to Different Memories

- External Memory
  - packing segments that have large lexical overlap using BM25, K=1024



exclude the local memory with p=90% during training to encourage model focus on information from other documents

TRIMELM vs. vanilla LM

Warmup -> train as formal LM in the first 5% updates (more stable)

Model	#Params	Dev (↓)	Test (↓)	Speed (↑)
Transformer (Baevski and Auli, 2019)	247M	17.96	18.65	-
+ continuous cache (Grave et al., 2017b)	247M	17.67	18.27	-
Transformer-XL (Dai et al., 2019)	257M	-	18.30	-
Transformer (our run)	247M	18.04	18.70	3.6k t/s
+ continuous cache	247M	17.65	18.26 ↓0.44	3.6k t/s
★ TRIMELM	247M	17.10	17.76 ↓0.94	3.6k t/s
★ $\text{TRIMELM}_{\text{long}}$	247M	**17.01**	**17.64** ↓1.06	3.6k t/s
kNN-LM (our run)	247M	16.40	16.37	300 t/s
+ continuous cache	247M	16.23	16.23 ↓0.14	300 t/s
★ $\text{TRIMELM}_{\text{ext}}$ (w/o $\mathcal{M}_{\text{long}}$)	247M	15.62	15.55 ↓0.82	300 t/s
★ $\text{TRIMELM}_{\text{ext}}$	247M	**15.51**	**15.41** ↓0.94	300 t/s
kNN-LM (Khandelwal et al., 2020)†	247M	16.06	16.12	50 t/s
+ continuous cache (Grave et al., 2017b)†	247M	15.81	15.79 ↓0.33	50 t/s
★ $\text{TRIMELM}^{\dagger}_{\text{ext}}$	247M	**15.40**	**15.37** ↓0.75	50 t/s

Table 2: Performance of our TRIME models on WIKITEXT-103 (247M models, L = 3,072). †: the results are based on computing actual distances instead of using approximated distances returned by FAISS indexes, which requires a large SSD storage. To measure the speed of models (tokens/second), we run the model with a single NVIDIA RTX 3090 GPU and run the FAISS indexer with 32 CPUs.

Domain adaptation

Model	$\mathcal{M}_{\text{ext}}$	**Dev** (↓)	**Test** (↓)
Transformer	-	62.72	53.98
★ TRIMELM	-	59.39	49.25
★ TRIMELM$_{\text{long}}$	-	49.21	**39.50**
kNN-LM + cont. cache	WIKI	53.27	43.24
★ TRIMELM$_{\text{ext}}$	WIKI	47.00	37.70
kNN-LM + cont. cache	BOOKS	42.12	32.87
★ TRIMELM$_{\text{ext}}$	BOOKS	36.97	**27.84**

Table 5: Domain-adaption performance on the BOOKSCORPUS dataset. All models are trained on WIKITEXT-103 and evaluated on BOOKSCORPUS without re-training or fine-tuning and we consider using WIKITEXT-103 and BOOKSCORPUS to build the external datastore respectively. We use a long-term memory of a size 49,152 for TRIMELM$_{\text{long}}$, TRIMELM$_{\text{ext}}$, and continuous cache in this experiment.

Conclusion

- TRIME: a training objective which leverages in-batch memories + three ways of memory construction

- adds very little computational overhead and does not modify model architectures