Key Idea #1: Pointers Between Examples

Encode the training set as linked lists of (key, value, pointer):

- `encode("by the president")` Joe
- `encode("by the president Joe")` Biden
- `encode("...")`...

This extends kNN-LM (Khandelwal et al., ’2020) by adding pointers between consecutive datastore entries.

Key Idea #2: Clustering Similar Keys

Cluster similar keys into automaton states:

- is
- with
- President
- Biden
- Joseph
- Joe
- Robinette

Retomaton

Test context: The president is ___

Automaton

- Nodes: clusters of training examples, encoded by the LM
- Edges: pointers between consecutive examples, shared in cluster
- Weights: $-||h_t, key||_2$

In-domain Datastore

- kNN-LM (Khandelwal, 2020)
- AdaptRet (He, 2021)
- Retomaton (this work)

Results

In-domain Datastore

- FoSS (fraction of saved searches)
- Perplexity
- kNN-LM (Khandelwal, 2020)
- AdaptRet (He, 2021)
- Retomaton (this work)
- w/o clustering

Domain Adaptation

- kNN-LM (Khandelwal, 2020)
- AdaptRet (He, 2021)
- Retomaton (this work)

Improving Fine-tuning

- Perplexity
- Fine-tuning kNN-LM AdaptRet Retomaton

Sample

- A sample of the automaton constructed from WIKITEXT-103

Figure: Experiments on WIKITEXT-103, where the datastore is created from the same training set that the base LM was trained on.

Figure: Domain adaptation experiments: the model was trained on News Crawl, and the datastore is constructed from Law-MT.

Figure: When constructing Retomaton on top of a fine-tuned model, Retomaton reduces perplexity by 17.5%.