Episodic Memory in Lifelong Language Learning

Abstract

We introduce a lifelong language learning setup where a model needs to learn from a stream of text examples without any dataset identifier. We propose an episodic memory model that performs sparse experience replay and local adaptation to mitigate catastrophic forgetting in this setup. Experiments on text classification and question answering demonstrate the complementary benefits of sparse experience replay and local adaptation to allow the model to continuously learn from new datasets. We also show that the space complexity of the episodic memory module can be reduced significantly (~50-90%) by randomly choosing which examples to store in memory with a minimal decrease in performance. We consider an episodic memory component as a crucial building block of general linguistic intelligence and see our model as a first step in that direction.

Introduction

The ability to continuously learn and accumulate knowledge throughout a lifetime and reuse it effectively to adapt to a new problem quickly is a hallmark of general intelligence. State-of-the-art machine learning models work well on a single dataset given enough training examples, but they often fail to isolate and reuse previously acquired knowledge when the data distribution shifts (e.g., when presented with a new dataset)—a phenomenon known as catastrophic forgetting BIBREF0 , BIBREF1 .

The three main approaches to address catastrophic forgetting are based on: (i) augmenting the loss function that is being minimized during training with extra terms (e.g., a regularization term, an optimization constraint) to prevent model parameters learned on a new dataset from significantly deviating from parameters learned on previously seen datasets BIBREF2 , BIBREF3 , BIBREF4 , (ii) adding extra learning phases such as a knowledge distillation phase, an experience replay BIBREF5 , BIBREF6 , and (iii) augmenting the model with an episodic memory module BIBREF7 . Recent methods have shown that these approaches can be combined—e.g., by defining optimization constraints using samples from the episodic memory BIBREF8 , BIBREF9 .

In language learning, progress in unsupervised pretraining BIBREF10 , BIBREF11 , BIBREF12 has driven advances in many language understanding tasks BIBREF13 , BIBREF14 . However, these models have been shown to require a lot of in-domain training examples, rapidly overfit to particular datasets, and are prone to catastrophic forgetting BIBREF15 , making them unsuitable as a model of general linguistic intelligence.

In this paper, we investigate the role of episodic memory for learning a model of language in a lifelong setup. We propose to use such a component for sparse experience replay and local adaptation to allow the model to continually learn from examples drawn from different data distributions. In experience replay, we randomly select examples from memory to retrain on. Our model only performs experience replay very sparsely to consolidate newly acquired knowledge with existing knowledge in the memory into the model. We show that a 1% experience replay to learning new examples ratio is sufficient. Such a process bears some similarity to memory consolidation in human learning BIBREF16 . In local adaptation, we follow Memory-based Parameter Adaptation BIBREF7 and use examples retrieved from memory to update model parameters used to make a prediction of a particular test example.

Our setup is different than a typical lifelong learning setup. We assume that the model only makes one pass over the training examples, similar to BIBREF9 . However, we also assume neither our training nor test examples have dataset identifying information (e.g., a dataset identity, a dataset descriptor). Our experiments focus on lifelong language learning on two tasks—text classification and question answering. BIBREF17 show that many language processing tasks (e.g., classification, summarization, natural language inference, etc.) can be formulated as a question answering problem. We argue that our lifelong language learning setup—where a model is presented with question-answer examples without an explicit identifier about which dataset (distribution) the examples come from—is a more realistic setup to learn a general linguistic intelligence model.

Our main contributions in this paper are:

Model

We consider a continual (lifelong) learning setup where a model needs to learn from a stream of training examples INLINEFORM0 . We assume that all our training examples in the series come from multiple datasets of the same task (e.g., a text classification task, a question answering task), and each dataset comes one after the other. Since all examples come from the same task, the same model can be used to make predictions on all examples. A crucial difference between our continual learning setup and previous work is that we do not assume that each example comes with a dataset descriptor (e.g., a dataset identity). As a result, the model does not know which dataset an example comes from and when a dataset boundary has been crossed during training. The goal of learning is to find parameters INLINEFORM1 that minimize the negative log probability of training examples under our model: INLINEFORM2

Our model consists of three main components: (i) an example encoder, (ii) a task decoder, and (iii) an episodic memory module. Figure FIGREF6 shows an illustration of our complete model. We describe each component in detail in the following.

Example Encoder

Our encoder is based on the Transformer architecture BIBREF19 . We use the state-of-the-art text encoder BERT BIBREF12 to encode our input INLINEFORM0 . BERT is a large Transformer pretrained on a large unlabeled corpus on two unsupervised tasks—masked language modeling and next sentence prediction. Other architectures such as recurrent neural networks or convolutional neural networks can also be used as the example encoder.

In text classification, INLINEFORM0 is a document to be classified; BERT produces a vector representation of each token in INLINEFORM1 , which includes a special beginning-of-document symbol CLS as INLINEFORM2 . In question answering, INLINEFORM3 is a concatenation of a context paragraph INLINEFORM4 and a question INLINEFORM5 separated by a special separator symbol SEP.

Task Decoder

In text classification, following the original BERT model, we take the representation of the first token INLINEFORM0 from BERT (i.e., the special beginning-of-document symbol) and add a linear transformation and a softmax layer to predict the class of INLINEFORM1 . INLINEFORM2

Note that since there is no dataset descriptor provided to our model, this decoder is used to predict all classes in all datasets, which we assume to be known in advance.

For question answering, our decoder predicts an answer span—the start and end indices of the correct answer in the context. Denote the length of the context paragraph by INLINEFORM0 , and INLINEFORM1 . Denote the encoded representation of the INLINEFORM2 -th token in the context by INLINEFORM3 . Our decoder has two sets of parameters: INLINEFORM4 and INLINEFORM5 . The probability of each context token being the start of the answer is computed as: INLINEFORM6

We compute the probability of the end index of the answer analogously using INLINEFORM0 . The predicted answer is the span with the highest probability after multiplying the start and end probabilities. We take into account that the start index of an answer needs to precede its end index by setting the probabilities of invalid spans to zero.

Episodic Memory

Our model is augmented with an episodic memory module that stores previously seen examples throughout its lifetime. The episodic memory module is used for sparse experience replay and local adaptation to prevent catastrophic forgetting and encourage positive transfer. We first describe the architecture of our episodic memory module, before discussing how it is used at training and inference (prediction) time in § SECREF3 .

The module is a key-value memory block. We obtain the key representation of INLINEFORM0 (denoted by INLINEFORM1) using a key network—which is a pretrained BERT model separate from the example encoder. We freeze the key network to prevent key representations from drifting as data distribution changes (i.e. the problem that the key of a test example tends to be closer to keys of recently stored examples).

For text classification, our key is an encoded representation of the first token of the document to be classified, so INLINEFORM0 (i.e., the special beginning-of-document symbol). For question answering, we first take the question part of the input INLINEFORM1 . We encode it using the key network and take the first token as the key vector INLINEFORM2 . For both tasks, we store the input and the label INLINEFORM3 as its associated memory value.

If we assume that the model has unlimited capacity, we can write all training examples into the memory.

However, this assumption is unrealistic in practice. We explore a simple writing strategy that relaxes this constraint based on random write. In random write, we randomly decide whether to write a newly seen example into the memory with some probability. We find that this is a strong baseline that outperforms other simple methods based on surprisal BIBREF20 and the concept of forgettable examples BIBREF21 in our preliminary experiments. We leave investigations of more sophisticated selection methods to future work.

Our memory has two retrieval mechanisms: (i) random sampling and (ii) INLINEFORM0 -nearest neighbors. We use random sampling to perform sparse experience replay and INLINEFORM1 -nearest neighbors for local adaptation, which are described in § SECREF3 below.

Training and Inference

Algorithm UID14 and Algorithm UID14 outline our overall training and inference procedures.

Experiments

In this section, we evaluate our proposed model against several baselines on text classification and question answering tasks.

Datasets

We use publicly available text classification datasets from BIBREF22 to evaluate our models (http://goo.gl/JyCnZq). This collection of datasets includes text classification datasets from diverse domains such as news classification (AGNews), sentiment analysis (Yelp, Amazon), Wikipedia article classification (DBPedia), and questions and answers categorization (Yahoo). Specifically, we use

AGNews (4 classes), Yelp (5 classes), DBPedia (14 classes), Amazon (5 classes), and Yahoo (10 classes) datasets. Since classes for Yelp and Amazon datasets have similar semantics (product ratings), we merge the classes for these two datasets. In total, we have 33 classes in our experiments. These datasets have varying sizes. For example, AGNews is ten times smaller than Yahoo. We create a balanced version all datasets used in our experiments by randomly sampling 115,000 training examples and 7,600 test examples from all datasets (i.e., the size of the smallest training and test sets). We leave investigations of lifelong learning in unbalanced datasets to future work. In total, we have 575,000 training examples and 38,000 test examples.

We use three question answering datasets: SQuAD 1.1 BIBREF23 , TriviaQA BIBREF24 , and QuAC BIBREF25 . These datasets have different characteristics. SQuAD is a reading comprehension dataset constructed from Wikipedia articles. It includes almost 90,000 training examples and 10,000 validation examples. TriviaQA is a dataset with question-answer pairs written by trivia enthusiasts and evidence collected retrospectively from Wikipedia and the Web. There are two sections of TriviaQA, Web and Wikipedia, which we treat as separate datasets. The Web section contains 76,000 training examples and 10,000 (unverified) validation examples, whereas the Wikipedia section has about 60,000 training examples and 8,000 validation examples. QuAC is an information-seeking dialog-style dataset where a student asks questions about a Wikipedia article and a teacher answers with a short excerpt from the article. It has 80,000 training examples and approximately 7,000 validation examples.

Models

We compare the following models in our experiments:

Enc-Dec: a standard encoder-decoder model without any episodic memory module.

A-GEM BIBREF9 : Average Gradient Episodic Memory model that defines constraints on the gradients that are used to update model parameters based on retrieved examples from the memory. In its original formulation, A-GEM requires dataset identifiers and randomly samples examples from previous datasets. We generalize it to the setting without dataset identities by randomly sampling from the episodic memory module at fixed intervals, similar to our method.

Replay: a model that uses stored examples for sparse experience replay without local adaptation. We perform experience replay by sampling 100 examples from the memory and perform a gradient update after every 10,000 training steps, which gives us a 1% replay rate.

MbPA BIBREF7 : an episodic memory model that uses stored examples for local adaptation without sparse experience replay. The original MbPA formulation has a trainable key network. Our MbPA baseline uses a fixed key network since MbPA with a trainable key network performs significantly worse.

MbPA INLINEFORM0 : an episodic memory model with randomly retrieved examples for local adaptation (no key network).

MbPA++: our episodic memory model described in § SECREF2 .

MTL: a multitask model trained on all datasets jointly, used as a performance upper bound.

Implementation Details

We use a pretrained INLINEFORM0 model BIBREF12 as our example encoder and key network. INLINEFORM1 has 12 Transformer layers, 12 self-attention heads, and 768 hidden dimensions (110M parameters in total). We use the default BERT vocabulary in our experiments.

We use Adam BIBREF26 as our optimizer. We set dropout BIBREF27 to 0.1 and INLINEFORM0 in Eq. EQREF16 to 0.001. We set the base learning rate to INLINEFORM1 (based on preliminary experiments, in line with the suggested learning rate for using BERT). For text classification, we use a training batch of size 32. For question answering, the batch size is 8. The only hyperparameter that we tune is the local adaptation learning rate INLINEFORM2 . We set the number of neighbors INLINEFORM3 and the number of local adaptation steps INLINEFORM4 . We show results with other INLINEFORM5 and sensitivity to INLINEFORM6 in § SECREF38 .

For each experiment, we use 4 Intel Skylake x86-64 CPUs at 2 GHz, 1 Nvidia Tesla V100 GPU, and 20 GB of RAM.

Results

The models are trained in one pass on concatenated training sets, and evaluated on the union of the test sets. To ensure robustness of models to training dataset orderings, we evaluate on four different orderings (chosen randomly) for each task. As the multitask model has no inherent dataset ordering, we report results on four different shufflings of combined training examples. We show the exact orderings in Appendix SECREF6 . We tune the local adaptation learning rate using the first dataset ordering for each task and only run the best setting on the other orderings.

A main difference between these two tasks is that in text classification the model acquires knowledge about new classes as training progresses (i.e., only a subset of the classes that corresponds to a particular dataset are seen at each training interval), whereas in question answering the span predictor works similarly across datasets.

Table TABREF33 provides a summary of our main results. We report (macro-averaged) accuracy for classification and INLINEFORM0 score for question answering. We provide complete per-dataset (non-averaged) results in Appendix SECREF7 . Our results show that A-GEM outperforms the standard encoder-decoder model Enc-Dec, although it is worse than MbPA on both tasks. Local adaptation (MbPA) and sparse experience replay (Replay) help mitigate catastrophic forgetting compared to Enc-Dec, but a combination of them is needed to achieve the best performance (MbPA++).

Our experiments also show that retrieving relevant examples from memory is crucial to ensure that the local adaptation phase is useful. Comparing the results from MbPA++ and MbPA INLINEFORM0 , we can see that the model that chooses neighbors randomly is significantly worse than the model that finds and uses similar examples for local adaptation. We emphasize that having a fixed key network is crucial to prevent representation drift. The original MbPA formulation that updates the key network during training results in a model that only performs slightly better than MbPA INLINEFORM1 in our preliminary experiments. Our results suggest that our best model can be improved further by choosing relevant examples for sparse experience replay as well. We leave investigations of such methods to future work.

Comparing to the performance of the multitask model MTL—which is as an upper bound on achievable performance—we observe that there is still a gap between continual models and the multitask model. MbPA++ has the smallest performance gap. For text classification, MbPA++ outperforms single-dataset models in terms of averaged performance (70.6 vs. 60.7), demonstrating the success of positive transfer. For question answering, MbPA++ still lags behind single dataset models (62.0 vs. 66.0). Note that the collection of single-dataset models have many more parameters since there is a different set of model parameters per dataset. See Appendix SECREF8 for detailed results of multitask and single-dataset models.

Figure FIGREF34 shows INLINEFORM0 score and accuracy of various models on the test set corresponding to the first dataset seen during training as the models are trained on more datasets. The figure illustrates how well each model retains its previously acquired knowledge as it learns new knowledge. We can see that MbPA++ is consistently better compared to other methods.

Analysis

Our results in § SECREF30 assume that we can store all examples in memory (for all models, including the baselines). We investigate variants of MbPA++ that store only 50% and 10% of training examples. We randomly decide whether to write an example to memory or not (with probability 0.5 or 0.1). We show the results in Table TABREF42 . The results demonstrate that while the performance of the model degrades as the number of stored examples decreases, the model is still able to maintain a reasonably high performance even with only 10% memory capacity of the full model.

We investigate the effect of the number of retrieved examples for local adaptation to the performance of the model in Table TABREF42 . In both tasks, the model performs better as the number of neighbors increases. Recall that the goal of the local adaptation phase is to shape the output distribution of a test example to peak around relevant classes (or spans) based on retrieved examples from the memory. As a result, it is reasonable for the performance of the model to increase with more neighbors (up to a limit) given a key network that can reliably compute similarities between the test example and stored examples in memory and a good adaptation method.

Training MbPA++ takes as much time as training an encoder-decoder model without an episodic memory module since experience replay is performed sparsely (i.e., every 10,000 steps) with only 100 examples. This cost is negligible in practice and we observe no significant difference in terms of wall clock time to the vanilla encoder-decoder baseline. MbPA++ has a higher space complexity for storing seen examples, which could be controlled by limiting the memory capacity.

At inference time, MbPA++ requires a local adaptation phase and is thus slower than methods without local adaptation. This can be seen as a limitation of MbPA++ (and MbPA). One way to speed it up is to parallelize predictions across test examples, since each prediction is independent of others. We set the number of local adaptation steps INLINEFORM0 in our experiments. Figure FIGREF44 shows INLINEFORM1 is needed to converge to an optimal performance.

Comparing MBpA++ to other episodic memory models, MBpA has roughly the same time and space complexity as MBpA++. A-GEM, on the other hand, is faster at prediction time (no local adaptation), although at training time it is slower due to extra projection steps and uses more memory since it needs to store two sets of gradients (one from the current batch, and one from samples from the memory). We find that this cost is not negligible when using a large encoder such as BERT.

We show examples of retrieved neighbors from our episodic memory model in Appendix SECREF9 . We observe that the model manages to retrieve examples that are both syntactically and semantically related to a given query derived from a test example.

Conclusion

We introduced a lifelong language learning setup and presented an episodic memory model that performs sparse experience replay and local adaptation to continuously learn and reuse previously acquired knowledge. Our experiments demonstrate that our proposed method mitigates catastrophic forgetting and outperforms baseline methods on text classification and question answering.

Dataset Order

We use the following dataset orders (chosen randomly) for text classification:

For question answering, the orders are:

Full Results

We show per-dataset breakdown of results in Table TABREF33 in Table TABREF54 and Table TABREF55 for text classification and question answering respectively.

Single Dataset Models

We show results of a single dataset model that is only trained on a particular dataset in Table TABREF56 .

Retrieved Examples

We show examples of retrieved examples from memory given a test example in Table TABREF57 .