Accurate Supervised and Semi-Supervised Machine Reading for Long Documents

Izzeddin Gur*
Department of Computer Science
University of California, Santa Barbara
izzeddingur@cs.ucsb.edu

Daniel Hewlett
Google Research
dhewlett@google.com

Alexandre Lacoste†
Element AI
alex.lacoste.shmu@gmail.com

Llion Jones
Google Research
llion@google.com

Abstract

We introduce a hierarchical architecture for machine reading capable of extracting precise information from long documents. The model divides the document into small, overlapping windows and encodes all windows in parallel with an RNN. It then attends over these window encodings, reducing them to a single encoding, which is decoded into an answer using a sequence decoder. This hierarchical approach allows the model to scale to longer documents without increasing the number of sequential steps. In a supervised setting, our model achieves state of the art accuracy of 76.8 on the WikiReading dataset. We also evaluate the model in a semi-supervised setting by downsampling the WikiReading training set to create increasingly smaller amounts of supervision, while leaving the full unlabeled document corpus to train a sequence autoencoder on document windows. We evaluate models that can reuse autoencoder states and outputs without fine-tuning their weights, allowing for more efficient training and inference.

1 Introduction

Recently, deep neural networks (DNNs) have provided promising results for a variety of reading comprehension and question answering tasks (Weston et al., 2014; Hermann et al., 2015; Rajpurkar et al., 2016), which require extracting precise information from documents conditioned on a query. While a basic sequence to sequence (seq2seq) model (Sutskever et al., 2014) can perform these tasks by encoding a question and document sequence and decoding an answer sequence (Hewlett et al., 2016), it has some disadvantages. The answer may be encountered early in the text and need to be stored across all the further recurrent steps, leading to forgetting or corruption; Attention can be added to the decoder to solve this problem (Hermann et al., 2015). Even with attention, approaches based on Recurrent Neural Networks (RNNs) require a number of sequential steps proportional to the document length to encode each document position. Hierarchical reading models address this problem by breaking the document into sentences (Choi et al., 2017). In this paper, we introduce a simpler hierarchical model that achieves state-of-the-art performance on our benchmark task without this linguistic structure, and use it as framework to explore semi-supervised learning for reading comprehension.

We first develop a hierarchical reader called Sliding-Window Encoder Attentive Reader (SWEAR) that circumvents the aforementioned bottlenecks of existing readers. SWEAR, illustrated in Figure 1, first encodes each question into a vector space representation. It then chunks each document into overlapping, fixed-length windows and, conditioned on the question representation, encodes each window in parallel. Inspired by recent attention mechanisms such as Hermann et al. (2015), SWEAR attends over the window representations and reduces them into a single vector for each document. Finally, the answer is decoded from this document vector. Our results show that SWEAR outperforms the previous state-of-the-art on the supervised WikiReading task (Hewlett et al., 2016), improving Mean F1 to 76.8 from the previous 75.6 (Choi et al., 2017).

While WikiReading is a large dataset with millions of labeled examples, many applications of machine reading have a much smaller number

*Work completed while interning at Google Research.
†Work completed while at Google Research.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2011–2020
Copenhagen, Denmark, September 7–11, 2017.

of labeled examples among a large set of unlabeled documents. To model this situation, we constructed a semi-supervised version of WikiReading by downsampling the labeled corpus into a variety of smaller subsets, while preserving the full unlabeled corpus (i.e., Wikipedia). To take advantage of the unlabeled data, we evaluated multiple methods of reusing unsupervised recurrent autoencoders in semi-supervised versions of SWEAR. Importantly, in these models we are able to reuse all the autoencoder parameters *without fine-tuning*, meaning the supervised phase only has to learn to condition the answer on the document and query. This allows for more efficient training and online operation: Documents can be encoded in a single pass offline and these encodings reused by all models, both during training and when answering queries. Our semi-supervised learning models achieve significantly better performance than supervised SWEAR on several subsets with different characteristics. The best-performing model reaches 66.5 with 1% of the WikiReading dataset, compared to the 2016 state of the art of 71.8 (with 100% of the dataset).

2 Problem Description

Following the recent progress on end-to-end supervised question answering (Hermann et al., 2015; Rajpurkar et al., 2016), we consider the general problem of predicting an answer A given a query-document pair (Q, D). We do not make the assumption that the answer should be present verbatim in the document.

2.1 Supervised Version

Given a document $D = \{d_1, d_2, \cdots, d_{N_D}\}$ and a query $Q = \{q_1, q_2, \cdots, q_{N_Q}\}$ as sequences of words, our task is to generate a new sequence of words that matches the correct answer $A = \{a_1, a_2, \cdots, a_{N_A}\}$. Because we do not assume that A is a subsequence of D, the answer may require blending information from multiple parts of the document, or may be precisely copied from a single location. Our proposed architecture supports both of these use cases.

The WikiReading dataset (Hewlett et al., 2016), which includes a mix of categorization and extraction tasks, is the largest dataset matching this problem description. In WikiReading, documents are Wikipedia articles, while queries and answers are Wikidata properties and values, respectively. Example Wikidata property-value pairs are `(place of birth, Paris)`, `(genre, Science Fiction)`. The dataset contains 18.58M instances divided into training, validation, and test with an 85/10/5 split. The answer is present verbatim in the document only 47.1% of the time, severely limiting models that label document spans, such as those developed for the popular SQUAD dataset (Rajpurkar et al., 2016).

2.2 Semi-Supervised Version

We also consider a semi-supervised version of the task, where an additional corpus of documents without labeled (Q, A) pairs is available. Taking advantage of the large size of the WikiReading dataset, we created a series of increasingly challenging semi-supervised problems with the following structure:

- **Unsupervised:** The entire document corpus (about 4M Wikipedia articles), with queries and answers removed.
- **Supervised:** Five smaller training sets created by sampling a random $(1\%, 0.5\%, 0.1\%)$ of the WikiReading training set, and taking (200, 100) random samples from each property in the original training set.

3 Supervised Model Architecture

We now present our model, called Sliding-Window Encoder Attentive Reader (SWEAR), shown in Figure 1, and describe its operation in a fully supervised setting. Given a (Q, D) pair, the model encodes Q into a vector space representation with a Recurrent Neural Network (RNN). The first layer of the model chunks the document D into overlapping, fixed-length windows and encodes all windows in parallel with an RNN conditioned on the question representation. The second layer attends over the window representations, reducing them into a single vector representing the latent answer. Finally, the answer sequence A is decoded from this vector using an RNN sequence decoder.

3.1 Preliminaries and Notation

Each word w comes from a vocabulary V and is associated with a vector e_w which constitutes the rows of an embedding matrix E. We denote by e^D, e^Q, and e^A the vector sequences corresponding to

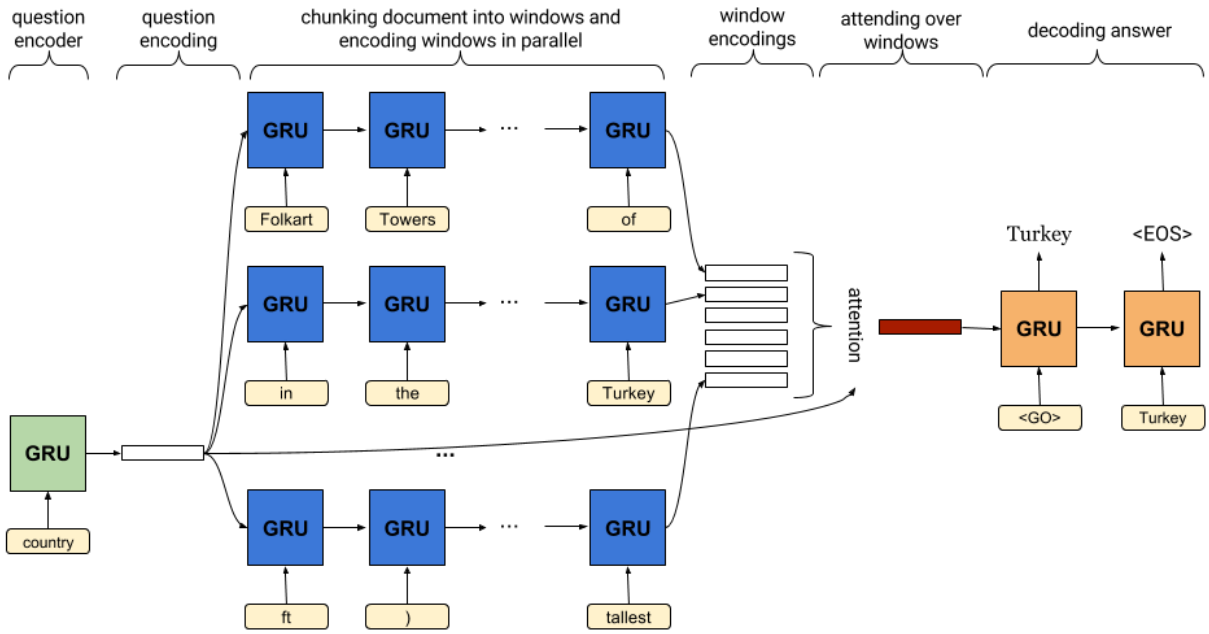

Figure 1: SWEAR model: Boxes are RNN cells, colors indicate parameter sharing.

the document, question, and answer sequences, respectively. More specifically, we aim at obtaining vector representations for documents and questions, then generating the words of the answer sequence.

Our model makes extensive use of RNN encoders to transform sequences into fixed length vectors. For our purposes, an RNN encoder consists of GRU units (Cho et al., 2014) defined as

$$h_t = f(x_t; h_{t-1}; \theta) \tag{1}$$

where h_t is hidden state at time t. f is a nonlinear function operating on input vector x_t and previous state, h_{t-1} with θ being its parameter vector. Given an input sequence, the encoder runs over the sequence of words producing the hidden vectors at each step. We refer to the last hidden state of an RNN encoder as the *encoding* of a sequence.

3.2 Sliding Window Recurrent Encoder

The core of the model is a sequence encoder that operates over sliding windows in a manner analogous to a traditional convolution. Before encoding the document, we slide a window of length l with a step size s over the document and produce $n = \lfloor \frac{N_D - l}{s} \rfloor$ document windows. This yields a sequence of sub-documents $(D_1, D_2, \cdots, D_n)$, where each D_i contains a subsequence of l words from the original document D. Intuitively, a precise answer may be present verbatim in one or more windows, or many windows may contain evidence suggestive of a more categorical answer.

Next, the model encodes each window conditioned on a question encoding. We first encode the question sequence once using a RNN (Enc) as

$$h^q = Enc(e^Q; \theta_Q) \tag{2}$$

where h^q is the last hidden state and θ_Q represents the parameters of the question encoder. Initialized with this question encoding, we employ another RNN to encode each document window as

$$\begin{aligned} h^w_{i,0} &= h^q \\ h^w_i &= Enc(e^{D_i}; \theta_W) \end{aligned} \tag{3}$$

where $h^w_{i,0}$ is the initial hidden state, h^w_i is the last hidden state, and θ_W represents the parameters of the window encoder. θ_W is shared for every window and is decoupled from θ_Q. As the windows are significantly smaller than the documents, encodings of windows will reflect the effect of question encodings better, mitigating any long-distance dependency problems.

3.3 Combining Window Encodings

SWEAR attends over the window encoder states using the question encoding to produce a single vector h^d for the document, given by

$$p_i \propto \exp(u_R^T \, tanh(W_R[h^w_i, h^q])) \tag{4}$$

$$h^d = \sum_i p_i h^w_i \tag{5}$$

Model	Mean F1
Placeholder seq2seq (HE16)	71.8
SoftAttend (CH17)	71.6
Reinforce (CH17)	74.5
Placeholder seq2seq (CH17)	75.6
SWEAR (w/ zeros)	76.4
SWEAR	**76.8**

Table 1: Results for SWEAR compared to top published results on the WikiReading test set.

	HE16 Best	SWEAR
Categorical	**88.6**	**88.6**
Relational	56.5	**63.4**
Date	73.8	**82.5**

Table 2: Mean F1 for SWEAR on each type of property compared with the best results for each type reported in Hewlett et al. (2016), which come from different models. Other publications did not report these sub-scores.

where $[.]$ is vector concatenation, and p_i is the probability window i is relevant to answering the question. W_R and u_R are parameters of the attention model.

3.4 Answer Decoding

Given the document encoding h^d, an RNN decoder (Dec) generates the answer word sequence:

$$h_0^a = h^d$$
$$h_t^a = Dec(h_{t-1}^a; \omega_A) \quad (6)$$
$$P(a_t^* = w_j) \propto \exp(e_j^T(W_A h_t^a + b_A)) \quad (7)$$
$$a_t^* = argmax_j(P(a_t^* = w_j)) \quad (8)$$

where h_0^a is the initial hidden state and h_t^a is the hidden vector at time t. $A^* = \{a_1^*, a_2^*, \cdots, a_{N_A}^*\}$ is the sequence of answer words generated. W_A, b_A, and ω_A are the parameters of the answer decoder. The training objective is to minimize the average cross-entropy error between the candidate sequence A^* and the correct answer sequence A.

3.5 Supervised Results

Before exploring unsupervised pre-training, we present summary results for SWEAR in a fully supervised setting, for comparison to previous work on the WikiReading task, namely that of Hewlett et al. (2016) and Choi et al. (2017), which we refer to as HE16 and CH17 in tables. For further experiments, results, and discussion see Section 5.2. Table 1 shows that SWEAR outperforms the best results for various models reported in both publications, including the hierarchical models SoftAttend and Reinforce presented by Choi et al. (2017).[1] Interestingly, SoftAttend computes an attention over sentence encodings, analogous to SWEAR's attention over overlapping window encodings, but it does so on the basis of less powerful encoders (BoW or convolution vs RNN), suggesting that the extra computation spent by the RNN provides a meaningful boost to performance.

Doc length	pct	seq2seq	SWEAR	imp
$[0, 200)$	44.6	79.7	80.7	1.2
$[200, 400)$	19.5	76.7	77.8	1.5
$[400, 600)$	11.0	74.5	76.3	2.3
$[600, 800)$	6.6	72.8	74.3	2.1
$[800, 1000)$	4.3	71.5	72.8	1.8
$[1000, max)$	14.0	64.8	65.9	1.7

Table 3: Comparison of Mean F1 for SWEAR and a baseline seq2seq model on the WikiReading test set across different document length ranges. *pct* indicates the percentage of the dataset falling in the given document length range. *imp* is the percentage improvement of SWEAR over baseline.

To quantify the effect of initializing the window encoder with the question state, we report results for two variants of SWEAR: In *SWEAR* the window encoder is initialized with the question encoding, while in *SWEAR w/ zeros*, the window encoder is initialized with zeros. In both cases the question encoding is used for attention over the window encodings. For SWEAR w/ zeros it is additionally concatenated with the document encoding and passed through a 2-layer fully connected neural network before the decoding step. Conditioning on the question increases Mean F1 by 0.4.

Hewlett et al. (2016) grouped properties by answer distribution: *Categorical* properties have a small list of possible answers, such as countries, *Relational* properties have an open set of answers, such as spouses or places of birth, and *Date* properties (a subset of relational properties) have date answers, such as date of birth. We reproduce this grouping in Table 2 to show that SWEAR improves performance for Relational and Date properties, demonstrating that it is better able to extract precise information from documents.

Finally, we observe that SWEAR outperforms a baseline seq2seq model on longer documents, as

[1] Document lengths differ between publications: We truncate documents to the first 600 words, while Choi et al. truncate to 1000 words or 35 sentences and Hewlett et al. truncate to 300 words.

shown in Table 3. The baseline model is roughly equivalent to the best previously-published result, Placeholder seq2seq (CH17) in Table 1, reaching a Mean F1 of 75.5 on the WikiReading test-set. SWEAR improves over the baseline in every length category, but the differences are larger for longer documents.

4 Semi-Supervised Model Architecture

We now describe semi-supervised versions of the SWEAR model, to address the semi-supervised problem setting described in Section 2.2. A wide variety of approaches have been developed for semi-supervised learning with Neural Networks, with a typical scheme consisting of training an unsupervised model first, and then reusing the weights of that network as part of a supervised model. We consider each of these problems in turn, describing two types of unsupervised autoencoder models for sequences in Section 4.1 before turning to a series of strategies for incorporating the autoencoder weights into a final supervised model in Section 4.3. All of these models reuse the autoencoder weights without modification, meaning a document can be encoded once by an offline process, and the resulting encodings can be used both during training and to answer multiple queries online in a more efficient manner.

4.1 Recurrent Autoencoders for Unsupervised Pre-training

Autoencoders are models that reconstruct their input, typically by encoding it into a latent space and then decoding it back again. Autoencoders have recently proved useful for semi-supervised learning (Dai and Le, 2015; Fabius and van Amersfoort, 2014). We now describe two autoencoder models from the recent literature that we use for unsupervised learning. The Recurrent Autoencoder (RAE) is the natural application of the seq2seq framework (Sutskever et al., 2014) to autoencoding documents (Dai and Le, 2015): In seq2seq, an encoder RNN already produces a latent representation h_N, which is used to initialize a decoder RNN. In RAE, the output sequence is replaced with the input sequence, so learning minimizes the cross-entropy between the reconstructed input sequence and the original input sequence. Encoder and decoder cells share parameters θ_U.

4.1.1 Variational Recurrent Autoencoder

The Variational Recurrent Autoencoder (VRAE), introduced by Fabius et al. (2014), is a RAE with a variational Bayesian inference step where an unobserved latent random variable generates the sequential data. The encoder and decoder are exactly the same as RAE, but the latent state h_N is not directly passed to the decoder. Instead, it is used to estimate the parameters of a Gaussian distribution with a diagonal covariance matrix: The mean is given by $\mu_x = W_\mu h_N + b_\mu$ and the covariance by $\Sigma_x = W_\Sigma h_N + b_\Sigma$, where W_μ, W_Σ, b_μ, and b_Σ are new variational step parameters. The decoder is initialized with a single vector sampled from this distribution, $z_x \sim \mathcal{N}(z|\mu_x, \Sigma_x)$. For VRAE, the Kullback-Leibler divergence between trained Normal distribution and standard normal distribution, i.e., $KL(\mathcal{N}(\mu_x, \Sigma_x)|\mathcal{N}(0, I))$, is added to the loss.

4.1.2 Window Autoencoders

We take advantage of the SWEAR architecture by training autoencoders for text windows, as opposed to the standard document autoencoders. These autoencoders operate on the same sliding window subsequences as the supervised SWEAR model, autoencoding all subsequences independently and in parallel. This makes them easier to train as they only have to compress short sequences of text into a fixed-length representation. As the task of autoencoding is independent from our supervised problem, we refer to the generated encodings as *global encodings*.

4.2 Baseline: Initialization with Autoencoder Embeddings

Our baseline approach to reusing an unsupervised autoencoder in SWEAR is to initialize all embeddings with the pre-trained parameters and fix them. We call this model SWEAR-SS (for semi-supervised). The embedding matrix is fixed to the autoencoder embeddings. All other parameters are initialized randomly and trained as in the fully supervised version. We found that initializing the encoders and decoder with autoencoder weights hurts performance.

4.3 Reviewer Models

Unfortunately, this baseline approach to semi-supervised learning has significant disadvantages in our problem setting. Pre-trained RNN parameters are not fully exploited since we observed

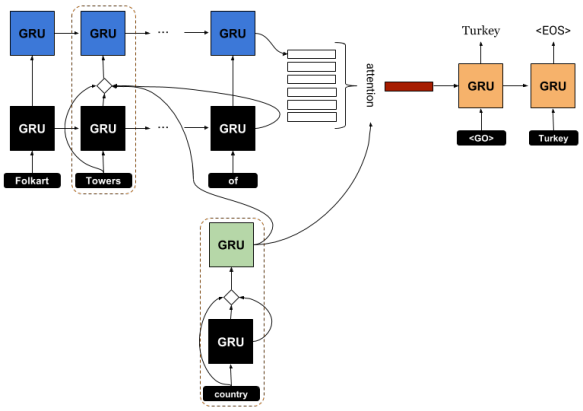

Figure 2: Multi-layer reviewer model (SWEAR-MLR), shown operating over a single window: Black boxes are RNN cells with fixed weights copied from the autoencoder, diamonds indicate vector concatenation with adapter and FC layers. For simplicity, only the cells in dashed boxes are fully illustrated (detailed in Figure 3), but the same structure is repeated for each cell.

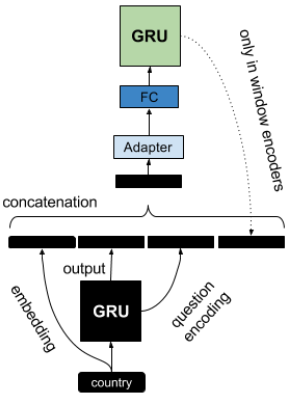

Figure 3: Detailed illustration of the dashed box in SWEAR-MLR question encoder. Black boxes are fixed parameters and encodings. The window encoder is similar, except that the output of the question reviewer layer is also added to the concatenated input (dashed line).

catastrophic forgetting when initializing and fine-tuning SWEAR with pre-trained weights. This includes fixing window encoder parameters with autoencoders and only fine-tuning question encoders. Second, conditioning the window encoders on the question eliminates the possibility to train window representations offline and utilize them later which causes a significant overhead during testing.

Inspired by recent trends in deep learning models such as Progressive Neural Networks (Rusu et al., 2016) and Reviewer Models (Yang et al., 2016), we propose multiple solutions to these problems. All of the proposed models process text input first through a fixed *autoencoder layer*: fixed pre-trained embeddings and fixed RNN encoder parameters, both initialized from the autoencoder weights. Above this autoencoder layer, we build layers of abstraction that learn to adapt the pre-trained models to the QA task.

4.3.1 Multi-Layer Reviewer (SWEAR-MLR)

The most straightforward extension to the baseline model is to fix the pretrained autoencoder RNN as the first layer and introduce a second, trainable *reviewer layer*. To make this approach more suitable for question answering, reviewer layers utilize corresponding global encodings as well as hidden states of the pre-trained autoencoders as input (Figure 2). The aim is to review both pre-trained question and window encodings to compose a single vector representing the window conditioned on the question.

Encoding questions: The question is first encoded by the autoencoder layer, $\tilde{h}^q = Enc(e^Q; \theta_U)$ where both word embeddings (E and e^Q) and encoder (θ_U) are fixed and initialized with pretrained parameters. A second, learnable RNN layer then takes the output of the autoencoder layer and corresponding input embeddings as input and produces the final question encoding, $h^q = Enc(FC([e^Q, \tilde{h}^q, \tilde{h}_t]); \theta_Q)$ where FC is a fully connected layer with ReLU activation function, and $\tilde{h}_t$ is the output of the autoencoder layer at time step t. Figure 3 illustrates a single time-step of the question encoder.

Encoding windows: Similarly, windows are encoded first by the fixed autoencoder layer and then by a reviewer layer, $\tilde{h}^w_i = Enc(e^{D_i}; \theta_U)$ and $h^w_i = Enc(FC([e^{D_i}, \tilde{h}^w_i, \tilde{h}^w_t, h^q]); \theta_W)$ where $\tilde{h}^w_t$ is the output of the autoencoder layer at time step t. Unlike supervised SWEAR, in SWEAR-MLR the window encoder is not initialized with the question encoder state. Instead, the question encoder state is an additional input to each unit in the reviewer layer (illustrated as the dashed line in Figure 3). Intuitively, the reviewer layer should reuse the global window and question information and encode only information relevant to the current question.

4.3.2 Progressive Reviewer (SWEAR-PR)

Although the reviewer layer in SWEAR-MLR has global window and question encodings as input, it requires a number of sequential steps equal to the window size, plus any additional reviewer steps. The reviewer layer also has to re-encode windows for each question, which is not ideal for online use.

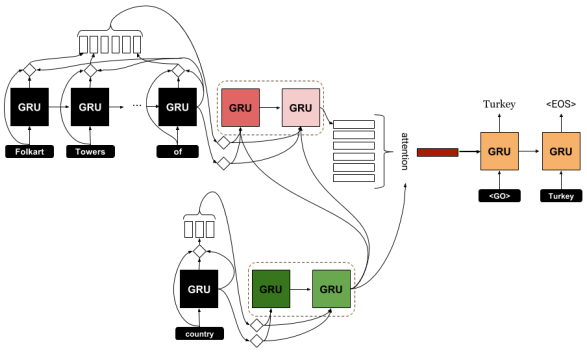

Figure 4: Progressive reviewer model (SWEAR-PR), shown operating over a single window: Black boxes are RNN cells with fixed weights copied from the autoencoder, diamonds indicate vector concatenation with adapter and FC layers. Dashed boxes contain reviewer layers. Cells within reviewer layers are decoupled as indicated by different colors.

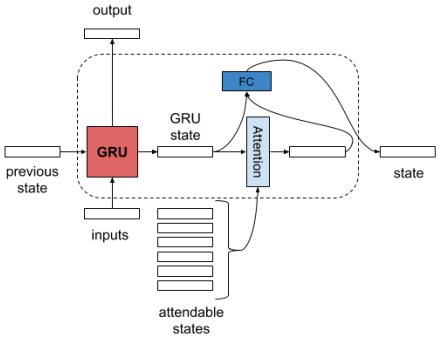

Figure 5: Illustration of attention cell: GRU state is used to attend over attendable states, then final state is computed by concatenating GRU state with context vector and passing through a fully connected neural network.

To address these issues, we now present a *Progressive Reviewer* model (SWEAR-PR) that reviews the outputs of the encoders using a separate RNN that is decoupled from the window size (Figure 4).

Encoding questions and windows: Similar to SWEAR-MLR, SWEAR-PR first encodes the questions and windows independently using autoencoder layers, $\tilde{h}^q = Enc(e^Q; \theta_U)$ and $\tilde{h}^w_i = Enc(e^{D_i}; \theta_U)$. To decouple the question and window encoders, however, SWEAR-PR does not have a second layer as a reviewer.

Reviewing questions and windows: SWEAR-PR employs two other RNNs to review the question and window encodings and to compose a single window representation conditioned on the question. Question reviewer takes the same pre-trained question encoding at each time step and attends over the hidden states and input embeddings of the pre-trained question encoder, $h^q = AttnEnc(FC(\tilde{h}^q); FC([\tilde{h}^q_t, e^Q]); \theta_Q)$ where $AttnEnc$ is an RNN with an attention cell which is illustrated in Figure 5. Outputs of the fixed autoencoder layer and fixed word embeddings, $[\tilde{h}^q_t, e^Q]$, are the attendable states. Window reviewer on the other hand takes the pre-trained window encoding and reviewed question encoding at each time step and attends over the hidden states of pre-trained window encoder, $h^w_i = AttnEnc(FC([\tilde{h}^w_i, h^q]); FC([\tilde{h}^w_t, e^{D_i}]); \theta_W)$ where outputs of the fixed autoencoder layer and fixed word embeddings, $[\tilde{h}^w_t, e^{D_i}]$, are the attendable states. As length of the windows is smaller than length of the reviewers, SWEAR-PR has significantly smaller overhead compared to other supervised and semi-supervised SWEAR variants.

4.3.3 Shared Components

Reducing window encodings and decoding: As in the supervised case described in 4, both reviewer models attend over the window encodings using the question encoding and reduce them into a single document encoding. Identical to answer decoding described in 6, the answer is decoded using another RNN taking the document state as the initial state. The parameters of this answer decoder are initialized randomly.

Adapter layer: As the distribution and scale of parameters may differ significantly between the autoencoder layer and the reviewer layer, we use an *adapter layer* similar to the adapters in Progressive Neural Networks (Rusu et al., 2016) to normalize the pre-trained parameters:

$$W_{out} = a * tanh(b * W_{in}) \quad (9)$$

where a and b are scalar variables to be learnt and W_{in} is a pre-trained input parameter. We put adapter layers after every pre-trained parameter connecting to a finetuned parameter such as on the connections from pre-trained embeddings to reviewer layer. We use dropout (Srivastava et al., 2014) regularization on both inputs and outputs of the reviewer cells.

5 Experimental Evaluation

As described in Section 2, we evaluate our models on the WikiReading task. In Section 3.5 we presented results for the supervised SWEAR on the full WikiReading dataset, establishing it as the highest-scoring method so far developed for WikiReading. We now compare our semi-supervised models SWEAR-MLR and SWEAR-

Model	1%	0.5%	0.1%
SWEAR	63.5	57.6	39.5
SWEAR-SS (RAE)	64.7	62.8	55.3
SWEAR-SS (VRAE)	**65.7**	**64.0**	**60.7**

Table 4: Mean F1 results for SWEAR (fully supervised) and SWEAR-SS (semi-supervised) trained on 1%, 0.5%, and 0.1% subsets, respectively. Variants of SWEAR-SS indicate different sources of fixed encoder weights. [2]

PR over various subsets of the WikiReading dataset, using SWEAR as a baseline.

5.1 Experimental Setup

Following Hewlett et al. (2016), we use the *Mean F1* metric for WikiReading, which assigns partial credit when there are multiple valid answers. We ran hyperparameter tuning for all models and report the result for the configuration with the highest Mean F1 on the validation set.

The supervised SWEAR model was trained on both the full training (results reported in Section 3.5) and on each subset of training data (results reported below). Unsupervised autoencoders were trained on all documents in the WikiReading training set. We selected the autoencoder with the lowest reconstruction error for use in semi-supervised experiments. After initialization with weights from the best autoencoder, learnable parameters in the semi-supervised models were trained exactly as in the supervised model.

Training Details

We implemented all models in a shared framework in TensorFlow (Abadi et al., 2016). We used the Adam optimizer (Kingma and Ba, 2014) for all training, periodically halving the learning rate according to a hyperparameter. Models were trained for a maximum of 4 epochs.

Table 7 shows which hyperparameters were tuned for each type of model, and the range of values for each hyperparameter. The parameters in the second group of the table are tuned for supervised SWEAR and the best setting (shown in bold) was used for other models where applicable. We fixed the batch size to 8 for autoencoders and 64 for semi-supervised models. We used a truncated normal distribution with a standard deviation of 0.01 for VRAE.

5.2 Results and Discussion

[2]Initialization with Word2Vec (Mikolov et al., 2013) embeddings on 1% subset gives 64.0 Mean F1 score.

Model	100	200
SWEAR	25.0	33.0
SWEAR-SS (VRAE)	**39.0**	**45.0**

Table 5: Results for SWEAR and the best SWEAR-SS initialization (VRAE) trained on 100- and 200- per-property subsets, respectively.

Model	Mean F1
SWEAR-PR	**66.5**
dropout on input only	65.4
no dropout	64.6
shared reviewer cells	63.8
SWEAR-MLR	63.0
w/o skip connections	60.0

Table 6: Results for semi-supervised reviewer models trained on the 1% subset of WikiReading.

Table 4 and 5 show the results of SWEAR and semi-supervised models with pretrained and fixed embeddings. Results show that SWEAR-SS always improves over SWEAR at small data sizes, with the difference become dramatic as the dataset becomes very small. VRAE pretraining yields the best performance. As training and testing datasets have different distributions in per-property subsets, Mean F1 for supervised and semi-supervised models drops compared to uniform sampling. However, initialization with pretrained VRAE model leads to a substantial improvement on both subsamples. We further experimented by initializing the decoder (vs. only the encoder) with pretrained autoencoder weights but this resulted in a lower Mean F1.

Table 6 shows the results of semi-supervised reviewer models. When trained on 1% of the training data, SWEAR-MLR and the supervised SWEAR model perform similarly. Without using skip connections between embedding and hidden layers, the performance drops. The SWEAR-PR model further improves Mean F1 and outperforms the strongest SWEAR-SS model, even without fine-tuning the weights initialized from the autoencoder.

The success of SWEAR-PR rests on multiple design elements working together, as shown by the reduced performance caused by altering or disabling them. Using dropout only on the inputs, or not using any dropout on reviewer cells, causes a substantial decrease in Mean F1 score (by 1.1 and 1.9, respectively). Configuring the model with many more review steps (15) but with

Parameter	Space
All Models	
Learning Rate	[0.0001; 0.005]
Learning Rate Decay Steps	{25k, 50k}
Gradient Clip	[0.01; 1.0]
Supervised & Semi-Supervised	
Embedding Size	{128, **256**, 512}
RNN State Size	{256, **512**}
Window Size	{10, 20, **30**}
Batch Size	{64, **128**}
Dropout	{0.3, **0.5**, 0.8}
Autoencoders Only	
Embedding Sharing	{Input, **All**}
KL-weight	[0.0001;0.01]
Semi-Supervised Only	
Finetune Pretrained Parameters	{YES, NO}
Dropout	{0.7, 0.8, **0.9**}
Reviewer State Size	{256, **512**}
Question Reviewer Steps	{0, 2, 3}
Window Reviewer Steps	{2, 3, 5, 8}

Table 7: Hyperparameter search spaces for each model type. We use {...} to denote a set of discrete values and [...] to denote a continuous range. Following Hewlett et al. (2016), we ran a random search over the possible configurations.

a smaller hidden vector size (128) reduced Mean F1 to 62.5. Increasing the number of review steps for the question to 5 caused a decrease in Mean F1 of 2.1.

6 Related Work

Our model architecture is one of many hierarchical models for documents proposed in the literature. The most similar is proposed by Choi et al. (2017), which uses a coarse-to-fine approach of first encoding each sentence with a cheap BoW or Conv model, then selecting the top k sentences to form a mini-document which is then processed by a standard seq2seq model. While they also evaluate their approach on WikiReading, their emphasis is on efficiency rather than model accuracy, with the resulting model performing slightly worse than the full seq2seq model but taking much less time to execute. SWEAR also requires fewer sequential steps than the document length but still computes at least as many recurrent steps in parallel.

Our model can also be viewed as containing a Memory Network (MemNet) built from a document (Weston et al., 2014; Sukhbaatar et al., 2015), where the memories are the window encodings. The core MemNet operation consists of attention over a set of vectors (memories) based on a query encoding, and then reduction of a second set of vectors by weighted sum based on the attention weights. In particular, Miller et al. (2016) introduce the Key-Value MemNet where the two sets of memories are computed from the keys and values of a map, respectively: In their QA task, each memory entry consists of a potential answer (the value) and its context bag of words (the key).

Our reviewer approach is inspired by "Encode, Review, Decode" approach introduced by Yang et al. (2016), which showed the value of introducing additional computation steps between the encoder and decoder in a seq2seq model.

The basic recurrent autoencoder was first introduced by Dai et al. (2015), a standard seq2seq model with the same input and output. Fabius et al. (2014) expanded this model into the Variational Recurrent Autoencoder (VRAE), which we describe in Section 4.1.1. VRAE is an application of the general idea of variational autoencoding, which applies variational approximation to the posterior to reconstruct the input (Kingma and Welling, 2013). While we train window autoencoders, an alternative approach is hierarchical document autoencoders (Li et al., 2015).

The semi-supervised approach of initializing the weights of an RNN encoder with those of a recurrent autoencoder was first studied by Dai et al. (2015) in the context of document classification and further studied by Ramachandran et al. (2016) for traditional sequence-to-sequence tasks such as machine translation. Our baseline semi-supervised model can be viewed as an extension of these approaches to a reading comprehension setting. Dai et al. (2015) also explore initialization from a language model, but find that the recurrent autoencoder is superior, which is why we do not consider language models in this work.

7 Conclusions

We have demonstrated the efficacy of the SWEAR architecture, reaching state of the art performance on supervised WikiReading. The model improves the extraction of precise information from long documents over the baseline seq2seq model. In a semi-supervised setting, our method of reusing (V)RAE encodings in a reading comprehension framework is effective, with SWEAR-PR reaching an accuracy of 66.5 on 1% of the dataset against last year's state of the art of 71.8 using the full dataset. However, these methods require careful configuration and tuning to succeed, and making them more robust presents an excellent opportunity for future work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Eunsol Choi, Daniel Hewlett, Alexandre Lacoste, Illia Polosukhin, Jakob Uszkoreit, and Jonathan Berant. 2017. Coarse-to-fine question answering for long document. In *Association for Computational Linguistics (ACL)*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc.

Otto Fabius and Joost R van Amersfoort. 2014. Variational recurrent auto-encoders. *arxiv*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *Association for Computational Linguistics (ACL)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arxiv*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Association for Computational Linguistics (ACL)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arxiv*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. Unsupervised pretraining for sequence to sequence learning. *CoRR*, abs/1611.02683.

A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. 2016. Progressive Neural Networks. *arxiv*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Zhilin Yang, Ye Yuan, Yuexin Wu, Ruslan Salakhutdinov, and William W. Cohen. 2016. Encode, review, and decode: Reviewer module for caption generation. *arxiv*.