Visualizing and Understanding Recurrent Networks

mingzailao

2016-10-11

Outline

RECURRENT NEURAL NETWORK MODELS

Basic Setting

- $h_t^0 = x_t$
- $\{h_t^L\}$ is used to predict an output vector y_t
- all $h \in \mathbb{R}^n$

Vanilla Recurrent Neural Network (RNN)

$$h_t^l = tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

RECURRENT NEURAL NETWORK MODELS

Long Short-Term Memory (LSTM)

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$\begin{aligned} c_t^l &= f \odot c_{t-1}^l + i \odot g \\ h_t^l &= o \odot tanh(c_t^l) \end{aligned}$$

RECURRENT NEURAL NETWORK MODELS

Gated Recurrent Unit (GRU)

$$\begin{pmatrix} r \\ c \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \end{pmatrix} W_r^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$\begin{aligned} \tilde{h}_t^l &= tanh(W_x^l h_t^{l-1} + W_g^l (r \odot h_{t-1}^l)) \\ h_t^l &= (1 - z) \odot h_{t-1}^l + z \odot \tilde{h}_t^l \end{aligned}$$

CHARACTER-LEVEL LANGUAGE MODELING

SETUP

- Input : a sequence of characters;
- Output : the next character for each character in the sequence;
- Between $h_t^L \rightarrow y$: Softmax Classifier;
- Encoding : assuming a fixed vocabulary of K characters, we encode all characters with K-dimensional 1-of-K vectors(OneHot);
- Loss : cross-entropy loss;

CHARACTER-LEVEL LANGUAGE MODELING

OPTIMIZATION

- Initialization : uniformly in range $[-0.08, 0.08]$;
- Batch size : 100;
- Optimizier : RMSProp;
- Timestep : 100;
- Epochs : 50;
- Learning rate decay : after 10 epochs with decay rate 0.95 after each additional epoch;
- Learning rate : 2×10^{-3}

EXPERIMENTS DATASETS AND PROCESSING

Datasets

- War and Peace (WP);
- Linux Kernel (LK);

Split the dataset

- 80/10/10 for WP;
- 90/5/5 for LK;

COMPARING RECURRENT NETWORKS

	LSTM			RNN			GRU		
Layers	1	2	3	1	2	3	1	2	3
Size	War and Peace Dataset								
64	1.449	1.442	1.540	1.446	1.401	1.396	1.398	**1.373**	1.472
128	1.277	1.227	1.279	1.417	1.286	1.277	1.230	**1.226**	1.253
256	1.189	**1.137**	1.141	1.342	1.256	1.239	1.198	1.164	1.138
512	1.161	1.092	1.082	-	-	-	1.170	1.201	**1.077**
	Linux Kernel Dataset								
64	1.355	**1.331**	1.366	1.407	1.371	1.383	1.335	1.298	1.357
128	1.149	1.128	1.177	1.241	**1.120**	1.220	1.154	1.125	1.150
256	1.026	**0.972**	0.998	1.171	1.116	1.116	1.039	0.991	1.026
512	0.952	0.840	0.846	-	-	-	0.943	0.861	**0.829**

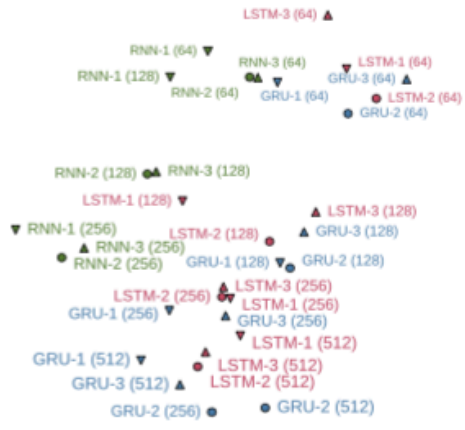


Figure 1: **Left:** The **test set cross-entropy loss** for all models and datasets (low is good). Models in each row have nearly equal number of parameters. The test set has 300,000 characters. The standard deviation, estimated with 100 bootstrap samples, is less than 4×10^{-3} in all cases. **Right:** A t-SNE embedding based on the probabilities assigned to test set characters by each model on War and Peace. The color, size, and marker correspond to model type, model size, and number of layers.

COMPARING RECURRENT NETWORKS

Analysis

- Depth of at least two is beneficial;
- The results are mixed between the LSTM and the GRU, but both significantly outperform the RNN;

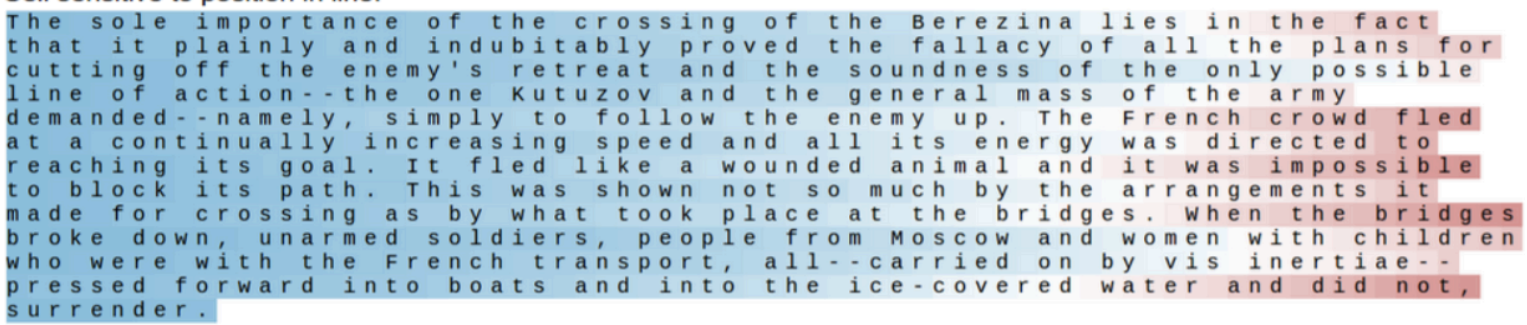
INTERNAL MECHANISMS OF AN LSTM

Interpretable, long-range LSTM cells

- In this experiment we verify that multiple interpretable cells do in fact exist in these networks.
- For instance, one cell is clearly acting as a line length counter, starting with a high value and then slowly decaying with each character until the next newline.

INTERNAL MECHANISMS OF AN LSTM

Cell sensitive to position in line:



INTERNAL MECHANISMS OF AN LSTM

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

INTERNAL MECHANISMS OF AN LSTM

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

INTERNAL MECHANISMS OF AN LSTM

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

INTERNAL MECHANISMS OF AN LSTM

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
           struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
      df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

INTERNAL MECHANISMS OF AN LSTM

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

INTERNAL MECHANISMS OF AN LSTM

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

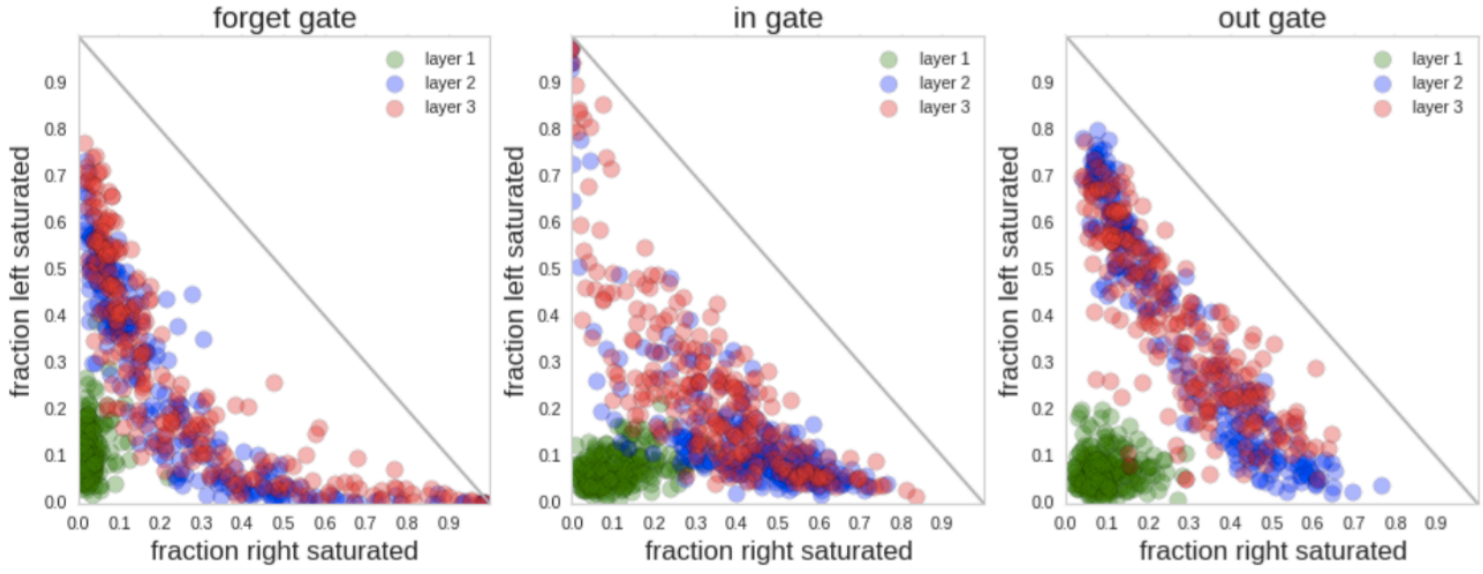
INTERNAL MECHANISMS OF AN LSTM

Gate activation statistics

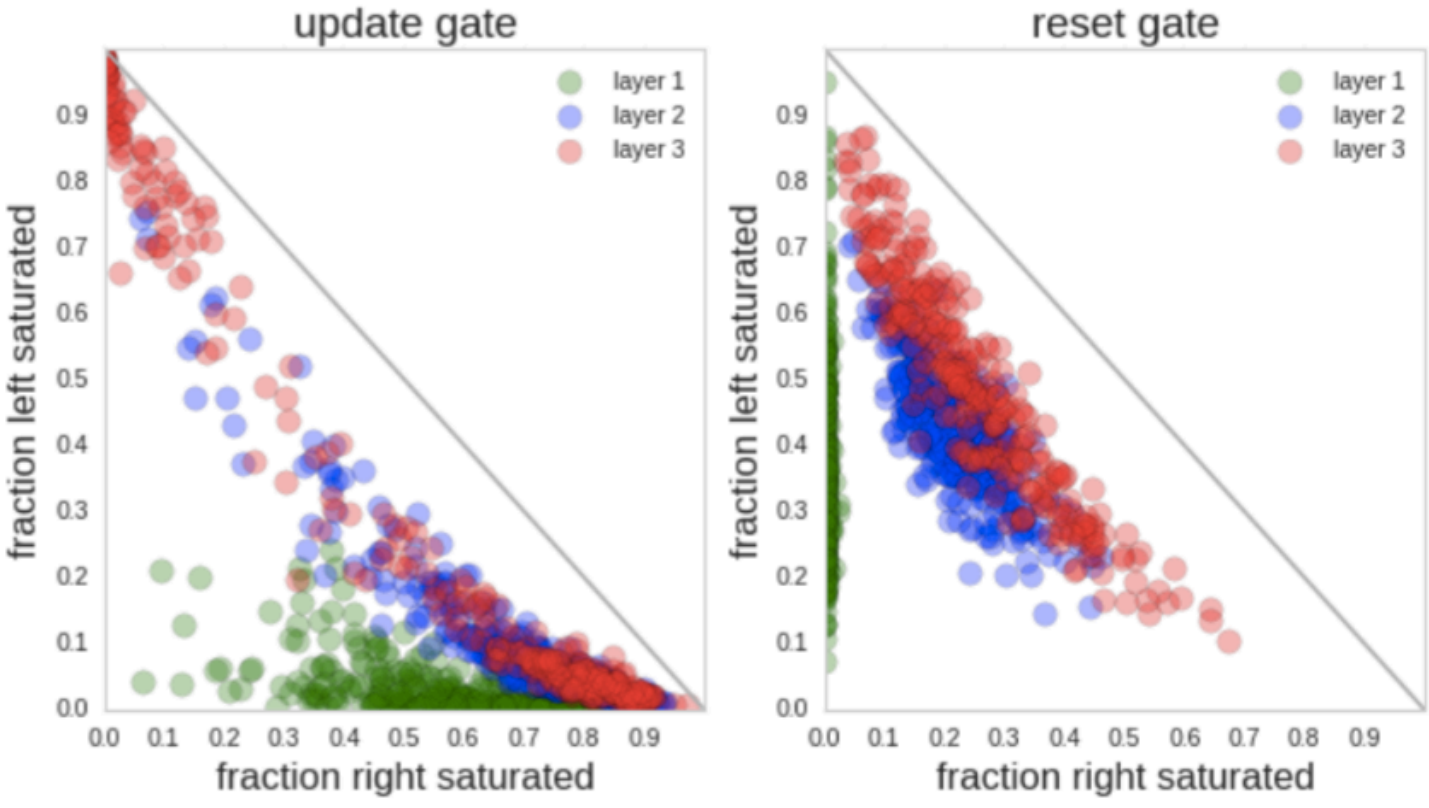
We were particularly interested in looking at the distributions of saturation regimes in the networks, where we define a gate to be left or right-saturated if its activation is less than 0.1 or more than 0.9, respectively, or unsaturated otherwise. We then compute the fraction of times that each LSTM gate spends left or right saturated.

INTERNAL MECHANISMS OF AN LSTM

LSTM



INTERNAL MECHANISMS OF AN LSTM



GRU

INTERNAL MECHANISMS OF AN LSTM

Analysis

- For instance, the number of often right-saturated forget gates is particularly interesting, since this corresponds to cells that remember their values for very long time periods.
- The output gate statistics also reveal that there are no cells that get consistently revealed or blocked to the hidden state.

INTERNAL MECHANISMS OF AN LSTM

Surprising finding(UNKNOWN)

unlike the other two layers that contain gates with nearly binary regime of operation (frequently either left or right saturated), the activations in the first layer are much more diffuse(near the origin in our scatter plots).

UNDERSTANDING LONG-RANGE INTERACTIONS

SETUP

Good performance of LSTMs is frequently attributed to their ability to store long-range information. In this section we test this hypothesis by comparing an LSTM with baseline models that can only utilize information from a fixed number of previous steps.

UNDERSTANDING LONG-RANGE INTERACTIONS

BASELINES(n-NN)

A fully-connected neural network with one hidden layer and tanh nonlinearities.

- Input : a sparse binary vector of dimension nK that concatenates the one-of-K encodings of n consecutive characters.

BASELINES(n-gram)

An unpruned (n + 1)-gram language model using modified Kneser-Ney smoothing

UNDERSTANDING LONG-RANGE INTERACTIONS

Model \ n	1	2	3	4	5	6	7	8	9	20
War and Peace Dataset										
n-gram	2.399	1.928	1.521	1.314	1.232	1.203	**1.194**	1.194	1.194	1.195
n-NN	2.399	1.931	1.553	1.451	1.339	**1.321**	-	-	-	-
Linux Kernel Dataset										
n-gram	2.702	1.954	1.440	1.213	1.097	1.027	0.982	0.953	0.933	**0.889**
n-NN	2.707	1.974	1.505	1.395	**1.256**	1.376	-	-	-	-

Table 2: The **test set cross-entropy loss** on both datasets for n-gram models (low is good). The standard deviation estimate using 100 bootstrap samples is below 4×10^{-3} in all cases.

UNDERSTANDING LONG-RANGE INTERACTIONS

Analysis

- The n-gram and n-NN models perform nearly identically for small values of n, but for larger values the n-NN models start to overfit and the n-gram model performs better.
- On both datasets our best recurrent network outperforms the 20-gram model (1.077 vs. 1.195 on WP and 0.84 vs.0.889 on LK).
- The 20-gram model file has 3GB, while our largest recurrent based model is about 11MB.

UNDERSTANDING LONG-RANGE INTERACTIONS

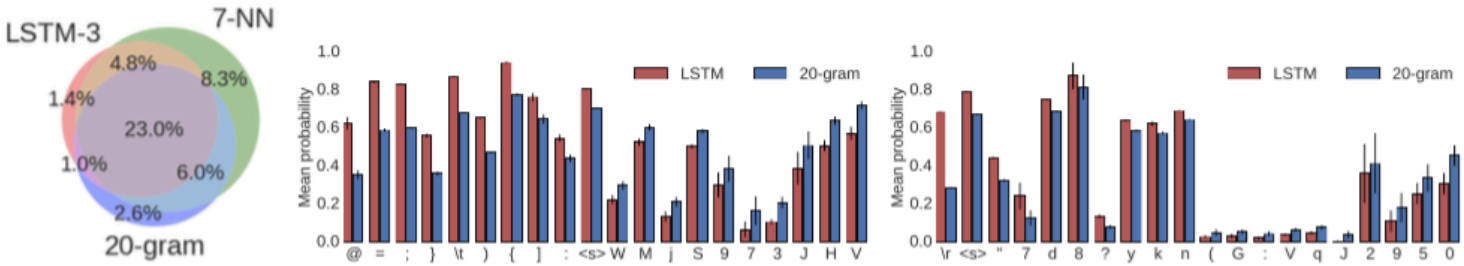


Figure 4: **Left:** Overlap between test-set errors between our best 3-layer LSTM and the n-gram models (low area is good). **Middle/Right:** Mean probabilities assigned to a correct character (higher is better), broken down by the character, and then sorted by the difference between two models. "<s>" is the space character. LSTM (red) outperforms the 20-gram model (blue) on special characters that require long-range reasoning. Middle: LK dataset, Right: WP dataset.

UNDERSTANDING LONG-RANGE INTERACTIONS

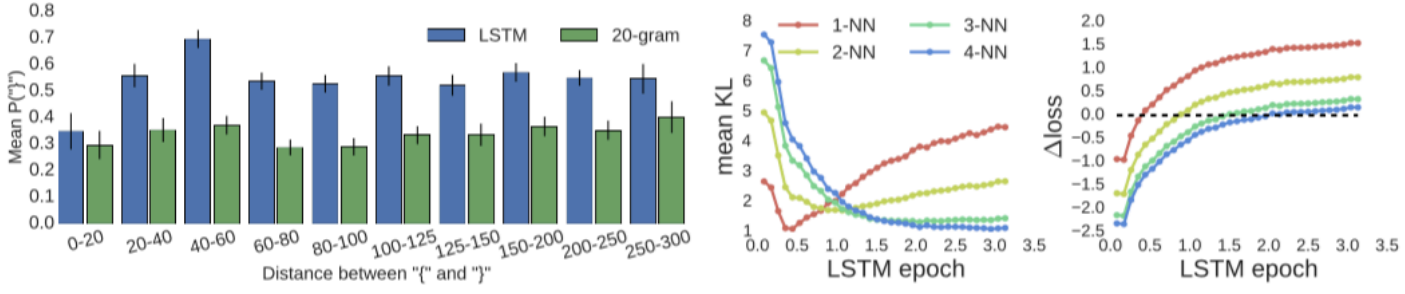


Figure 5: **Left**: Mean probabilities that the LSTM and 20-gram model assign to the "}" character, bucketed by the distance to the matching "{". **Right**: Comparison of the similarity between 3-layer LSTM and the n-NN baselines over the first 3 epochs of training, as measured by the symmetric KL-divergence (middle) and the test set loss (right). Low KL indicates similar predictions, and positive Δloss indicates that the LSTM outperforms the baseline.