Представление исправлений

Ольга Лупуляк

Вступление

Существует очень много данных (github, правки в со всевозможных текстовых документах и т.д.)

Хочется как-то обработать эти данные, для того, чтобы:

- Сгруппировать семантически похожие правки, выявить наиболее часто встречающиеся, на основе этой информации сделать какие-то выводы
- Научиться автоматически производить известную правку в другом контексте (один и тот же документ на разных языках)

Представление исправлений

$$f \triangle (x_-, x_+) \in \mathbb{R}^n$$

Мы хотим по исходному и отредактированному элементам получать векторное представление произведённой правки.

Похожие по смыслу правки должны иметь близкое представление.

Область значений должна быть ограничена.

Нейронный редактор

Конструирует по исходному элементу и правке исправленную версию.

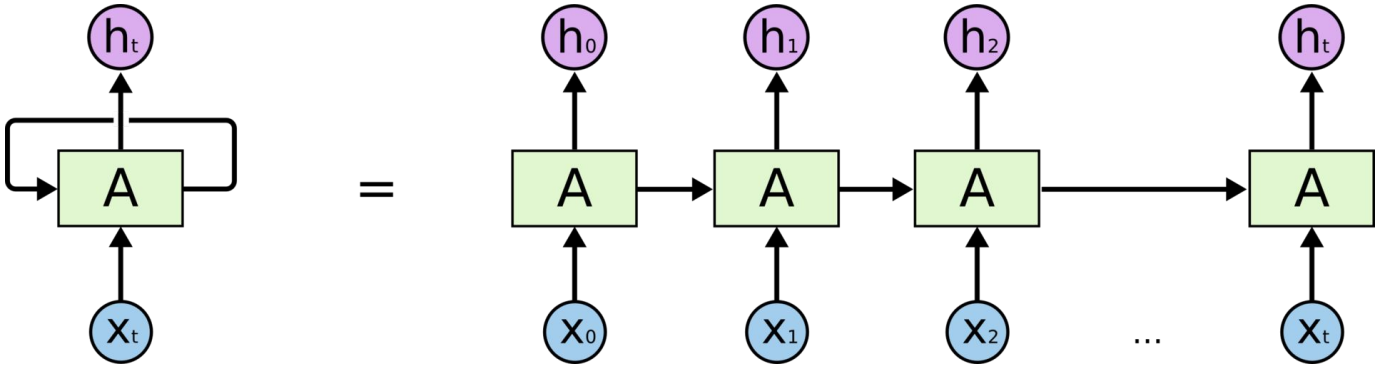
$$\alpha(x'_-, f\triangle(x_-, x_+)) = x'_+$$

Помогает применять существующие правки в новом контексте

Немного теории: рекуррентные нейронные сети

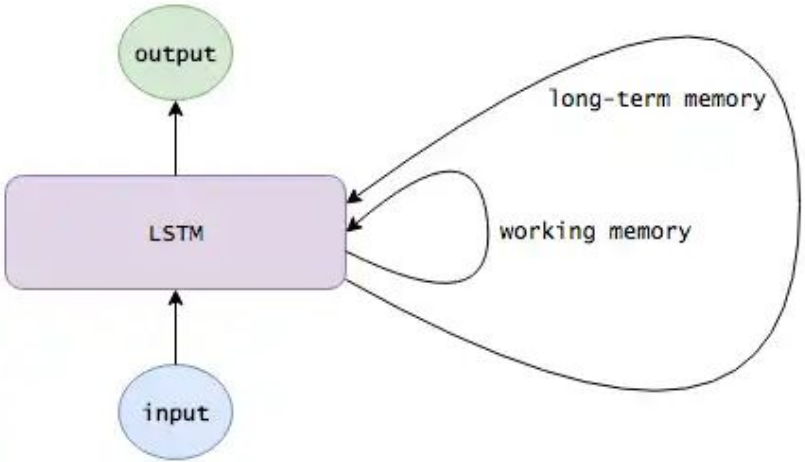
Помогают при ответе на вопрос учитывать предыдущий контекст.

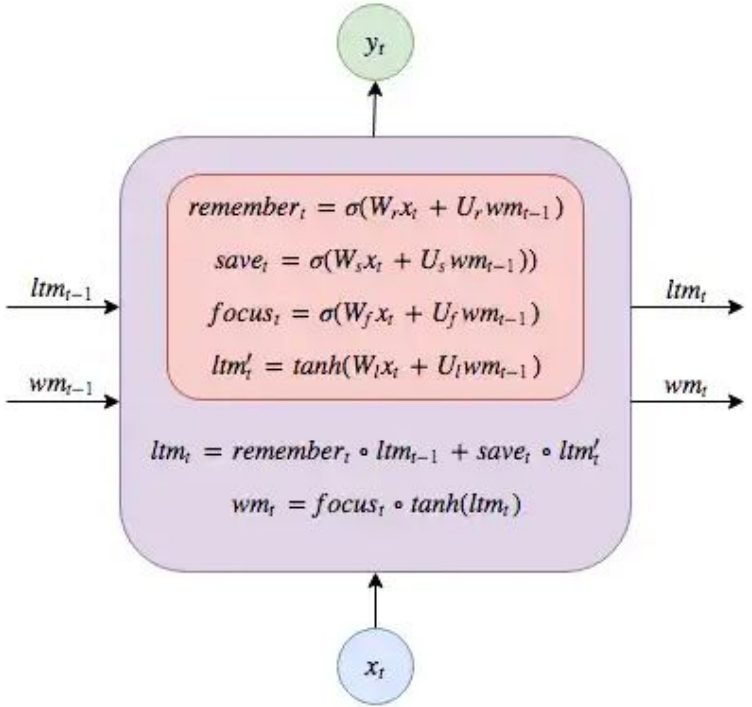
Проблема: долговременная зависимость



Немного теории: LSTM

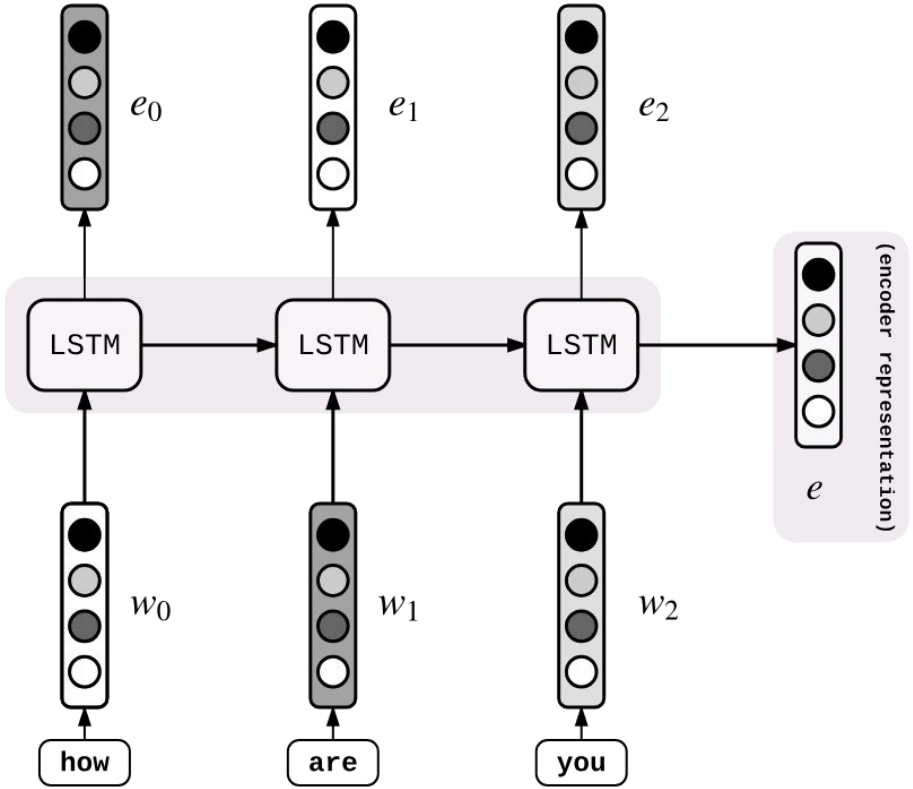
- Remember gate (forget gate) -- позволяет забывать ненужную информацию
- Save gate (input gate) -- определяет, что из поступившей информации следует добавить в долговременную память
- Focus gate (output gate) -- определяет, что из долговременной памяти пригодится в ближайшее время


output
long-term memory
LSTM
working memory
input


y_t
$remember_t = \sigma(W_r x_t + U_r wm_{t-1})$
$save_t = \sigma(W_s x_t + U_s wm_{t-1}))$
$focus_t = \sigma(W_f x_t + U_f wm_{t-1})$
$ltm'_t = tanh(W_l x_t + U_l wm_{t-1})$
ltm_{t-1}
ltm_t
wm_{t-1}
wm_t
$ltm_t = remember_t \circ ltm_{t-1} + save_t \circ ltm'_t$
$wm_t = focus_t \circ tanh(ltm_t)$
x_t

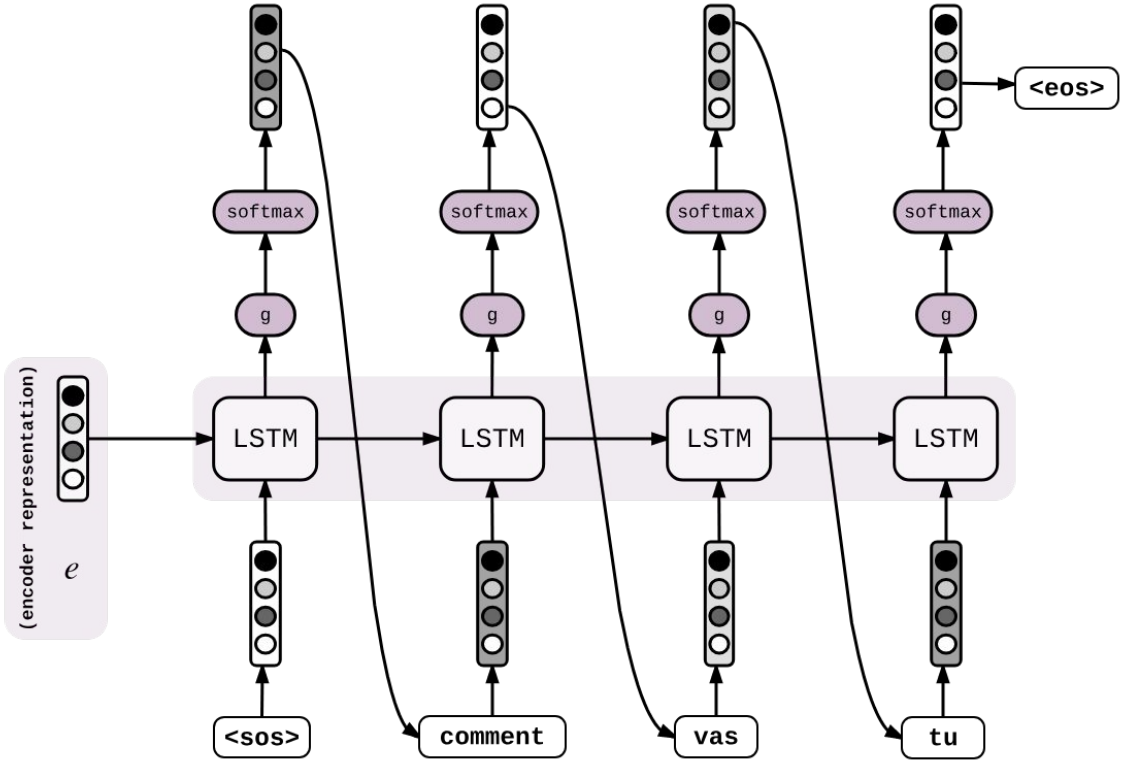
Немного теории: sequence-to-sequence model

Состоит из кодировщика и декодировщика, каждый из которых -- это сеть LSTM.



Немного теории: sequence-to-sequence model

$$g : \mathbb{R}^h \rightarrow \mathbb{R}^V$$
$$h_0 = LSTM(e, w_{sos})$$
$$s_0 = g(h_0)$$
$$p_0 = softmax(s_0)$$
$$i_0 = argmax(p_0)$$
$$h_t = LSTM(h_{t-1}, w_{i_{t-1}})$$
$$s_t = g(h_t)$$
$$p_t = softmax(s_t)$$
$$i_t = argmax(p_t)$$



Sequence-to-sequence model with attention

$$h_t = \text{LSTM}(h_{t-1}, [w_{i_{t-1}}, c_t])$$

$$s_t = g(h_t)$$

$$p_t = \text{softmax}(s_t)$$

$$i_t = \text{argmax}(p_t)$$

$$\alpha_{t'} = f(h_{t-1}, e_{t'}) \in \mathbb{R} \; \forall t'$$

$$\bar{\alpha} = \text{softmax}(\alpha)$$

$$c_t = \sum_{t'=0}^{n} \bar{\alpha}_{t'} e_{t'}$$

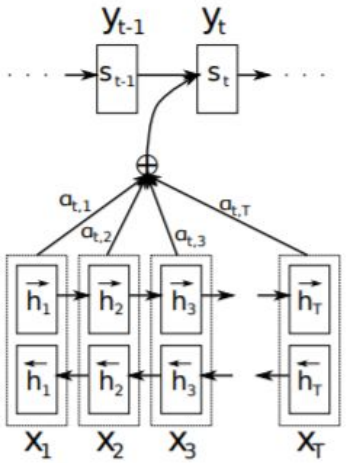


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Graph-to-Tree Neural Editor: graph-based encoder

- Исходный элемент представляется в виде дерева (например, AST)
- Добавляем дополнительные связи (например, следующий токен)
- Кодируем граф в вектор, используя Gated Graph Neural Network
- Это позволяет декодировщику использовать attention mechanisms

Gated Graph Neural Network

$$\mathbf{h}_v^{(1)} = [\boldsymbol{x}_v^\top, \mathbf{0}]^\top \tag{1}$$

$$\mathbf{a}_v^{(t)} = \mathbf{A}_{v:}^\top \left[\mathbf{h}_1^{(t-1)\top} \dots \mathbf{h}_{|\mathcal{V}|}^{(t-1)\top}\right]^\top + \mathbf{b} \tag{2}$$

$$\mathbf{z}_v^t = \sigma\left(\mathbf{W}^z \mathbf{a}_v^{(t)} + \mathbf{U}^z \mathbf{h}_v^{(t-1)}\right) \tag{3}$$

$$\mathbf{r}_v^t = \sigma\left(\mathbf{W}^r \mathbf{a}_v^{(t)} + \mathbf{U}^r \mathbf{h}_v^{(t-1)}\right) \tag{4}$$

$$\widetilde{\mathbf{h}_v^{(t)}} = \tanh\left(\mathbf{W}\mathbf{a}_v^{(t)} + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{(t-1)}\right)\right) \tag{5}$$

$$\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{(t-1)} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^{(t)}}. \tag{6}$$

$$\mathbf{h}_{\mathcal{G}} = \tanh\left(\sum_{v \in \mathcal{V}} \sigma\left(i(\mathbf{h}_v^{(T)}, \boldsymbol{x}_v)\right) \odot \tanh\left(j(\mathbf{h}_v^{(T)}, \boldsymbol{x}_v)\right)\right)$$

Graph-to-Tree Neural Editor: tree-based decoder

- EXPANDR -- расширяет существующий нетерминал, используя грамматическое правило
- GENTERM -- генерирует терминальный токен из словаря или берёт существующий токен из входа
- TREECP -- копирует поддерево из входа

$$p(a_t | a_{<t}, s)$$

Graph-to-Tree Neural Editor: tree-based decoder

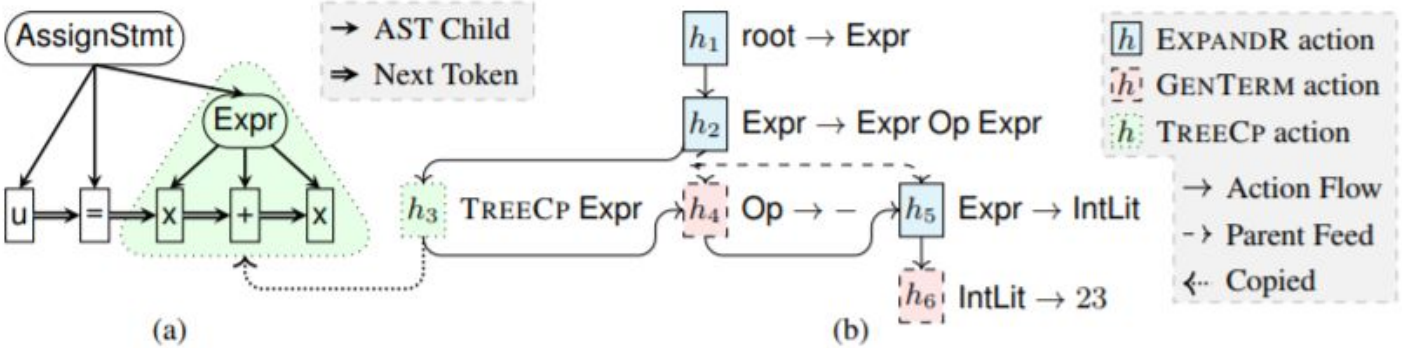


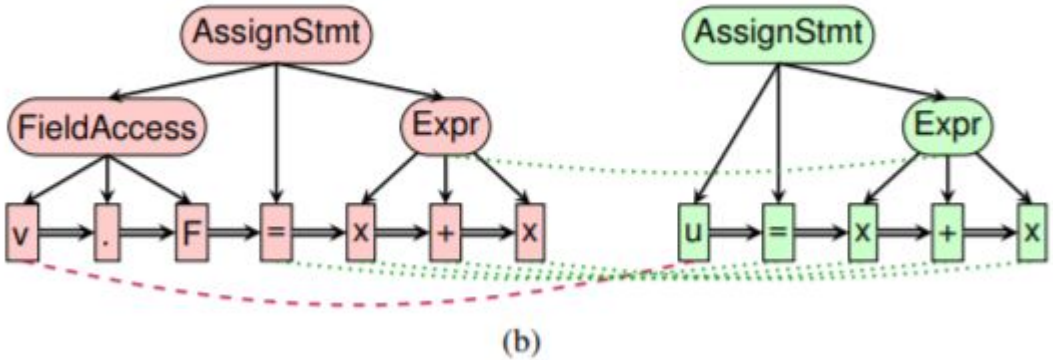
Figure 2: (a) Graph representation of statement `u = x + x`. Rectangular (resp. rounded) nodes denote tokens (resp. non-terminals). (b) Sequence of tree decoding steps yielding `x + x - 23`, where `x + x` is copied (using the TREECP action) from the context graph in (a).

Представление правок: seq2seq

Стандартный diffing algorithm. Используя дополнительные символы и объединяя все три представление, скармливаем результат в обычный sequence encoder.

Представление правок: граф

Соединяем два дерева, добавляя рёбра “added”, “removed” и “replaced”.



Datasets

- WikiAtomicEdits -- исправления в Википедии; 1040K правок
- 54 проекта с Github’a на C#; 111724 правок размер не более 3х строк
- C# fixers -- библиотека распространённых рефакторингов для C#, было выбрано 6 штук и сгенерировано 2878 примеров с известной семантикой

Результаты: представление

- Source code:
  C# fixers; 100 примеров правок
  на каждый fixer (есть визуализация)

- Естественный язык:
  200 рандомных примеров, оценивалось людьми

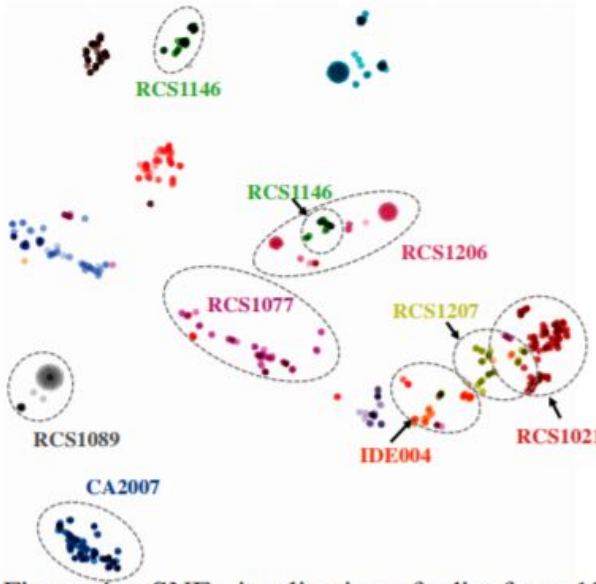


Figure 4: t-SNE visualization of edits from 13 C# fixers, where point color indicates the fixer. Labels indicate the id of the fixer, see main text.

Precision

Table 4: Test Performance of Different Neural Editors.

Model	Acc.@1 (%)	Recall@5 (%)	PPL per token
GitHubEdits			
Seq2Seq – Seq Edit Encoder	**59.63**	**65.46**	**1.2792**
Graph2Tree – Seq Edit Encoder	57.49	62.94	1.3043
Graph2Tree – Graph Edit Encoder	48.05	56.51	1.3712
WikiAtomicEdits			
Seq2Seq – Seq Edit Encoder	72.94	76.53	1.0527

Table 5: Transfer learning results on C# fixers data, averaged across all fixer categories.

Model	Acc.(%)	Acc.*(%)	Recall@5(%)	Recall@5*(%)
Seq2Seq – Seq Edit Encoder	38.35	**77.67**	41.50	**83.84**
Graph2Tree – Seq Edit Encoder	**49.21**	77.30	**51.93**	81.77

*: upper-bound performance of predicting $\boldsymbol{x}_+$ using the gold-standard edit representations.

Вопросы?

Ссылка на статью: https://arxiv.org/pdf/1810.13337.pdf