Основы обработки текстов

Лекция #4:
Символьные представления слов

Лектор: м.н.с. ИСП РАН Андрианов Иван Алексеевич

План лекции

- Важность анализа символьного состава слова на примере задачи NER
- ngramы по символам
- Byte-pair encoding
- Рекуррентные сети
- Классические сверточные сети
- «Расширяющиеся» сверточные сети
- Модель fasttext

Распознавание именованных сущностей (NER)

- На входе: текст, разбитый на предложения и токены
- На выходе: множество сущностей (начало, конец, тип)

Александр Пушкин родился в Москве, столице России

личность город страна

Microsoft — один из крупнейших производителей ПО в мире

компания

Обработка текстов — область на стыке ИИ и лингвистики

научная дисциплина НД НД

Общая схема решения NER

x Александр Пушкин погиб в результате дуэли с Дантесом

ŷ I-лич I-лич O O O O O I-лич

- Нейронная сеть
- v_i — векторное представление (skip-gram) i-ого слова
- $h_i = \text{BiLSTM}(v_1, v_2, \ldots, v_T, i)$ кодирование контекста
- $p(y|x) = e^{\text{score}(y|x)} / \Sigma_{y'} e^{\text{score}(y'|x)}$ целевая функция – CRF

  $\text{score}(y|x) = \Sigma_i ((Wh_i + b)[y_i] + p(y_i|y_{i-1}))$

Символьный состав слова в задаче NER

Кикабидзе, Мкртчян и Леонов сыграли в фильме «Мимино»

Вдоль Дона расположены Ростов-на-Дону и Калач-на-Дону

Природный газ состоит из метана, этана, пропана, бутана

- В небольших тренировочных выборках не могут содержаться все релевантные наименования
- Векторные представления слов содержат бОльшие словари из-за бОльших выборок, однако:
  - словари все равно неполны
  - наименования распределены по степенному закону

Ngramы по символам

- Простейшее представление слова по символам — «мешок» символьных Ngram

Пушкин

N=3 => {<w>Пу, Пуш, ушк, шки, кин, ин</w>}

N≥8 => {<w>Пушкин</w>}

- Перенумеровываем все ngramы в обучающей выборке
- Строим «мешок» и затем используем k-hot кодирование: вектор, где стоят 1 на позициях, соответствующих ngramам текущего слова, и 0 на остальных

Byte-pair encoding

- В языке есть устойчивые сочетания: «дзе», «ая», «ина»
- «Мешок» не учитывает расположение, а увеличение размера ngram приводит к появлению редких «шумных»
- Можно расширить словарь символов за счет сочетаний
- Вводим D — максимальный размер словаря
- Ищем самую частую пару символов в выборке, считаем ее самостоятельным символом и добавляем в словарь
- Повторяем процесс, пока не исчерпано D

М, ал, ая; род, ина; К, и, к, а, б, и, дзе.

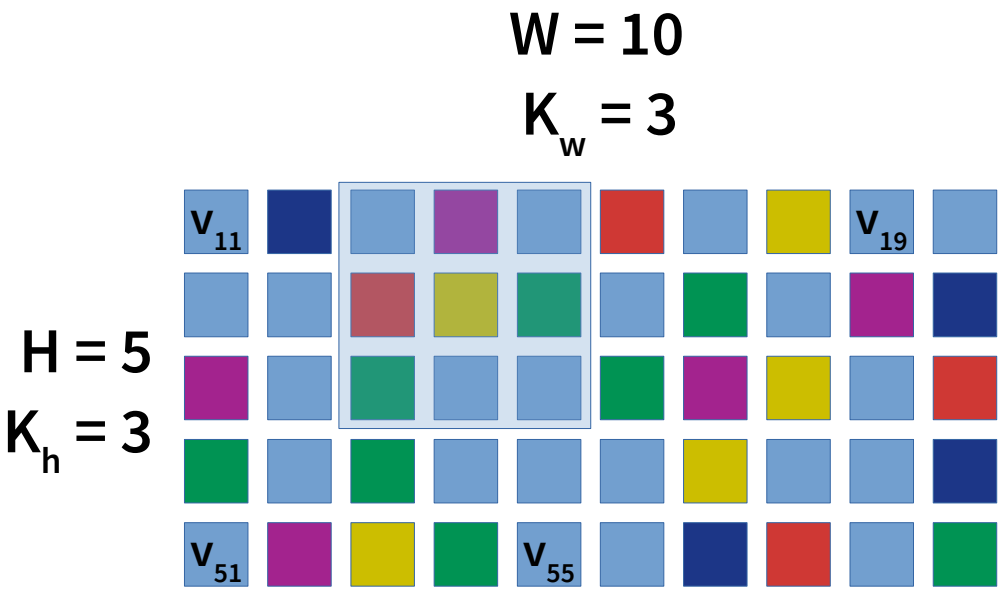
Рекуррентные сети

- Символы слова — это последовательность, как и слова предложения => можно применить рекуррентные сети
- $q_{i,j}$ — вектора символов: обучаются вместе с остальной сетью; определяют схожесть символов с точки зрения целевой задачи (NER)
- $f_{i,j} = LSTM(q_{i,1}, q_{i,2}, \ldots, q_{i,R}, j)$; $g_{i,j} = LSTM(q_{i,R}, \ldots, q_{i,2}, q_{i,1}, j)$
- $f_{i,R}$ и $g_{i,1}$ соединяются с v_i и в следующих слоях выступают в роли составного векторного представления слова x_i/c_i

Классические сверточные сети

- Рекуррентные сети обрабатывают вход последовательно: длиннее последовательности => медленнее обработка
- В анализе изображений используются сверточные сети: позволяют обрабатывать вход параллельно
- Каждый пиксель (i,j) изображения представим, например, 3х-мерным RGB-вектором $v_{i,j}$
- Пройдем по изображению скользящим окном размера (K_h, K_w) (обычно 2-4) с шагом (S_h, S_w) (обычно 1-2)
- Для каждого окна получим $K_h \cdot K_w \cdot |v_{i,j}|$ признаков

Классические сверточные сети



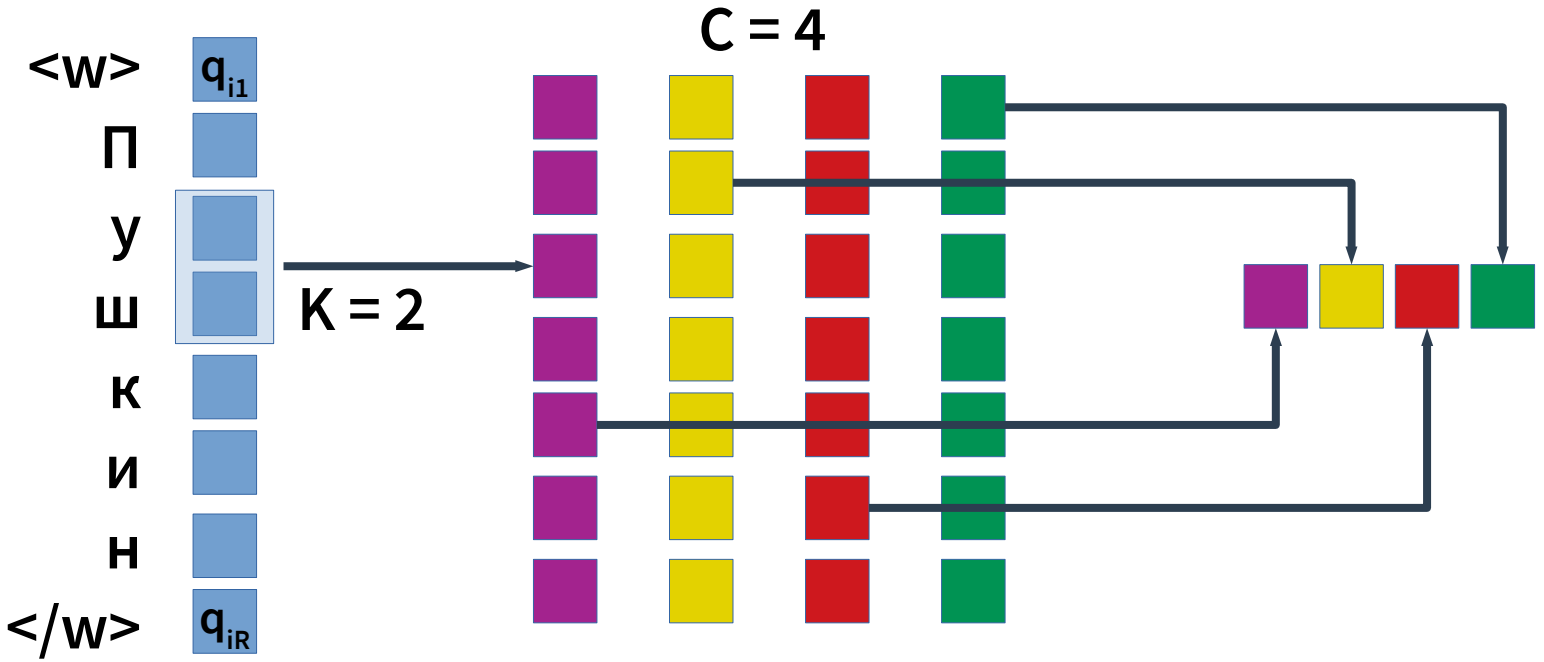
Классические сверточные сети

- К этим признакам применим линейное преобразование с матрицей W_c (фильтром) и получим признак для всего окна
- Можно применить C таких преобразований с разными фильтрами и получить C-мерный вектор признаков окна
- Вместо 3-мерных (RGB) признаков для изображения размера (H, W) получим C-мерные признаки для изображения размера $((H - K_h + 1) / S_h, (W - K_w + 1) / S_w)$
- Важно: линейные преобразования с каждым фильтром для каждого окна можно выполнять независимо и параллельно (для таких задач идеально подходят GPU)

Классические сверточные сети

- В нашем случае изображение — это вектора символов слова $q_{i,j}$, $H = R$, $K_h = K$, $S_h = S$, $W = K_w = S_w = 1$
- При $S = 1$ — аналог символьных K-gram, однако ngramы рассматриваются не как независимые атомарные единицы, а как конкатенация $q_{i,j}$ для содержащихся в них символов => выше обобщающая способность модели
- Чтобы получить векторное представление слова по его символам, применим max-pooling: выберем максимальное значение среди всех окон/ngram по каждому из фильтров

Классические сверточные сети



«Расширяющиеся» сверточные сети

- Чем длиннее последовательность, тем операция max-pooling менее эффективна
- Классическая свертка сокращает последовательность с R до (R — K + 1) / S элементов
- S > 1 приводит к пропускам и потере информации
- При S = 1 и K << R понадобится слишком много уровней свертки, чтобы сократить последовательность хотя бы вдвое
- Слишком большой K приводит к слишком большому числу параметров в фильтрах

«Расширяющиеся» сверточные сети

- Обобщение классической свертки: l — коэффициент расширения, при l = 1 получаем классическую свертку
- Фильтр остается неизменным: его матрица по-прежнему содержит $K \cdot |q_{i,j}|$ параметров
- Применение фильтра производится не к последовательным символам

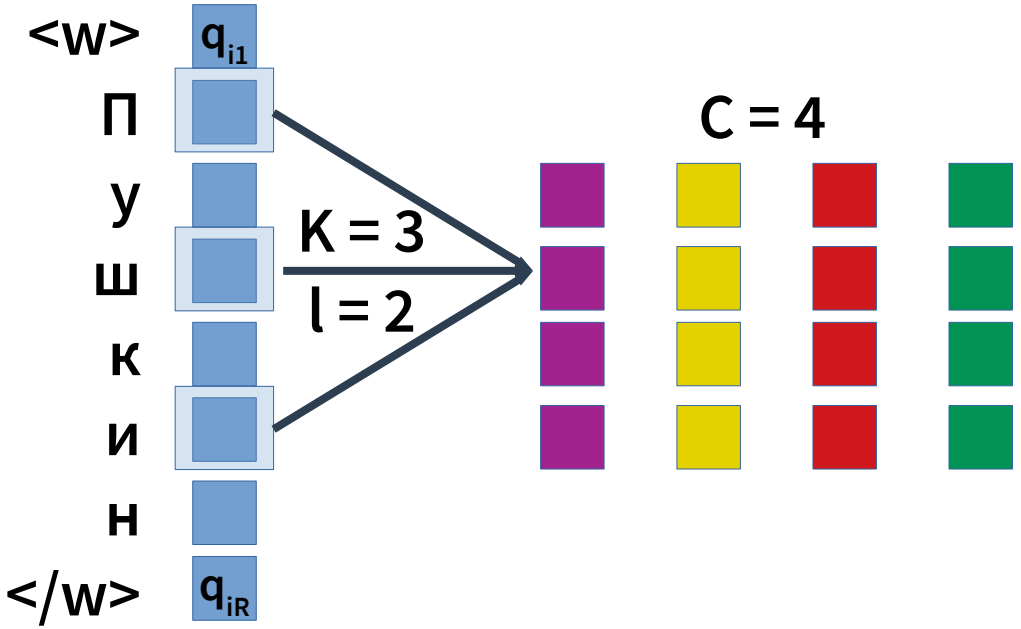
  $(q_{i,j-(K-1)/2}, \dots, q_{i,j-1}, q_{i,j}, q_{i,j+1}, \dots, q_{i,j+(K-1)/2})$
- а к каждому l-ому символу

  $(q_{i,j-l(K-1)/2}, \dots, q_{i,j-l}, q_{i,j}, q_{i,j+l}, \dots, q_{i,j+l(K-1)/2})$

«Расширяющиеся» сверточные сети

- В классической свертке с S > 1 берется каждое S-ое окно
- В «расширяющейся» свертке берутся все окна, но их ширина увеличивается, хотя кол-во параметров остается неизменным (за счет пропусков элементов в окне)
- Применяя такую свертку многократно, можно сокращать размер последовательности экспоненциально, практически не теряя информации

«Расширяющиеся» сверточные сети



Модель fasttext

- Модель skip-gram оптимизирует близость между вектором v_c центрального слова и вектором u_o слова из контекста
- Проблемы такого подхода:
  - Нет векторов у слов, которые не встречались в выборке
  - Для более редких слов вектора «хуже», чем для более частых
- В языках с богатой морфологией:
  - У каждого слова множество форм, встречающихся с разной (в т.ч. очень низкой) частотой, что усугубляет указанные проблемы
  - Указанные формы имеют практически идентичный смысл => хотелось бы иметь близкие вектора

Модель fasttext

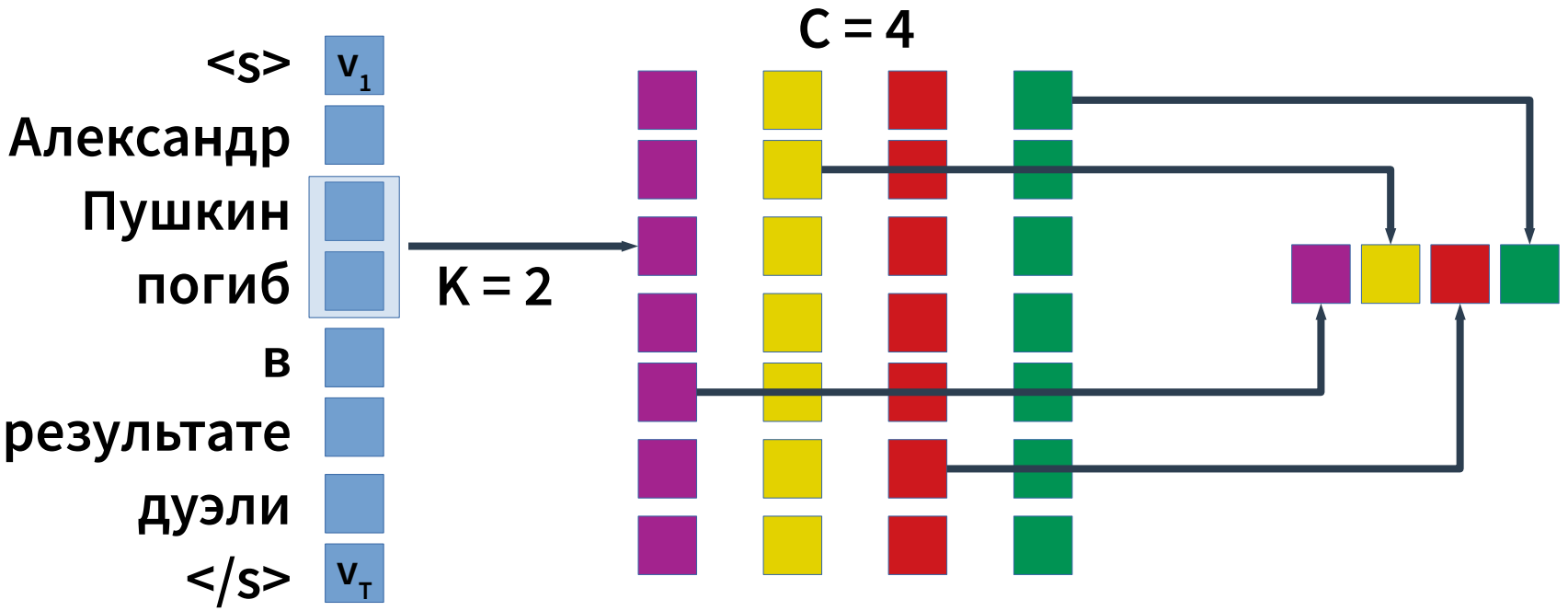
- Идея: представлять центральное слово как сумму вектора словоформы и векторов символьных ngram

  $v_c = w_c + \sum_{g \in \Gamma(c)} w_g$

- Вектора слов, которые не встречались в выборке, строятся исключительно по их символьным ngramam
- Вектора редких форм становятся более «богатыми» за счет разделения общих ngram-слагаемых с частыми формами
- fasttext эффективнее skip-gram в языках с богатой морфологией, «шумных» областях (соц. сети), на задачах морфологического и синтаксического анализа

Бонус. Классификация текстов

- Многие прикладные задачи сводятся к классификации текста: выявление спама, анализ эмоциональной окраски, рубрикация; можно применять те же модели!



Следующая лекция

- Базовые задачи обработки текстов
- Лектор: Майоров Владимир Дмитриевич