Обработка текста на естественном языке (NLP) Лекция 3: От Word2Vec/fastText к RNN, Attention и Transformers (в контексте BERT/ruBERT)

Преподаватель: *Четвергов Андрей Сергеевич*

РАНХиГС — ИЭМИТ
Кафедра эконометрики и математической экономики

Весенний семестр 2026

План лекции

1. Линия развития: от статических векторов к контексту
2. Word2Vec и fastText: как учатся статические вектора
3. RNN и LSTM: от статических векторов к последовательностям
4. Attention, Transformer и маски: контекст «напрямую»
5. BERT: предобучение, дообучение и «эмбеддинг текста»
6. Материалы
7. Итоги

Хронология: внимание было раньше Transformer

Что менялось и почему (коротко и по делу)

2013 **Word2Vec**: плотные вектора слов из локальных контекстов.

2014 **seq2seq RNN**: перевод и последовательности требуют порядка и памяти.

2014 **Attention**: длинные фразы ломают seq2seq без «взгляда на вход».

2016–2017 **fastText**: подслова помогают с морфологией и OOV.

2017 **Transformer**: self-attention становится ядром и даёт параллелизацию.

2018 **BERT**: энкодер Transformer + предобучение, затем fine-tuning.

Первоисточники

Word2Vec (2013) seq2seq (2014) Attention (2014) fastText (2017) Transformer (2017) BERT (2018)

Карта задач: где «включался» контекст

Статические эмбеддинги

- похожие слова, кластеры
- поиск похожих документов
- baseline-классификация через doc-vectors

Последовательные модели

- перевод (seq2seq)
- language modeling (следующий токен)
- NER/POS (метка для каждого токена)
- тональность и интенты (важен порядок)

Но

«банк» (финансы) и «банк» (река) получают один вектор.

Дальше

Attention и Transformer дают более прямой доступ к нужному контексту.

Word2Vec: идея и что реально обучается

Интуиция

Мы «заставляем» вектор слова быть полезным для угадывания его окружения. Если «телефон» часто стоит рядом с «купить», «зарядка», «смартфон», то вектор закрепляет эти связи.

Что происходит при обучении

- берём окно контекста и строим пары (центр, контекст) или (контекст, центр)
- обновляем **матрицы эмбеддингов** так, чтобы реальные пары получали высокий score

Ссылки

Mikolov et al., 2013 Illustrated Word2Vec

CBOW и Skip-gram (формулы + мини-пример)

CBOW и Skip-gram

$$\text{CBOW: } \max \sum_{t=1}^{T} \log p(w_t \mid C_t), \qquad \text{Skip-gram: } \max \sum_{t=1}^{T} \sum_{c \in C_t} \log p(w_c \mid w_t)$$

Мини-пример

Фраза: «вчера я купил новый телефон».
Skip-gram делает «купил» полезным для «я/новый/телефон», а «телефон» полезным для «новый/купил».

Что фиксировано в постановке

Окно контекста, размерность векторов, стратегия негативов — гиперпараметры (задаём руками).

Negative Sampling: почему это быстро (формула + смысл)

Идея

Вместо полного softmax по словарю обучаем контраст: реальные пары притягиваем, случайные отталкиваем.

SGNS-потеря

$$\mathcal{L}_{\text{SGNS}}(w, c) = -\log \sigma(v_c^\top v_w) \; - \sum_{k=1}^{K} \log \sigma(-v_{n_k}^\top v_w)$$

Что обучается на практике

Меняются вектора слов/контекстов; K и распределение негативов задаются как настройки.

fastText: подслова и морфология (формула + пример)

Зачем это появилось

Русский язык: много форм и редких словоформ. Хотим, чтобы «эконометрик» и «эконометрический» были похожи даже при редких встречах.

Как строится вектор слова

$$v_w = \sum_{g \in G(w)} z_g$$

где z_g — обучаемые вектора символьных n-грамм.

Ссылка

Bojanowski et al., 2017

Почему появились RNN: когда «мешка слов» не хватает

Интуиция

В задачах перевода, NER или тональности порядок важен. «не понравился» и «понравился» — разные смыслы, хотя слова почти те же.

Где RNN стали стандартом

- language modeling: следующий токен по истории
- seq2seq: перевод/суммаризация (вход $\rightarrow$ выход)
- tagging: POS/NER (метки по токенам)

Ссылка

Sutskever et al., 2014

RNN: что в ней обучается (формула + пример)

Рекуррентная динамика

$$h_t = \phi(W_x x_t + W_h h_{t-1} + b)$$

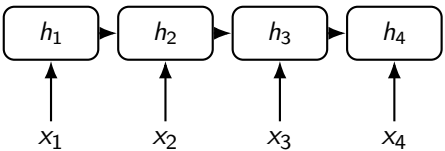
Пояснение

Матрицы W_x, W_h (и выходные слои) обучаются градиентным спуском. На вход x_t часто подавали готовые Word2Vec/fastText-вектора, а иногда доучивали эмбеддинги вместе с моделью.

Пример

В «я **не** думаю, что он прав» отрицание далеко: простая модель может «забыть» его без хорошей памяти.

Схема RNN (развёртка по времени)



последовательно: шаг t зависит от шага $t-1$

Ограничения

Сложно параллелить и трудно держать очень дальний контекст (затухающие градиенты).

LSTM: почему это улучшение RNN (формулы + смысл)

Интуиция

LSTM добавляет отдельную «память» c_t и гейты, которые учатся хранить и стирать информацию. Параметры гейтов обучаются так же, как обычные веса нейросети.

Формулы

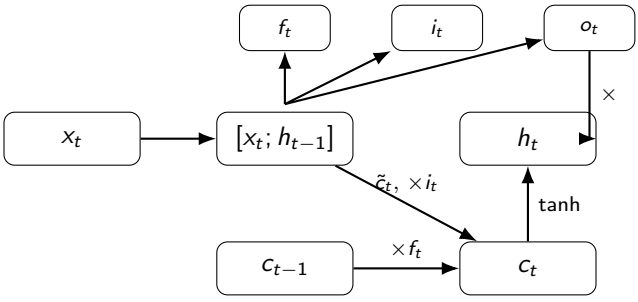
$$f_t = \sigma(W_f[x_t; h_{t-1}] + b_f),\ i_t = \sigma(W_i[x_t; h_{t-1}] + b_i),\ o_t = \sigma(W_o[x_t; h_{t-1}] + b_o)$$

$$\tilde{c}_t = \tanh(W_c[x_t; h_{t-1}] + b_c),\ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t,\ h_t = o_t \odot \tanh(c_t)$$

На пальцах

f_t — «что забыть», i_t — «что добавить», o_t — «что выдать».

Схема LSTM-ячейки (память как отдельный поток)



идея: память c_t помогает хранить информацию дольше

Attention «на пальцах»: как модель выбирает важные места

Метафора

Когда вы отвечаете на вопрос по тексту, вы **выделяете нужные фрагменты** и игнорируете остальное. То же делает attention: расставляет веса важности и собирает контекст как сумму.

Пример

«**не** понравился фильм»: если внимание игнорирует «не», получим неправильную тональность.

Ссылка

Bahdanau et al., 2014

Attention: формулы (общий вид)

Вес и контекст

$$\alpha_i = \mathrm{softmax}\big(\mathrm{score}(q, k_i)\big), \qquad c = \sum_{i=1}^{n} \alpha_i v_i$$

Score функции

$$\mathrm{score}(q, k) = q^\top k \quad \text{(dot-product)}, \qquad \mathrm{score}(q, k) = w^\top \tanh(W_q q + W_k k) \quad \text{(additive)}$$

Комментарий

Параметры score-функции обучаются вместе с моделью; softmax и взвешенная сумма — фиксированная часть механизма.

Transformer: что именно обучается в self-attention

Проекции и внимание

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \qquad \mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

Интуиция на примере

Для токена «фильм» полезно смотреть на «понравился» и «не». Для токена «прав» — на «не думаю» в начале.

Что меняется при обучении

Матрицы W_Q, W_K, W_V (и далее W_O и FFN) обучаются; формула softmax и нормировка фиксированы.

Ссылка

Vaswani et al., 2017

Маски в энкодере: какие бывают и зачем (BERT-случай)

Два разных «mask» в BERT

- **attention mask**: технически скрывает `[PAD]` (padding mask)
- **MLM mask**: часть данных предобучения (в тексте появляются `[MASK]`)

Какие маски в attention обычно нужны энкодеру

Padding mask — да. Causal mask — нет (он нужен для генерации, но BERT — энкодер).

Как работает attention mask (padding) в формуле

Идея

Перед softmax добавляем большой отрицательный штраф туда, куда смотреть нельзя.

Формула

$$A = \mathrm{softmax}\left(\frac{QK^\top + M}{\sqrt{d_k}}\right), \qquad M_{ij} \approx -10^9 \text{ если позиция } j \text{ запрещена}$$

$$\mathrm{Attention}(Q, K, V) = A\,V$$

Пример

Если дополнили последовательность `[PAD]`, padding mask гарантирует, что `[PAD]` не влияет на контекст.

BERT: что именно «учится» на предобучении

Интуиция

BERT учится восстанавливать пропуски, используя **левый и правый контекст** сразу. Поэтому представление токена становится контекстным: одно и то же слово в разных предложениях даёт разные вектора.

Потеря MLM

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in M} \log p_\theta(x_i \mid x_{1:n} \setminus x_i)$$

Что реально обучается

Обновляются все веса энкодера Transformer и выходная проекция в словарь; доля масок и правило 80/10/10 задаются как настройка.

Ссылка

Devlin et al., 2018

MLM: правило 80/10/10 + пример

Как маскируют (классический BERT)

Берём 15% позиций:

- 80% заменяем на `[MASK]`
- 10% заменяем на случайный токен
- 10% оставляем без изменения

Пример

«я `[MASK]` понравился фильм» $\Rightarrow$ модель восстанавливает «не».

Зачем 10% “оставить как есть”

Чтобы модель не «привыкала», что при предсказании всегда видит `[MASK]`.

BERT в задаче: feature-based и fine-tuning (без громких заголовков)

Два режима использования

- **Feature-based**: энкодер BERT не трогаем, берём его вектора и сверху обучаем простую модель.
- **Fine-tuning**: энкодер и классификационная голова дообучаются вместе под задачу.

Классификация через [CLS]

$$\hat{y} = \mathrm{softmax}(W h_{\mathrm{CLS}} + b)$$

На практике

Feature-based быстрее и проще как baseline; fine-tuning обычно даёт лучшее качество при достаточной разметке.

Почему BERT можно считать «супер-эмбеддингом» текста

Что делает его сильнее статических векторов

- **контекстность**: «банк» в финансах и «банк» у реки дают разные представления
- **дальние зависимости**: attention соединяет далёкие слова напрямую
- **перенос**: предобучение на больших корпусах даёт сильные признаки даже для малых датасетов

Как получить вектор текста

- `[CLS]` как «сводка» предложения: h_{CLS}
- **mean pooling** по токенам (без `[PAD]`):

$$h_{\text{mean}} = \frac{\sum_{i=1}^{n} m_i h_i}{\sum_{i=1}^{n} m_i}$$

Интуитивная картинка

`[CLS]` через много слоёв self-attention «вбирает» информацию со всех токенов.

Ссылки по темам

- Word2Vec: Mikolov et al., 2013 / jalammar
- seq2seq: Sutskever et al., 2014
- Attention: Bahdanau et al., 2014
- fastText: Bojanowski et al., 2017
- Transformer: Vaswani et al., 2017 / jalammar
- BERT: Devlin et al., 2018

Как читать

Сначала illustrated-посты (интуиция), затем первоисточники (Vaswani, Devlin).

Итог: логика перехода к BERT

Цепочка идей

- Word2Vec/fastText дают статические вектора, но без контекста
- RNN/LSTM учитывают порядок, но тяжело держат дальние зависимости и плохо параллелятся
- Attention даёт прямой доступ к важным словам
- Transformer делает self-attention ядром, а BERT использует энкодер + MLM

Вопросы?