Обработка текста на естественном языке (NLP)
Лекция 2: Word2Vec/fastText и семантическое сходство

Преподаватель: *Четвергов Андрей Сергеевич*

РАНХиГС — ИЭМИТ
Кафедра эконометрики и математической экономики

Весенний семестр 2026

План лекции

1. Зачем эмбеддинги после TF-IDF
2. Представления слов: one-hot vs эмбеддинги
3. Распределительная гипотеза и «смысл = контекст»
4. Count-based vs prediction-based (контекст к Word2Vec)
5. Word2Vec: что это, кто сделал, как устроено
6. Семантическое сходство и оценка эмбеддингов
7. fastText: подслова и русский язык
8. Как использовать эмбеддинги в пайплайне
9. Инструменты и материалы
10. Следующий шаг

Коротко: где мы были на лекции 1

Базовый pipeline

данные → предобработка → представление (TF-IDF) → модель (LogReg/SVM) → метрики → анализ ошибок

Что TF-IDF делает хорошо

- быстрый и честный baseline
- интерпретируемость (важные слова/фразы)
- работает на небольших данных

Что TF-IDF не умеет (мотивация лекции 2)

- не знает, что **«машина»** ≈ **«автомобиль»**
- плохо переносит **морфологию** (особенно русский)
- разреженность признаков, слабое «обобщение»

Главная идея лекции 2

Смена представления

слово $\rightarrow$ **вектор** (плотный, небольшой размерности)

Зачем это нужно

- измерять семантическую близость (похожие слова рядом)
- работать с редкими/новыми словами лучше, чем TF-IDF
- строить более «смысловые» признаки для классификации/поиска

Чему научимся

- что такое count-based и prediction-based эмбеддинги
- что такое Word2Vec и fastText (кто/когда/зачем)
- как понимать «смысл = геометрия» и считать сходство

Что такое представление слова

Почему вообще нужен «вектор»

ML-модели работают с числами: текст $\rightarrow$ токены $\rightarrow$ **вектора** (вход в модель).

Lookup table (идея эмбеддингов)

- фиксированный словарь $|V|$
- каждому слову сопоставляем вектор размерности d
- матрица эмбеддингов: $E \in \mathbb{R}^{|V| \times d}$

Практика

Вектор слова — это «адрес» в семантическом пространстве: близкие слова должны быть рядом.

One-hot: почему это плохая идея для «смысла»

One-hot

Для i-го слова: вектор длины $|V|$ с 1 в позиции i и 0 в остальных.

Проблемы one-hot

- размерность слишком большая ($|V|$ может быть 10^5–10^7)
- любые два разных слова ортогональны $\Rightarrow$ **нет сходства**
- «не знает» про синонимы/тематику/контекст

Вывод

Нужны **плотные** представления, которые кодируют контексты.

Распределительная гипотеза (интуиция)

Короткая формулировка

Слова с похожими контекстами имеют похожие значения.

Классическая цитата

"You shall know a word by the company it keeps." (J. R. Firth)

Примеры

- «купить ___», «продажа ___», «новый ___» $\Rightarrow$ **товар/объект**
- «президент ___», «столица ___», «границы ___» $\Rightarrow$ **страна**

Вывод

Можно **обучать** представление слов из больших корпусов без ручной разметки.

Смысл как геометрия: три базовых операции

1) Сходство (similarity)

Слова близки $\Rightarrow$ похожие контексты $\Rightarrow$ похожий смысл.

2) Кластеры

Профессии рядом, города рядом, бренды рядом.

3) Аналогии (популярная демонстрация)

$$\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen}$$

Важно

Аналогии красивы, но в прикладных задачах важнее **сходство** и **качество на метриках**.

Два семейства эмбеддингов: в чём разница

Count-based (статистические)	**Prediction-based (предсказательные)**
Глобальная статистика co-occurrence	Локальный контекст в скользящем окне
Строим матрицу *слово–контекст*, затем сжимаем	Не строим матрицу: учим модель предсказывать слово/контекст
Часто: Raw counts / PPMI + SVD	Градиентный спуск обновляет веса; веса $\approx$ эмбеддинги
Примеры: LSA/LSI, PPMI+SVD, GloVe (смешанный)	Примеры: Word2Vec (CBOW/Skip-gram), fastText

Зачем это знать

И count-based, и prediction-based реализуют одну идею: **контекст кодирует смысл**, но делают это разными путями.

Count-based в 3 шага: co-occurrence → PPMI → SVD

1) Co-occurrence

Считаем, как часто слово w встречается рядом с контекстом c (окно $\pm k$).

2) PPMI (взвешивание)

PMI показывает, встречаются ли w и c вместе чаще, чем «по случайности»; PPMI оставляет только положительные связи.

$$\mathrm{PMI}(w,c) = \log \frac{P(w,c)}{P(w)P(c)}, \qquad \mathrm{PPMI}(w,c) = \max(0, \mathrm{PMI}(w,c))$$

3) SVD (сжатие)

Сжимаем матрицу до k компонент $\Rightarrow$ получаем плотные вектора слов/документов (LSA/LSI).

GloVe: «глобальные» вектора слов (идея)

Интуиция

Смысл слова проявляется в **глобальной** статистике co-occurrence по всему корпусу.

Суть в одном предложении

Подбираем вектора слов и контекстов так, чтобы их скалярное произведение приближало log co-occurrence, при этом редкие/частые пары взвешиваются разумно.

Почему это здесь

GloVe часто рассматривают как «мост» между count-based и prediction-based.

Word2Vec: минимум исторического контекста

Кто и когда

- **2013**: Tomas Mikolov и команда Google
- цель: быстрые эмбеддинги слов на больших корпусах

Что это по сути

Не «одна модель», а **семейство простых задач**, которые учатся предсказывать слова и контексты.

Открыть при желании

- J. Alammar: Illustrated Word2Vec
- Mikolov et al., 2013 (arXiv)

Pipeline Word2Vec (как обучается на корпусе)

Общая логика

- берём большой корпус
- идём по тексту скользящим окном
- для «фокусного» слова предсказываем контекст (или наоборот)
- обновляем вектора так, чтобы правильные пары получали большую вероятность

Деталь, которую полезно помнить

В классическом Word2Vec часто есть **два вектора на слово**: как «центр» и как «контекст»; на практике их усредняют/берут один из них.

Две архитектуры Word2Vec: CBOW и Skip-gram

CBOW (Continuous Bag-of-Words)

- вход: контекст (окно вокруг слова)
- выход: центральное слово
- быстрее, хорошо на частых словах

Skip-gram

- вход: центральное слово
- выход: слова контекста
- лучше на редких словах, но тяжелее

Ключ

Обе модели учатся так, чтобы **вектор слова** был полезен для предсказания его окружения.

Как это выглядит «на пальцах» (схема окна)

Skip-gram: **«купил»** $\rightarrow$ предсказываем слова контекста

Практический смысл

Модель «вынуждена» закодировать в векторе свойства слова, которые помогают угадывать окружение.

Почему обучение Word2Vec быстрое: negative sampling

Проблема

Softmax по словарю размера $|V|$ дорог: обновлять параметры «для всех слов» медленно.

Идея negative sampling (упрощённо)

- есть **позитивная** пара (центр, контекст) $\rightarrow$ хотим score $\uparrow$
- берём K **негативных** слов из словаря $\rightarrow$ хотим score $\downarrow$
- вместо полного softmax учим «контраст» на малом числе примеров

Что запомнить

NEG — ключевая причина, почему Word2Vec стал практически применим на больших данных.

Что влияет на качество эмбеддингов (гиперпараметры)

Главные настройки

- размерность вектора (часто 100–300)
- размер окна контекста (часто 5–10)
- CBOW vs Skip-gram
- число negative samples (K)

Практические ориентиры

- маленькие датасеты: $K \approx 15$–20
- большие корпуса: $K \approx 2$–5
- окно меньше $\Rightarrow$ больше «синтаксики»
- окно больше $\Rightarrow$ больше «тематики»

Практика

Часто берут готовые pretrained-вектора, а обучение «с нуля» оставляют для больших корпусов.

Как измерять близость: cosine similarity

Косинусная близость

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}$$

Почему именно она (интуитивно)

- важнее **направление** вектора (смысл), чем длина
- устойчивее к масштабу

Альтернатива (когда полезно)

Jaccard similarity — для сравнения множеств/BoW (например, наборов слов/шинглов).

Оценка эмбеддингов: intrinsic vs extrinsic

Intrinsic (внутренняя)

- быстро считать
- помогает понять, «что выучилось»
- примеры: похожие слова, аналогии, word similarity datasets

Extrinsic (внешняя)

- оценка на реальной задаче (классификация/поиск/NER)
- дорого: надо честно сравнить пайплайны
- главный критерий: прирост вашей метрики

Правило

Красивые соседи в embedding space не гарантируют прироста F1/PR-AUC — это нормально.

fastText: что добавили по сравнению с Word2Vec

Кто и когда

- **2016–2017**: Facebook AI Research (FAIR)
- ключевая идея: **подсловные n-граммы** (subword)

Зачем это нужно

- редкие/новые слова (OOV) не «обнуляются»
- морфология (русский) учитывается лучше

Открыть

- Bojanowski et al., 2017: subword vectors (arXiv)
- fastText (официальный сайт)

Как fastText собирает вектор слова

Упрощенная формула

Вектор слова = сумма (или среднее) векторов его символьных n-грамм.

Пример (идея)

слово: **эконометрика**

n-граммы: эко, кон, ном, оме, ...

Почему это важно для RU

«эконометрика», «эконометрический», «эконометрик» имеют общие подстроки $\Rightarrow$ близкие вектора.

Практический вывод

fastText часто — «безболезненный» шаг вперед по сравнению с Word2Vec на русском.

Ограничения классических эмбеддингов (честно)

Контекст

Word2Vec/fastText дают **одно** представление на слово (не различают «лук» как оружие и как овощ).

Полисемия и дискурсы

Смысл зависит от контекста, а классические эмбеддинги контекст не видят.

Мост к следующей теме

Контекстные эмбеддинги (ELMo/BERT) решают это: вектор зависит от окружения.

Быстрые способы применить эмбеддинги (без deep learning)

1) Документ как среднее по словам

$$\vec{d} = \frac{1}{n} \sum_{i=1}^{n} \vec{w}_i$$

Потом: **LogReg/SVM** на документных векторах.

2) Взвешенное среднее (часто лучше)

TF-IDF веса $\times$ эмбеддинги $\rightarrow$ один вектор документа.

3) Поиск/дедупликация

Сравниваем документы по cosine similarity, находим «похожие» обращения/новости.

Сравнение подходов: что обещаем студентам

Baseline

- TF-IDF + LogReg/SVM
- метрики + разбор ошибок

Альтернатива

- fastText/Word2Vec $\rightarrow$ doc vectors
- LogReg/SVM
- сравнение по тем же метрикам

Правило курса

Одинаковая постановка задачи + одинаковое разбиение данных $\Rightarrow$ честное сравнение.

Мини-кейс на семантическое сходство (идея для практики)

Задача

Есть обращения пользователей. Нужно:

- находить «похожие» обращения (кластер/дедуп)
- искать похожий кейс для ответа поддержки

Пайплайн

1. чистка + токенизация
2. doc-vector (среднее fastText)
3. cosine similarity $\rightarrow$ top-k соседей

Что важно в выводах

Как меняются результаты, если убрать стоп-слова, числа, имена? Что считается «хорошим совпадением»?

Инструменты (минимум для семинара)

Что используем

- **gensim** — Word2Vec (обучение/загрузка)
- **fastText** (или готовые модели fastText)
- **numpy** — векторы
- **scikit-learn** — классификация/метрики/пайплайны

Совет

На практике важно: **сравнить с TF-IDF baseline** и показать **ошибки**.

Мини-набор ссылок (чтобы не «плавать»)

- Word2Vec (визуально): jalammar: illustrated-word2vec
- Word2Vec (оригинал): Mikolov 2013 (arXiv)
- fastText subword: Bojanowski 2017 (arXiv)
- GloVe: stanford: glove
- RusVectores: rusvectores.org
- Embedding Projector: projector.tensorflow.org

Мост к следующей лекции

Контекстные эмбеддинги и трансформеры: почему BERT «понимает контекст».

Следующее занятие: семинар 2

На семинаре 2 сделаем

- загрузим готовые эмбеддинги (Word2Vec/fastText)
- посчитаем **cosine similarity** и top-k похожих слов/документов
- соберём doc-vector и сравним классификацию с TF-IDF baseline

Что подготовить

- `numpy`, `scikit-learn`
- `gensim` (опционально)

Вопросы?