Home

Информационный поиск

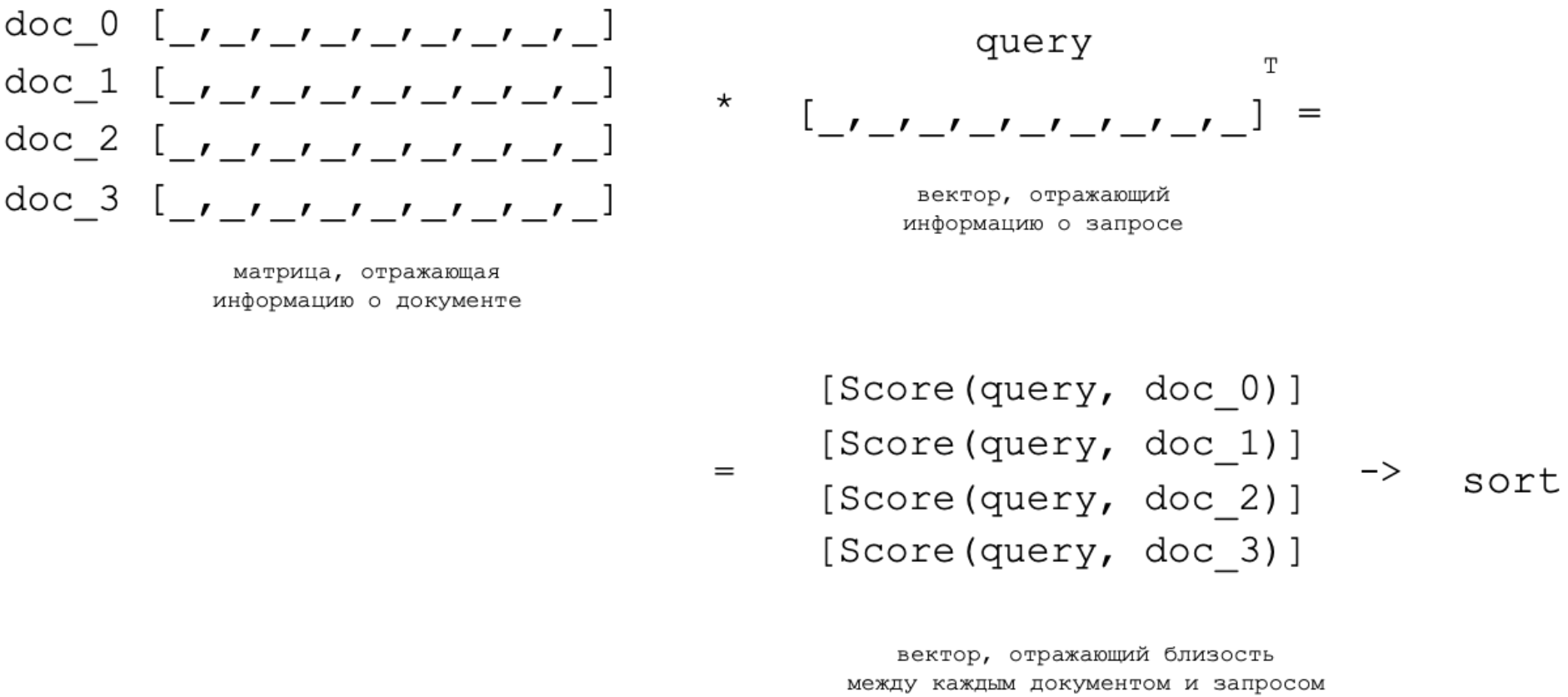
Лекция 4. Эмбеддинги

Telegram

GitHub

Ника Зыкова
2023/09/06

Векторная постановка задачи



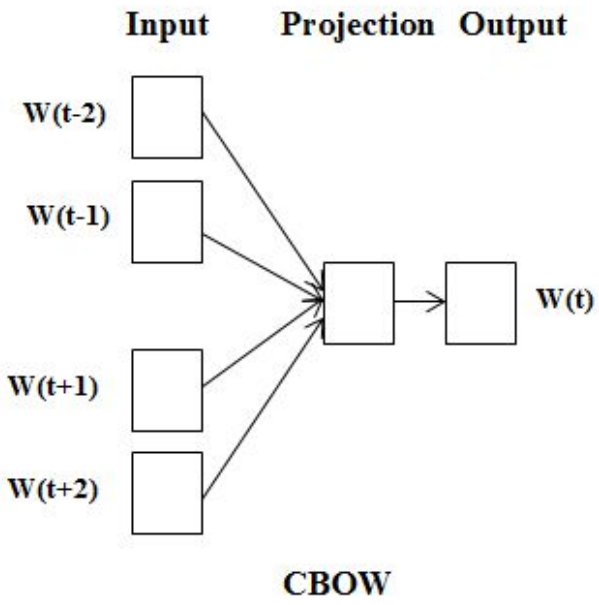
Чем можно наполнить матрицу?

Как уже было сказано, способов множество:

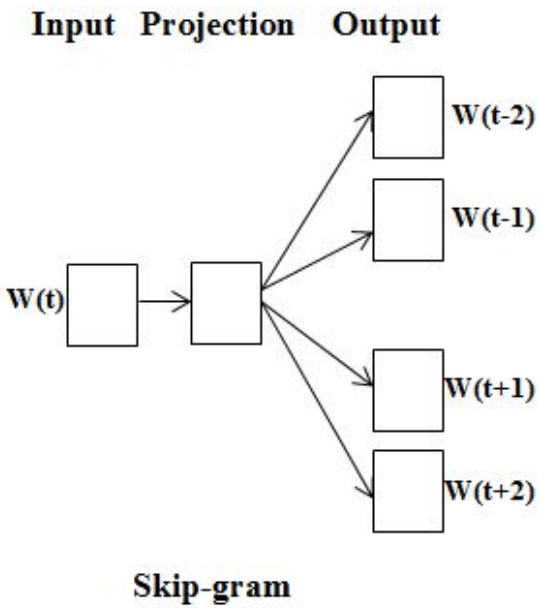
- бинарное кодирование (есть/нет) - OneHotEncoder (sklearn, но так никто не делает)
- кодирование по частоте - CountVectorizer (sklearn)
- TF-IDF - TfidfVectorizer (sklearn)
- BM25 - библиотека rank-bm25
- word2vec - gensim (чаще всего)
- bert - pytorch (чаще всего)
- и тд

Word2Vec

Учим на две задачи: восстанавливать слово по контексту (CBOW) и контекст по слову (skipgram). Плюс negative sampling.
Какие есть минусы у word2vec?



CBOW



Skip-gram

Чем плох word2vec?

- Плохо понимает разницу между синонимами и антонимами
  - Как мы уже видели, антонимы часто оказываются ближайшими друг для друга, что плохо в общем случае. Хотелось бы решить эту проблему
- Омонимы - это один и тот же вектор
  - Так как у word2vec для каждого слова есть всего один вектор, то для слов зАмок и замОк мы получим одно и то же, что однозначно неверно
- Не учитывает сочетаемость слов
  - Мы не можем учесть смысл слова в контексте, из-за чего модель не понимает идиомы, метафору и т.д.
- Не умеет работать с незнакомыми словами
  - Для слов, которые отсутствуют в словаре модели, у нас просто не будет вектора

Что еще?

- Можно использовать модели, близкие к word2vec:
  - Navec
  - FastText
  - Glove
- RNN и LSTM модели: ELMo and others
- Transformers модели:
  - BERT
  - RoBERTa, ALBERT, DistilBERT
  - ELECTRA

GloVe

- Минимизируем разницу между произведением векторов слов и логарифмом вероятности их совместного появления
- Учитываем частоту совместной встречаемости напрямую
- Не нейросеть (!), но оптимизируется похожими методами (стохастический градиентный спуск)
- Обгоняет word2vec на многих бенчмарках (но меньше open source моделей)

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Navec

- На основе GloVe
- Значительно меньше по занимаемой памяти, чем word2vec от RusVectores
- Не нужны части речи
- По опыту, знает больше слов из современного лексикона
- Вот тут можно почитать: https://natasha.github.io/navec/ (и табличка тоже оттуда)

	Качество	Размер модели, МБ	Размер словаря, ×10³
Navec	0.719	50.6	500
RusVectores	0.638–0.726	220.6–290.7	189–249

Home | Recent Tabs | Current Page

You can type something here...

FastText

- Использует все задачи word2vec: CBOW, Skipgram, negative sampling
- К word2vec добавлена модель символьных n-грамм: каждое слово - несколько цепочек символов заданной длины (с наложением), вектор слова – сумма всех его n-грамм.
- Умеет работать с очень редкими и незнакомыми словами
- Более устойчив к опечаткам (как следствие к предыдущему)

RNN and etc

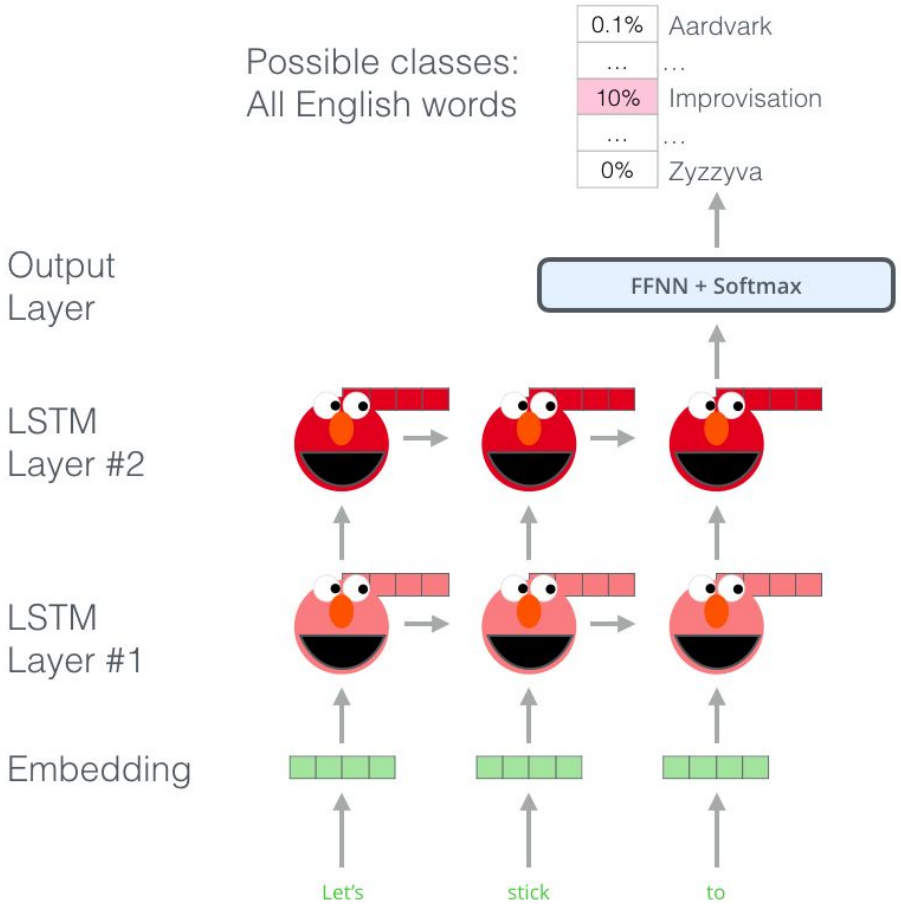
- RNN: помнит только ограниченный предшествующий контекст -> плохо работает с большими текстами (длинными предложениями)
- LSTM: суложненная работа с памятью, модель умеет динамически забывать/помнить -> более длинные тексты
- Attention Models: попытки реализовать внимание до Transfrormers

Общая проблема: таких моделей почти нет в свободном доступе, так как они проиграли BERT & Co. Но если обучать свою, то это хороший компромис между точностью и сложностью.

Elmo

- Есть статический эмбеддинг (он хранится как часть модели) и есть итоговый, который получается на его основе
- Обучается на задачу **языкового моделирования** (генерации)
- Но Bi-LSTM - учитывается контекст справа и слева

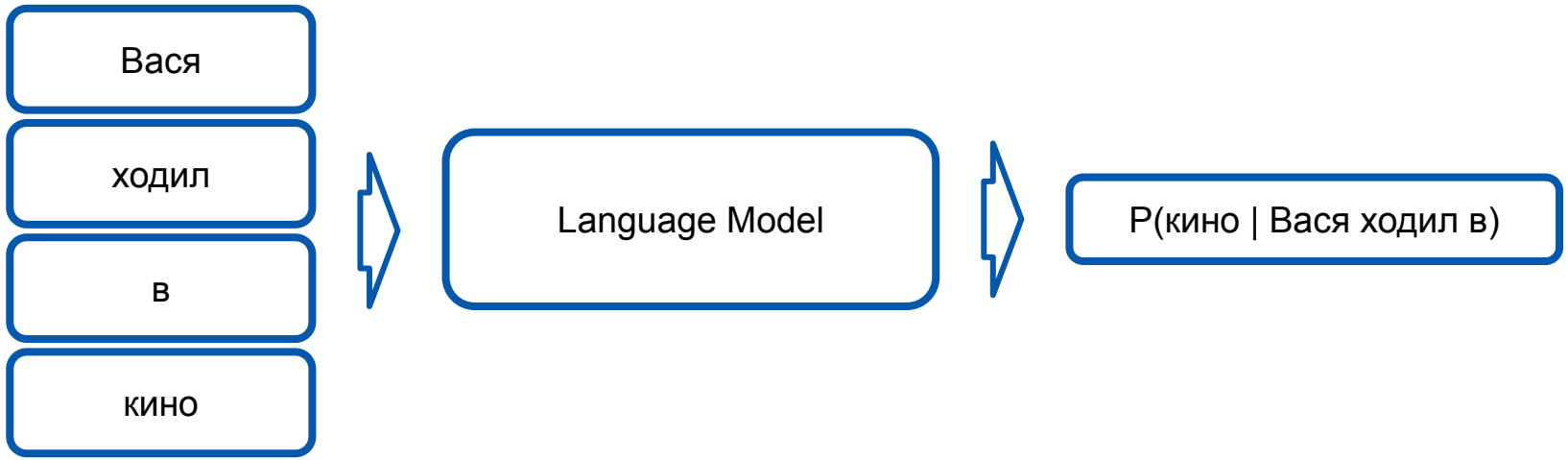
Неплохая модель, но с популярностью трансформеров используется редко.



Языковая модель

Языковая модель - это модель, способная оценить вероятность конкретной конструкции в языке.

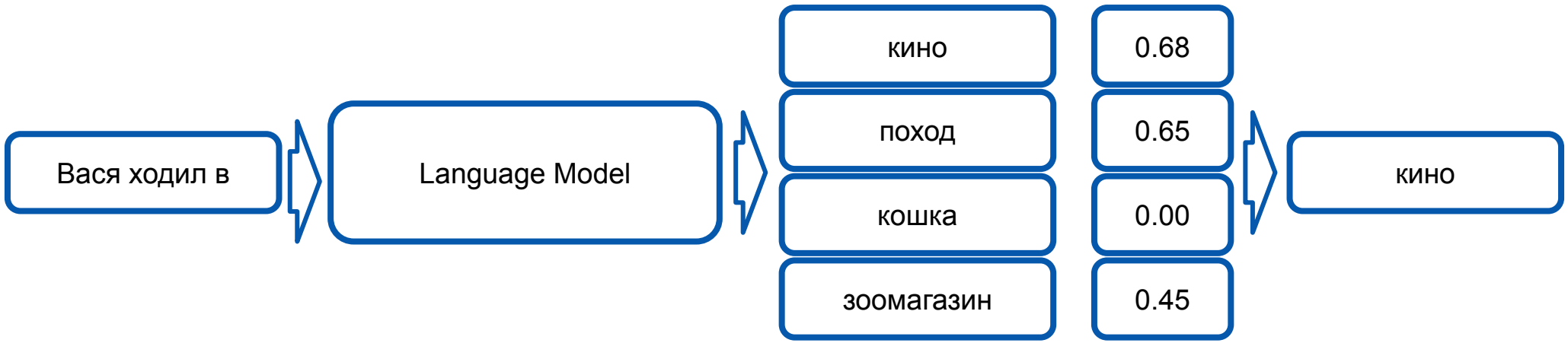
Чаще всего это означает, что по последовательности слов она может выдать вероятность последнего с учетом всех предыдущих.



Языковое моделирование

Модель учат выдавать наибольшую вероятность для последующего слова.

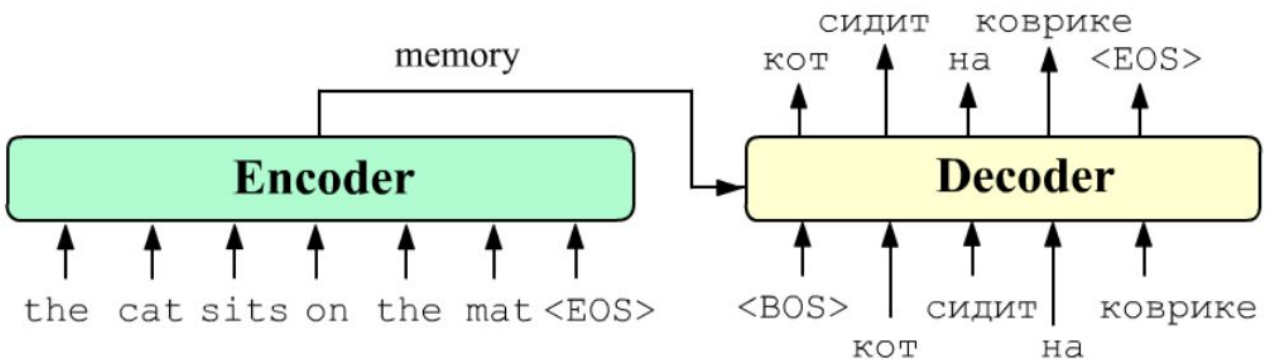
Требует много времени и данных, так как модель имеет только половину контекста (левую), но является единственным вариантом для генеративных моделей.

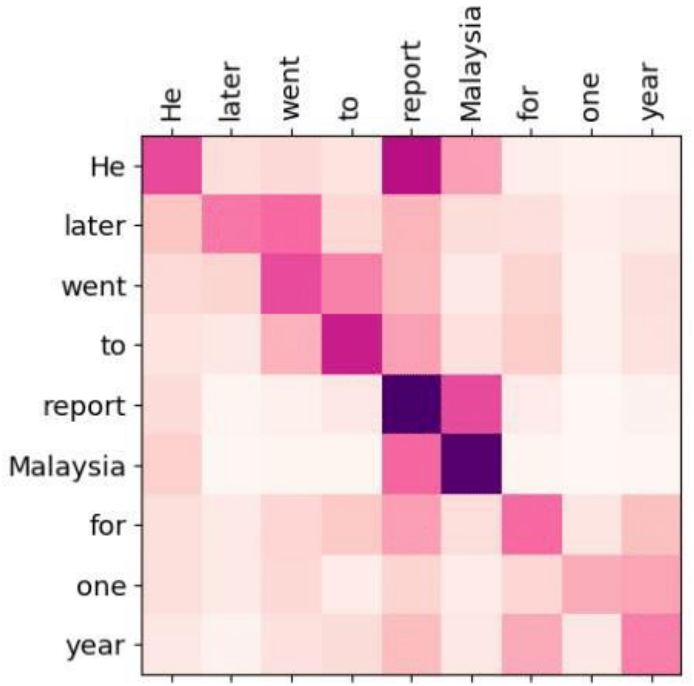


Transformers

В 2017 году Google представил архитектуру Transformers и ее механизм внимания.

Классическая архитектура Трансформера подразумевает наличие энкодера и декодера, но на практиве встречаются любые комбинации: только энкодер, только декодер или





Виды моделей

Encoder

- BERT
- RoBERTa
- ALBERT
- ELECTRA

Decoder

- GPT1, 2, 3, 4
- ChatGPT
- PaLM
- LLaMA

Encoder-Decoder

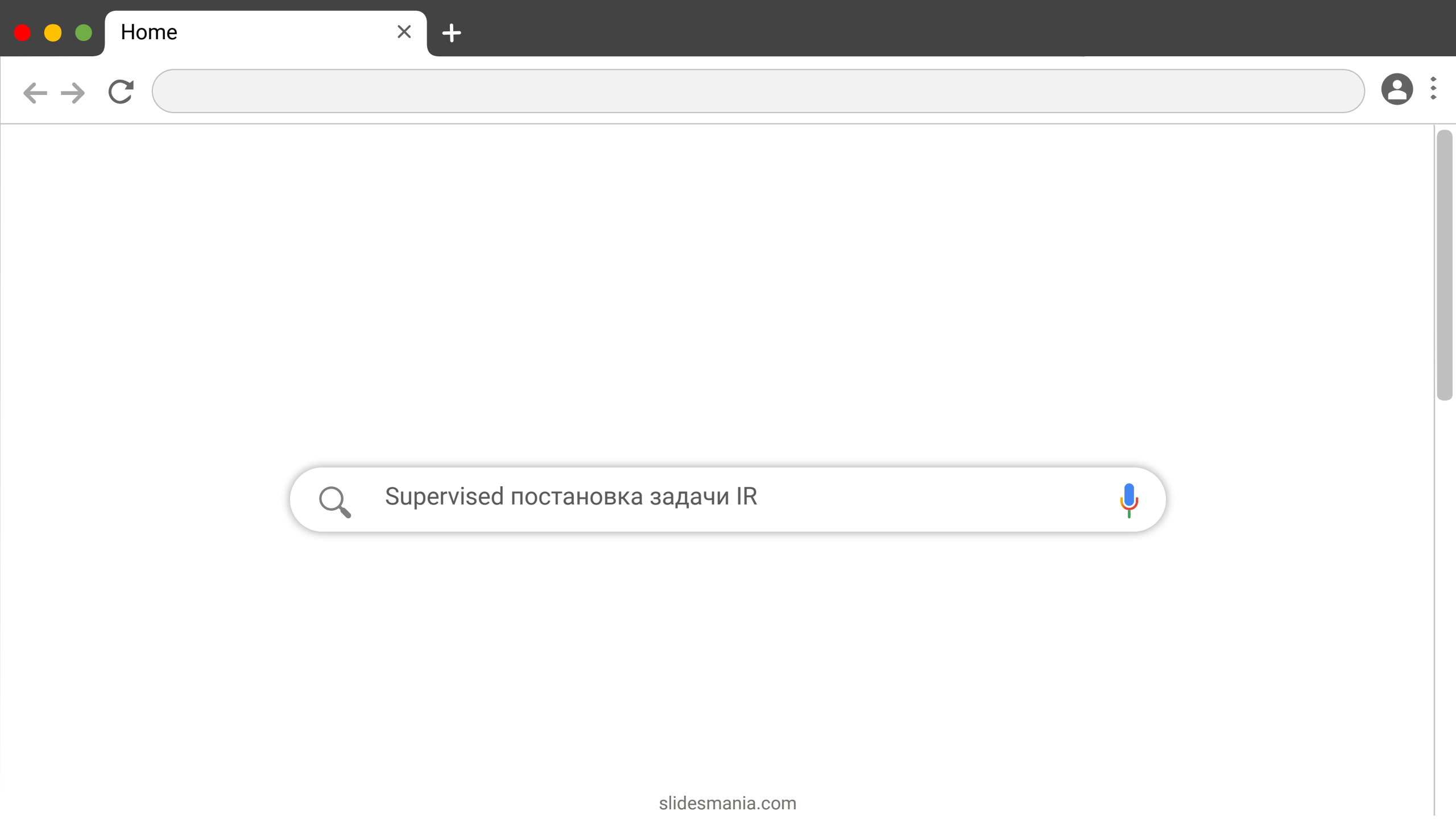
- T5
- Bart
- Pegasus
- ProphetNet

What do BERT, RoBERTa, ALBERT, SpanBERT, DistilBERT, SesameBERT, SemBERT, SciBERT, BioBERT, MobileBERT, TinyBERT and CamemBERT all have in common? And I'm not looking for the answer "BERT" 😉.

Выводы

1. Предобработка -> Индексирование -> Запрос -> Ранжирование
2. Документ - это вектор, запрос - вектор (и вообще, все на свете - это вектор...)
3. Можно использовать любой способ векторизации, обычно находят компромис между доступными ресурсами (время, железо, данные) и качеством поиска

А вообще, это все unsupervised постановка задачи, если решить ее в том виде, в котором решаем мы!



Постановка задачи

У нас есть данные:

- Список документов
- Список запросов
- Тройки вида (q, d1, d2), где q - это запрос, d1 - более релевантный запросу документ, d2 - менее релевантный.

Задача:

Для каждого запроса q упорядочить документы d1...dn так, чтобы это как можно точнее соответсвовало соотношениям в тройках. То есть, нужна модель, которая для менее релевантного выдаст меньшую оценку.

Другая постановка задачи

С парами документов неудобно работать: непонятно, как формализовать такой формат данных для фукнции потерь модели.

Будем работать по-другому:

- Объекты: пары из запроса и документа
- Ответы: такие числа, что большему числу означает большая релевантность (и должна соотвествовать большая, но не обязательно такая же оценка от модели)

Фактически, вместо того, чтобы руками определить меру релевантности, мы учим для этого модель.

Варианты задачи

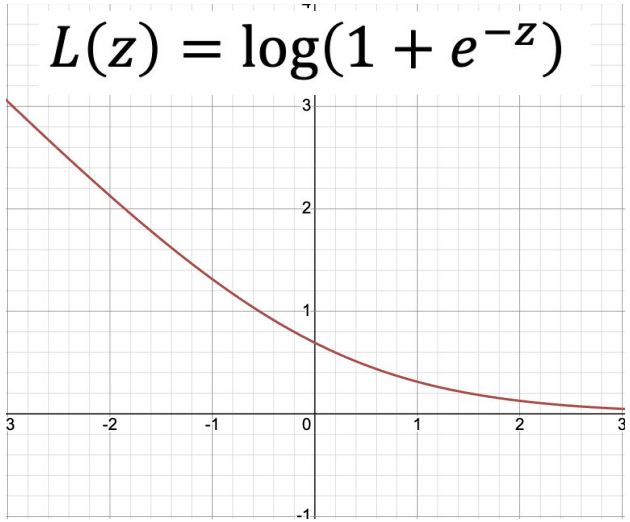
1. Pointwise (поточечная):
   - ➢ Предсказываем конкретное число.
   - ➢ Можно использовать любую модель для задачи регрессии
   - ➢ Проблема: никак не учитывается порядок и относительность оценок
2. Pairwise (попарная):
   - ➢ Предсказываем конкретное число, но учимся на парах: штрафуем за неправильную разницу в паре предсказаний
   - ➢ Сложнее постановка задачи, сложнее оптимизировать
   - ➢ Обычно дает более высокое качество

Pairwise и математика

Источник формул: https://github.com/hse-ds/iad-intro-ds/blob/master/2023/lectures/lecture18-ranking.pdf

Сначала запишем в том виде, в котором сформулировали:

$$\sum_{(q,d_i,d_j)\in R} [a(q,d_i) - a(q,d_j) < 0]$$



Неприятно, что полученная штука дискретна и производную посчитать не полчится. Сделаем так, чтобы было можно:

$$\sum_{(q,d_i,d_j)\in R} [a(q,x_i) - a(q,x_j) < 0] \leq \sum_{(q,d_i,d_j)\in R} L\left(a(q,x_i) - a(q,x_j)\right)$$

$$L(z) = \log(1 + e^{-z})$$

DCG (Discounted cumulative gain)

$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} \frac{2^{y_i} - 1}{\log(i+1)}$$

k - для скольки первых документов мы считаем оценку

y_i - правильный ответ для документа на позиции i (то, что лежит в датасете)

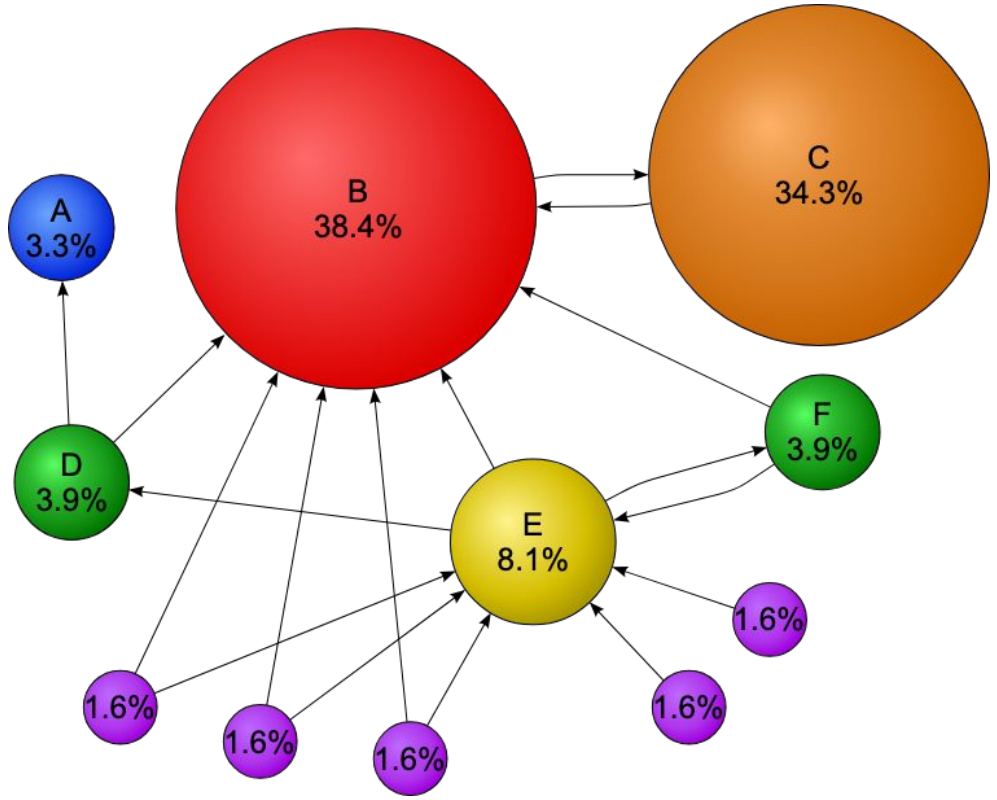
Усредняем для получения общей оценки.

PageRank

Чем больше документов, на которые ссылается данный, тем меньше ценность его ссылки.

Чем больше “дорогих” ссылок у документа, тем выше его уровень важности.

Метрика считается со случайного документа итеративно.



Итого

- Можем не подбирать руками способ оценки близости, а обучить для этого модель
- Можем обучить модель на векторизацию + оценку, таким образом решив все части задачи
- Задача имеет конкретное название в ML: ranking, в нашем случае Document Ranking