

KARPOV.COURSES >>> КОНСПЕКТ



> Конспект >1 урок >Вводная лекция: ранжирование, матчинг, архитектурные особенности

> Оглавление

> Оглавление

> Задача ранжирования в машинном обучении

Ранжирование как класс задач

Почему отдельный класс задач

Схематичное изображение системы ранжирования

> Ранжирование с точки зрения ML

> Матчинг

Для чего нужен матчинг

Кому нужен матчинг

Пример

КУПЛЕНО НА
SKLADCHIK.COM

Возможная проблема

- > SKU
- > Мультимодальные модели
- > Проблемы в матчинге
- > Приложение пайплайна матчинга

Кандидатная модель

Проблемы KNN:

- > Резюме

> Задача ранжирования в машинном обучении

Ранжирование — процесс упорядочивания набора объектов в соответствии с некоторой мерой, то есть задание частично упорядоченного множества.

Область применения ранжирования: от выдачи поисковых систем до формирования траекторий беспилотных автомобилей.

Множество **частично упорядочено**, если указано, какие элементы следуют за какими (какие элементы больше каких). В общем случае может оказаться так, что некоторые пары элементов не связаны отношением «следует за».

Важно: может оказаться так, что некоторые пары документов **не связаны** между собой, т.е. они могут быть либо на одной ступени некоторой шкалы, либо вообще не находиться на одной шкале, так как они никак не соотносятся с этим множеством.



Рассмотрим картинку: пусть есть некоторый запрос q . В результате часть документов упорядочена от лучшей релевантности к худшей, при этом есть

документы, которые не попали на шкалу, так как они никак не относятся к запросу.

Ранжирование как класс задач

Learning to rank (обучение ранжированию) — класс задач машинного обучения **с учителем** (*supervised*) или **с частичным привлечением учителя** (*semi-supervised*), заключающихся в нахождении модели, целью которой является наилучшее приближение и обобщение способа ранжирования в обучающей выборке на новые данные.

Обучение с учителем: указано, насколько какой-то документ подходит к конкретному запросу (насколько они релевантные).

Частичное привлечение учителя: использование, как правило, большого корпуса неразмеченных данных с целью улучшения качества предсказания.

Пример техник для semi-supervised обучения — **псевдолейблинг (pseudo-labeling)**: модель обучается на маленьком наборе данных с разметкой, а затем делает предсказания на всём огромном объёме данных. Предсказанные лейблы становятся новыми таргетами для обучения (новой целевой переменной). Таким образом можно получить улучшение качества в semi-supervised режиме.

Почему отдельный класс задач

Классификация отвечает на вопрос "какие документы релевантны", но не ранжирует их.

Регрессия может использовать какой-то бизнес показатель, например CTR (то, как много кликают на документ при выдаче в ответ на запрос) или время пребывания на странице, и учиться предсказывать такие показатели. Упорядочивать по ним — уже ближе к постановке исходной задачи ранжирования.

Однако, ранжирование выделяется в отдельный класс задач потому, что **задачи классификации и регрессии**, как правило, в один момент времени **оперируют лишь одним объектом**. То есть для каждого документа в пару к запросу нужно посчитать независимые от других документов предсказания. В то же время **задача ранжирования** решает проблему упорядочивания документов **над некоторым множеством объектов**.

Например, можно перебрать все пары документов, и определить, какой из них в паре должен находиться в выдаче выше, а какой — ниже. Таким образом, мы явно имеем возможность определить точный порядок, в котором необходимо расположить наши объекты. Этот **порядок задаётся мерой релевантности** между запросом и документом.

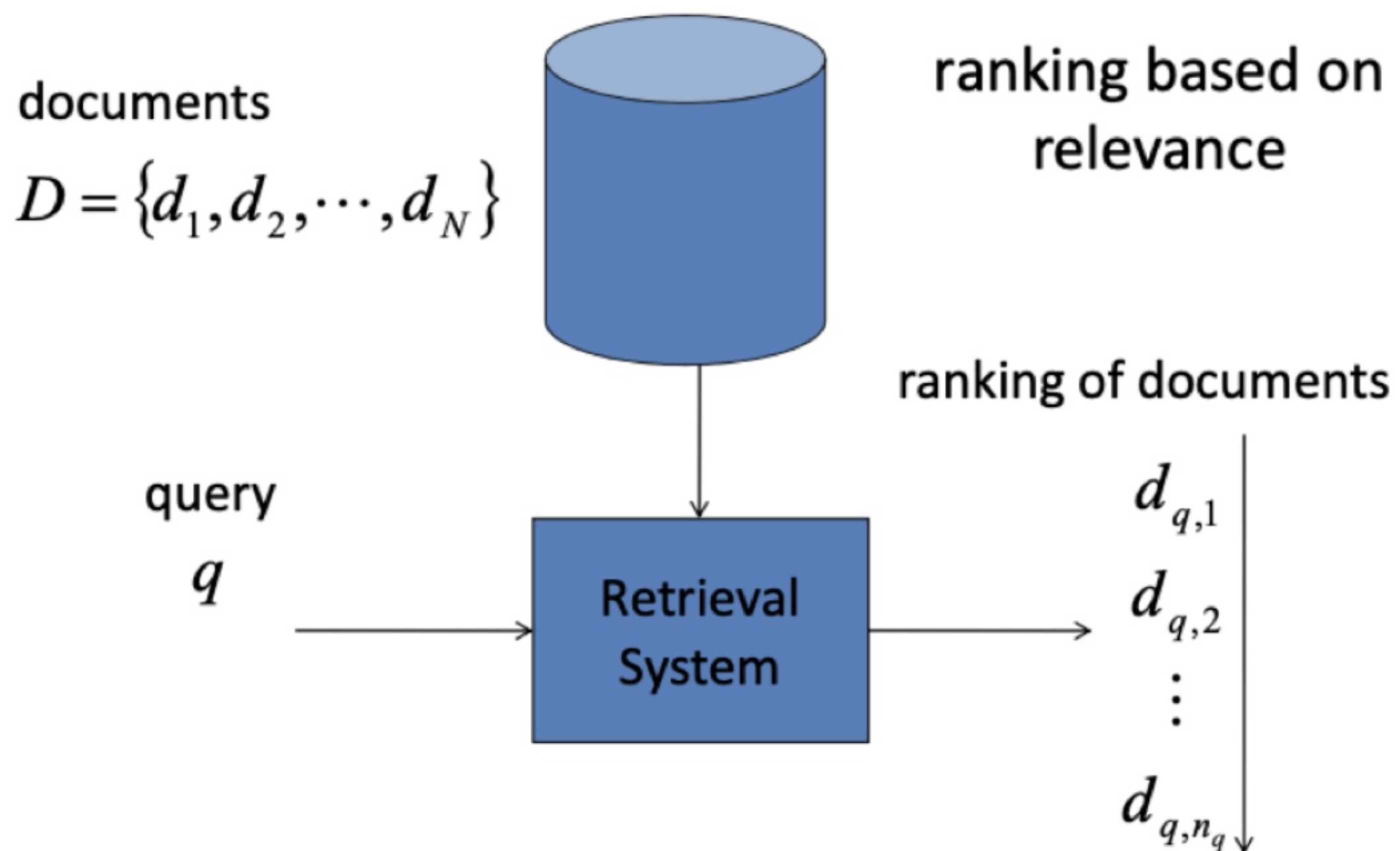
Мера релевантности — степень соответствия между запросом и документом (или набором документов). Чем выше это соответствие, тем выше в списке ранжирования должен находиться документ.

Схематичное изображение системы ранжирования



Устройство поиска Яндекса: извлечение совместных признаков из запроса и документа, выдача релевантности на их основе

+15



Общий вид работы с новым запросом

Рассмотрим последнюю картинку: есть заранее заготовленная база документов, которая подаётся в систему. В ответ на запрос из этих документов формируется упорядоченный по релевантности список. **Список не обязан содержать все документы**, он может быть сильно меньше, то есть точно нерелевантные документы можно отсекаать.

> Ранжирование с точки зрения ML

Q – набор запросов $\{q_1, q_2, \dots, q_m\}$

D – набор документов

$D_i = \{d_{i,1}, d_{i,2}, \dots, d_{i,n_i}\}$ – набор документов, релевантных i -му запросу q

$d_{i,j}$ – элемент с индексом j в D_i

$y_i = \{y_{i,1}, y_{i,2}, \dots, y_{i,n_i}\}$ – набор оценок релевантности для i -го запроса (размер тот же, что и у D_i)

$S = \{(q_i, D_i), y_i\}_{i=1}^m$ – тренировочный набор данных

$\overline{x_{i,j}} = \phi(q_i, d_{i,j})$ – вектор признаков для i -го запроса и j -го документа ($i = \overline{1, m}, j = \overline{1, n_i}$)

ϕ – функция для получения признаков (BM25, PageRank, мультимодальные модели)

$\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,n_i}\}$ – признаки набора документов, релевантных i -му запросу q

$f(q, d) = f(x)$ – ранжирующая модель, оценивающая релевантность для пары q, d на основе признаков x

$F(q, D) = F(\mathbf{x})$ – глобальная ранжирующая модель

Задача — создание такой глобальной модели, которая будет работать над всем корпусом документов.

> Матчинг

Матчинг — процесс сопоставления объектов на основе сравнения и расчета некоторой меры схожести, где объекты, с одной стороны, представляют собой «запросы», а с другой — «документы».

Для чего нужен матчинг

- Ценообразование;
- Мониторинг промоакций;
- Ассортиментное планирование;
- Мониторинг рынка (в том числе и со стороны производителя);
- Реализация функционала маркетплейса;
- Оперативный поиск.

Кому нужен матчинг

Ценообразование: без сопоставления цен с конкурентами в вакууме выставить и корректировать свои цены на товары может быть очень затруднительно.

Пример

Допустим, мы — ритейлер с названием "Х6". У нас есть 10 000 товаров в ассортименте для наших покупателей. Каждый товар со своими атрибутами,

включая цену, является **документом**, а все вместе они образуют **базу документов** — прямо как в ранжировании.

Х6



- Товар «Хлеб Черный Чусовой»
14.5 Рублей
280 Грамм
Поставщик «ООО Моя Оборона»
- ...

МАНИТ



- Товар «Чусовой (черный), Хлеб»
15.39 Рублей
280 Грамм
Поставщик - ?
- ...

+15

Сравнение своего товара с похожим товаром конкурента

Решаем, что наша стратегия ценообразования будет основана на сравнении с магазином "Манит". Заходим на его сайт с онлайн-продажей товаров, используем поиск и находим товар, очень похожий на один из товаров нашего магазина — это хлеб того же веса. Мы видим его цену, так как это открытая информация, но **не видим часть атрибутов**.

Все товары с сайта магазина-конкурента со своими полями (цена, название, картинка) — это **база запросов**.

На самом деле конкретно в этом примере можно безболезненно эти базы менять между собой местами.

Привычно называть "**базой документов**" ту часть, к которой у вас есть **полный доступ и права**. Наша база документов — это список товаров, которые мы продаём. Она есть и в экселях менеджеров, и в базах данных, то есть мы напрямую **видим все атрибуты с полным доступом**.

База документов не так часто меняется, и меняется не кардинально. А **запросы могут меняться в любом направлении** — например, если мы выберем другой магазин-конкурент для сравнения, то нам придётся заново весь его ассортимент прогонять через систему для сопоставления с нашими товарами, заново рассчитывать фичи и т.п.

Итак, мы внимательно смотрим на одну базу, на вторую, находим матч, видим разницу в цене, и если этот процесс повторить для большого количества

позиций, то от этого **можно строить свою стратегию**.

Скажем, почти все товары первой необходимости для людей преклонного возраста у вас дешевле. Останется сделать две вещи:

1. Сделать остальные товары дешевле для этой группы;
2. Уведомить их об этом, например, с помощью рекламы.

Это пример того, как от сопоставления цен можно выиграть.

Конкретное сопоставление конкретных товаров и сравнение их цен позволяет нам принимать решения, потенциально увеличивающие доход нашего магазина.

Вывод: с помощью матчинга вы можете корректировать ассортиментную матрицу.

Возможная проблема

Крайне важно иметь низкое количество ложных срабатываний, то есть избегать случаев, когда делаем матч, находим товар у конкурента, но при этом матч неправильный. Это **напрямую влияет на нашу ценовую стратегию**.

В задаче ранжирования мы практически всегда выдаём что-нибудь, хоть отдалённо напоминающее тему запроса. В случае матчинга, в отличие от ранжирования, можно **ничего не выдавать**. Потому в конец пайплайна матчинга добавляют классификатор, определяющий, корректен ли матч. Нам **лучше ничего не выдать, если нет сильной уверенности в правильности** ответа, нежели дать неправильный матч.

> SKU

SKU (*Stock Keeping Unit*, складская учётная единица) — идентификатор товарной позиции (*артикул*), единица учёта запасов, складской номер, используемый в торговле для отслеживания статистики по реализованным товарам/услугам. Каждой продаваемой позиции, будь то товар, вариант товара, комплект товаров (продаваемых вместе), услуга или некий взнос, назначается свой SKU.

SKU не всегда ассоциируется с физическим товаром, являясь скорее идентификатором сущности, представляемой к оплате.

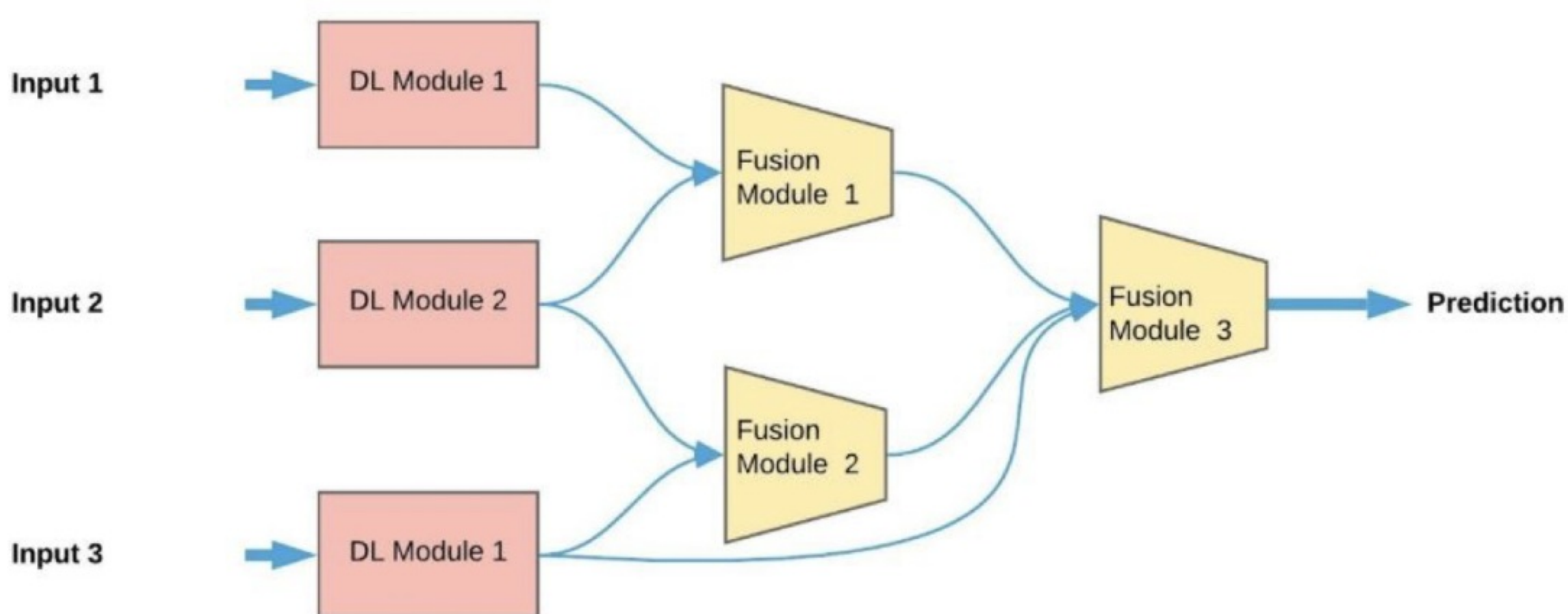
SKU не то же самое, что модель — на примере того же айфона понятно, что разные варианты цвета телефона соответствуют разным артикулам, но при этом модель телефона одна. Таким образом, **модель может содержать несколько SKU** (тип связи **один ко многим**), но одна **SKU может иметь только одну модель** (тип связи **один к одному**).

Для одного SKU может быть как ноль, так и сотня офферов (*предложений о продаже*). Разные офферы могут быть от разных магазинов.

> Мультимодальные модели

Говоря о данных, можно отметить, что они могут быть разной природы: изображения товара, набор каких-то параметров, текстовая информация из описания товара или заголовка.

Мультимодальные модели — модели, которые принимают и обрабатывают совместно данные разной природы; они оперируют совместными представлениями всех данных.



Общий вид: разные данные обрабатываются разными подсетями, затем создается общее представление. Перед вынесением решения данные смешиваются.

> Проблемы в матчинге

- Неполнота информации (отсутствуют параметры товаров);
- Разные стандарты заполнения параметров;
- Англицизмы, сокращения, опечатки;

- Высокие требования к качеству;
- Большая схожесть между разными товарами.

> Приложение пайплайна матчинга

- Поиск и удаление дубликатов в базе документов;
- Кластеризация входного потока запросов;
- Замена товаров из корзины пользователя.

Типичное использование систем ранжирования с машинным обучением — **re-ranking** (перанжирование или переранжирование): не строим модель, которая на самом деле оперирует всеми документами, которые у нас есть, так как это будет очень долго (представьте себе расчёт для миллионов объектов).

Вместо замены всего алгоритма поиска или поискового движка (*search engine*) с помощью модели машинного обучения мы добавляем в пайплайн дополнительный шаг. После того, как запрос был направлен в индекс, лучшие результаты соответствия будут поданы в модель, и после чего следует переаранжирование перед тем, как вернуть ответ на запрос.

Модель первого уровня в данном случае можно назвать **кандидатной моделью** — из всех документов она выдаёт кандидатов на ранжирование, т.е. самые перспективные, по ее мнению, документы, отвечающие на запрос.

То же применимо и для матчинга. Сначала мы **находим очень похожие товары**: например, в случае продуктового оффлайн ритейла наши кандидаты — весь хлеб от одного производителя. Затем уже пристально **в автоматическом режиме сравниваем все параметры товаров из этого списка** и выносим финальный вердикт — что является матчем, а что нет.

Кандидатная модель

Можно делать очень **простую и легковесную модель**, которая будет успевать оценивать релевантность всех документов, затем отбирать топ-k кандидатов и отдавать их модели второго уровня.

А можно попытаться **сразу найти ближайшие релевантные документы**. В этом поможет поиск по ближайшим соседям — аналог алгоритма kNN. *Подразумевается, что объекты некоторым образом предварительно векторизованы.*

Проблемы KNN:

1. Очень медленный, ведь нужно посчитать расстояние до всех документов, что аналогично модели машинного обучения, т.е. предыдущему подходу с обработкой всех объектов.
2. При большом количестве документов требуется колоссальный объём ресурсов, в частности, оперативной памяти.

Есть алгоритмы приближённого поиска ближайших соседей — **ANN** (*approximated nearest neighbours*). Они позволяют делать то же самое, что и KNN, **с минимальными затратами ресурсов**, но с некоторыми **потерями**: **трейдоффом между точностью** индекса, который строит алгоритм под капотом, **и задержкой**.

> Резюме

1. Ранжирование может применяться в широком спектре задач.
2. Под задачей матчинга в рамках модуля мы подразумеваем соотнесение предложений о продаже с базой документов (SKU/моделей).
3. Задача матчинга — подзадача ранжирования, тесно с ней связана.
4. В матчинге в конце стоит классификатор. Для быстрого поиска в наборе данных нужно использовать эвристики и приближённое вычисление соседей.

КУПЛЕНО НА
SKLADCHIK.COM