1 Введение

1.1 Основные понятия

Основная единица данных, рассматривающаяся в работе – элемент одежды, далее будем называть его *объектом* или *элементом*, множество всех рассматриваемых объектов – $\mathcal{X}$.

Каждый объект $X \in \mathcal{X}$ представим как пару $X = (I, T)$ соответственно изображения объекта (может отсутствовать) и его текстового описания(может быть пустым). Пусть $\mathcal{I}$ – множество изображений объектов.

Введем также множество категорий $\mathcal{C}$, $|\mathcal{C}| = n_c$ элементов одежды, дальше называемых *категориями*.
Каждый объект принадлежит некоторой категории $C_i \in \mathcal{C}, i = \overline{1, n_c}$.
Категорию объекта $X \in \mathcal{X}$ будем обозначать C_X.
Множество всех объектов категории C_i — $\mathcal{X}_{C_i}$

Некоторое подмножество $O = \{X_i\}_{i=1}^{k} \subset \mathcal{X}$ множества всех элементов будем называть *образом*, если

1. $O \neq \{\emptyset\}$
2. $|O| \leqslant K$
3. $\forall X_i, X_j \in O, i \neq j \longrightarrow C_{X_i} \neq C_{X_j}$

где K – определяемая задачей константа. Множество всех образов обозначим $\mathcal{O}$. Стоит заметить, что из такого определения следует, в частности:

$$O \in \mathcal{O}, O' \subset O \longrightarrow O' \in \mathcal{O}$$

Для элементов и образов будем рассматривать *функции близости*

$$S_X : \ \mathcal{X} \times \mathcal{X} \longrightarrow [-1, 1], \ \ \forall X \in \mathcal{X} \ S_X(X, X) = 1$$

$$S_O : \ \mathcal{O} \times \mathcal{O} \longrightarrow [-1, 1], \ \ \forall O \in \mathcal{O} \ S_O(O, O) = 1$$

Такой функцией может выступать например косинусное сходство в некотором латентном пространстве.

Для оценки образов введем функцию *оценки* или *совместимости* его элементов:

$$\mathcal{S} : \ 2^{\mathcal{X}} \longrightarrow [0, 1]$$

причем выполнено следующее:

$$\forall O \in \mathcal{O} : \ \mathcal{S}(O) > 0$$
$$\forall O' \in 2^{\mathcal{X}} \setminus \mathcal{O} : \mathcal{S}(O') = 0$$

Совместимостью или *оценкой совместимости* образа O будем называть результат применения функции совместимости к этому образу $\mathcal{S}(O)$

1.2 Описание рассматриваемых задач

1.2.1 Оценка образа

Выше введена функция $\mathcal{S}$, ассоциирующая с каждым образом его оценку совместимости, однако вид такой функции неизвестен. Задача оценки образа – это классическая задача регрессии:

- **Дано:**
  $\{O_1 \dots O_n\} \subset \mathcal{O}$
  $\{\mathcal{S}(O_1) \dots \mathcal{S}(O_n)\}$

- **Требуется:**
  Найти наилучшую в некотором смысле аппроксимацию функции $\mathcal{S}$ функциями заданного класса, т.е. решить задачу оптимизации:

$$\hat{\mathcal{S}} = \underset{S \in \mathscr{S}}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathcal{S}(O_i), S(O_i)) \right]$$

  где $\mathcal{L}(\cdot, \cdot)$ некоторая метрика, например евклидова, а $\mathscr{S}$ – рассматриваемое множество функций, например нейросети заданной архитектуры.

1.2.2 Описание образа

Задача описания образа есть задача построения наилучшего текстового описания для данного образа по изображениям его элементов. Полагая функцию оценки известной получаем следующую формальную постановку:

- **Дано:**
  $\{X_i\} = O_n,\ |O_n| = n$ — образ, где $X_i = (I_i, \varnothing),\ I_i \in \mathcal{I},\ i = \overline{1, n}$ — его элементы с пустым текстовым описанием.
  $O_n(T) = \{X_i(T) = (I_i, T)\}_{i=1}^{n}$ для некоторого общего описания T

- **Требуется:**
  Найти оценку $\hat{T}$ общего для всех элементов описания T, максимизирующую значение функции оценки $\mathcal{S}(O_n)$, т.е.:

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, \mathcal{S}(O_n(T))$$

Стоит заметить, что в данном случае задача генеративная — лучшее описание находится, из решения задачи максимизации функции оценки, а не просто выбирается из заранее заданного конечного множества.

1.2.3 Дополнение (восстановление) образа

Для решения задачи восстановления образа необходимо знать функцию совместимости или ее аппроксимацию. Предполагая ее известной, получаем следующую постановку:

- **Дано:**
  $O_n \in \mathcal{O},\ |O| = n$
  $k \in \mathbb{N},\ k > n$ — количество недостающих элементов
  $J \subset \mathbb{N},\ |J| = m \geqslant k$ — множество индексов категорий недостающих элементов
  $\{\hat{T}_i\}_{i=1}^{k}$ — текстовые представления недостающих элементов, возможно пустые. В случае если предлагается только текстовое описание всего образа T, оно используется в качестве описания каждого элемента, т.е. $\forall i \in \overline{1, k} :\ T_i = T$.

- **Требуется:**
  Найти наилучшее в смысле максимизации функции оценки дополнение образа O_n до $O_k \in$

$\mathcal{O}, |O_k| = k$ элементами из категорий $\{C_j\}_{j \in J}$, т.е. решить следующую задачу:

$$\{\hat{X}_i\}_{i=1}^k = \underset{\{X_i\}_{i=1}^k \in \left(\bigcup\limits_{j \in J} \mathcal{X}_j\right)^k}{\operatorname{argmax}} \left[\alpha \cdot \mathcal{S}\left(O_n \cup \{X_i\}_{i=1}^k\right) + (1-\alpha) \cdot \sum_{i=1}^k S_X((I_i, T_i), (I_i, \hat{T}_i)) \cdot \mathbb{I}\{\hat{T}_i \neq \emptyset\}\right]$$

здесь $X_i = (I_i, T_i)$, $\alpha \in [0, 1]$. Второе слагаемое отвечает за соответствие предсказанного элемента предъявленному текстовому представлению и равно нулю, если представление пусто. Задача, в отличие от предыдущей, дискриминативная и может быть решена точно полным перебором всех объектов из $\mathcal{X}$.

1.2.4 Генерация образа

Задача генерации состоит в выборе образа произвольного размера, наиболее подходящего к предоставленному текстовому описанию. В терминах введенных выше получаем:

- **Дано:**
  T — текстовое описание образа.
- **Требуется:**
  Найти наилучший в смысле максимизации функции оценки образ $O \in \mathcal{O}$ элементы которого наилучшим образом соответствуют предложенному описанию T, т.е.:

$$\hat{O} = \underset{k, O=\{X_i\}_{i=1}^k, O \in \mathcal{O}}{\operatorname{argmax}} \left[\alpha \cdot \mathcal{S}(O) + (1-\alpha) \cdot \sum_{i=1}^k S_X((I_i, T_i), (I_i, T)) \cdot \mathbb{I}\{\hat{T} \neq \emptyset\}\right]$$

здесь $X_i = (I_i, T_i)$, $\alpha \in [0, 1]$. Стоит заметить, что если зафиксировать $k = 1$, получаем обычную задачу поиска наиболее подходящего под описание элемента в коллекции. Учитывая то, что множество образов определено множеством элементов и определением образа, задача генерации образа также является дискриминативной и состоит в переборе всех возможных образов.

1.3 Обзор литературы

1.3.1 Подходы к построению функций близости

При рассмотрении проблемы построения достаточно информативной функции от образов встает вопрос о том какие особенности структуры исходных данных могут быть полезны и на чем можно основывать выбор архитектуры.

Поскольку образ состоит из переменного числа первоначально независимых элементов, естественно перед рассмотрением образа независимо получать некоторое представление каждого его элемента. Кроме того, понятно, что операции с элементами внутри образа должны быть эквивариантны к перестановке элементов, поскольку на них не возникает естественного порядка. Развивая эту идею, можно сказать что каждому элементу присущ некоторый набор признаков, часть из которых может существенно влиять на совместимость, а значит декомпозиция элементов на признаки, моделирование их взаимосвязей а также взаимосвязей элементов между собой может помочь в построении оценки всего образа.

Подобным образом декомпозированная на разные масштабы задача может быть представлена например в виде графа, а значит одним из способов решения будут графовые нейронные сети. Такой подход в контексте оценки и подбора образов рассматривают в частности в статьях «Fashion Retrieval via Graph Reasoning Networks on a Similarity Pyramid» [6] и «Hierarchical Fashion Graph Network for Personalized Outfit Recommendation» [7].

В [6] авторы предлагают GRNet – модель для определения похожести элементов, которая обрабатывает несколько представлений изображения каждого элемента (текстовые описания в статье не рассматривается) в разном масштабе. Т.е. авторы использует еще один промежуточный уровень – масштаб – для построение латентного представления каждого элемента.

В предложенной модели для каждого элемента сначала с помощью GoogLeNet [10] создаются несколько представлений как раз и называемых масштабами, далее несколько вырезанных частей каждого из полученных представлений называются признаками. Считаются локальные векторы близости между соответствующими признаками в соответствующих масштабах, в конце осуществляется message passing [2] по полному графу, вершинами которого являются вычисленные векторы. Таким образом, вместо рассмотрения единого векторного представления, каждый элемент, пользуясь предполагаемой симметрией задачи, декомпозируют на несколько масштабов, а каждый масштаб на несколько признаков и моделируют их взаимодействие.

Оставим за пределами нашего рассмотрения подробности процесса получения признаков, тогда пусть для элементов $X, Y \in \mathcal{X}$, $\{x_l^i \in \mathbb{R}^{C\times 1}\}$ и $\{y_l^i \in \mathbb{R}^{C\times 1}\}$ векторные представления i-го локального признака в l масштабе. Для каждого масштаба l и номеров признаков i и j вектор локальной близости — s_l^{ij}:

$$s_l^{ij} = \frac{P|x_l^i - y_l^j|^2}{\|P|x_l^i - y_l^j|^2\|_2}$$

где P – так называемая проекционная матрица с обучаемыми весами.
Из этих векторов составляется граф, называемый авторами пирамидой. Далее для каждой пары вершин $s_{l_1}^{ij}$ и $s_{l_2}^{mn}$ определяется скаляный вес $w_p^{l_1 ij l_2 mn}$:

$$w_p^{l_1 ij l_2 mn} = \frac{\exp((\mathbf{T}_{out} s_{l_1}^{ij})^\intercal (\mathbf{T}_{in} s_{l_2}^{mn}))}{\sum\limits_{l,p,q} \exp((\mathbf{T}_{out} s_{l_1}^{ij})^\intercal (\mathbf{T}_{in} s_l^{pq}))}$$

где $\mathbf{T}_{in}, \mathbf{T}_{out}$ – обучаемые матрицы. Тогда для $l_1 = l_2$ это веса внутри одного масштаба, а для $l_1 \neq l_2$ – между разными, что позволяет им "сообщаться". Таким образом мы определили граф близости $G = (\mathbb{N}, \mathbb{E})$, где $\mathbb{N} = \{s_l^{ij}\}$ а $\mathbb{E} = \{w_p^{l_1 ij l_2 mn}\}$.
Далее вектора в каждой вершине обновляются по следующему правилу:

$$\widehat{s}_{l_1}^{ij} = ReLU\left(\mathbf{W} \sum_{l_2,m,n} w_p^{l_1 ij l_2 mn} s_{l_2}^{mn}\right)$$

где $\mathbf{W}$ – еще одна обучаемая матрица параметров. Итоговая оценка близости получается в дополнительной вершине графа, которая предварительно инициализируется некоторой простой функцией s_g:

$$s_g = S_g(A(X), A(Y))$$

где $A(.)$ — например average pooling или max pooling по всем признакам во всех масштабах, а S_g — например косинусное сходство.

В [7] авторы используют схожий подход к декомпозиции для задачи рекомендации образов. Рассматривается иерархическая структура из трех уровней — уровня элементов, уровня образов уровня пользователей. Для рассматриваемых пользователей, образов и элементов всех этих образов строится соответствующих трехуровневый граф с ребрами направленными с уровня элементов в уровень образов и из уровня образов в уровень пользователей. Далее, начиная с нижнего уровня, с помощью фактически чуть упрощенного подхода графовых сверточных сетей [5], латентные представления элементов последовательно агрегируются в латентные представления на уровне образов и на уровне пользователей, которые непосредственно используются для рекомендации путем выбора для пользователя с представлением u^* образа $\hat{O}$ по правилу

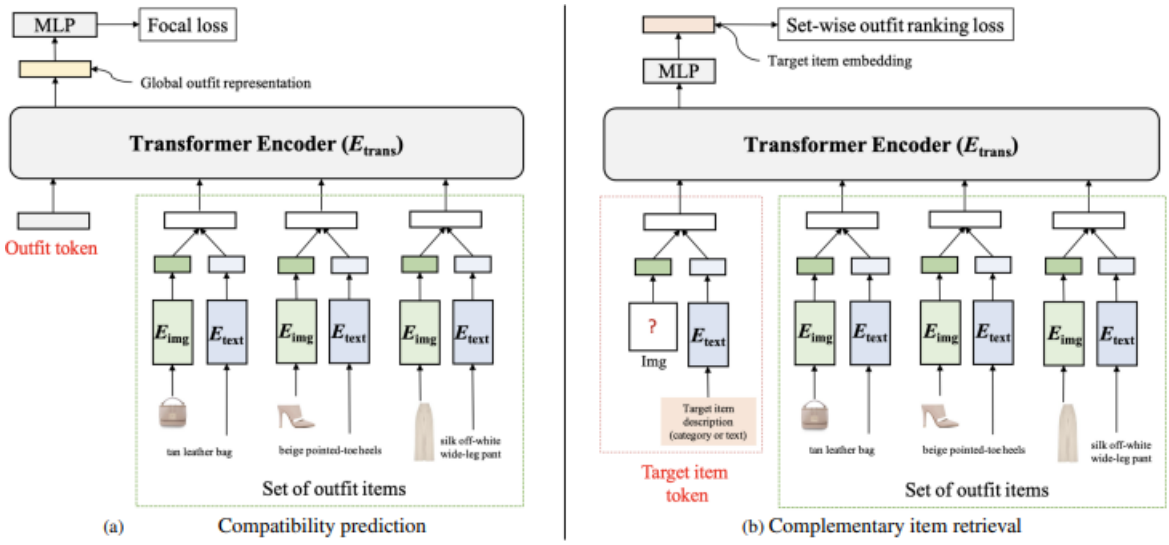
$$\hat{O} = \operatorname*{argmax}_{O\in\mathcal{O}} f(O)^\intercal u^*$$

где $f(O)$ — представление образа, полученное при подсчете в том же графе.

1.3.2 Восстановление, генерация и оценка образов

В задачах оценки, генерации и рекомендации образов, как и во многих других, хорошо показывают себя специальным образом обученные трансформерные [11] архитектуры. Заметим, что в силу отсутствия порядка на элементах и их признаках трансформер без позиционного кодирования [11] фактически эквивалентен graph attention network [12] с несколькими LayerNorm [1] и линейными слоями. Таким образом, при использовании этой архитектуры мы также неявно учитываем рассмотренную выше симметрию данных. Варианты применение трансформеров для задач связанных с образами рассмотрены в частности в статьях [3] и [9].

В [9] авторы предлагают использовать архитектуру трансформера для обучения представления образа целиком. Полученное представление используется для оценки совместимости образа и восстановления образа с пропущенными элементами. В качестве токенов в трансформер подаются латентные представления элементов, т.е. фактически обрабатывается полный граф из всех элементов образа. Помимо изображений элементов, авторы также используют их текстовое описание и одновременно тренируют 2 энкодера для текстового и графического представления каждого объекта. Для задачи восстановления/рекомендации в модель также подается дополнительный токен, инициализированный случайный образом, после прохождения через модель агрегирующий в себе информацию из всех остальных и использующийся в итоге непосредственно для выбора ближайшего в смысле максимизации скалярного произведения дополняющего образ элемента.



(перерисовать картинку из статьи??)

Предложенную модель обучают сначала для оценки совместимости образа, а затем заменяют последние линейные слои и дообучают получать представление образа для подбора подходящих элементов на основании полученного представления и текстового описания нового элемента. Из этого можно заключить, что учет внутренней структуры оказывается полезным и для оценки образа целиком, и для восстановления отдельных его частей, причем для разных задач возможно использовать одни и те же латентные представления элементов.

Идею с общими внутренними представлениями для разных задач развивают авторы [3]. В статье рассматривается вопрос обучения мультимодального трансформера одновременно на несколько задач связанных с образами.

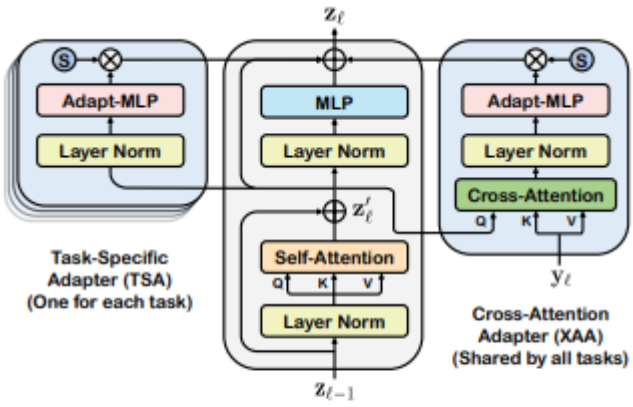


Figure 3. An illustration of a task-versatile Transformer layer equipped with two newly-introduced adapters: cross-attention adapter (XAA) and task-specific adapter (TSA).

Основой архитектуры предложенной модели выступает предтренированный CLIP [8]. Для того, чтобы приспособить модель к выполнению различных задач рекомендации, распознавания и оценки образов авторы предлагают 2 вида адаптеров: TSA – Task-Specific Adapter для обучения специфическим для каждой задачи особенностям и XAA – Cross-Attention Adapter для обеспечения возможности взаимодействиями между различными модальностями, общий для всех задач. Для TSA предлагается ввести дополнительные линейные слои (AdaptMLP) после каждого self-attention блока параллельно с основными:

$$z_l^{tsa} = s \cdot AdaptMLP(LN(z'_l))$$

где s – Обучаемый множитель.
В XAA используется дополнительный Multi-Head Cross Attention (MHXA) с группой линейных слоев после него:

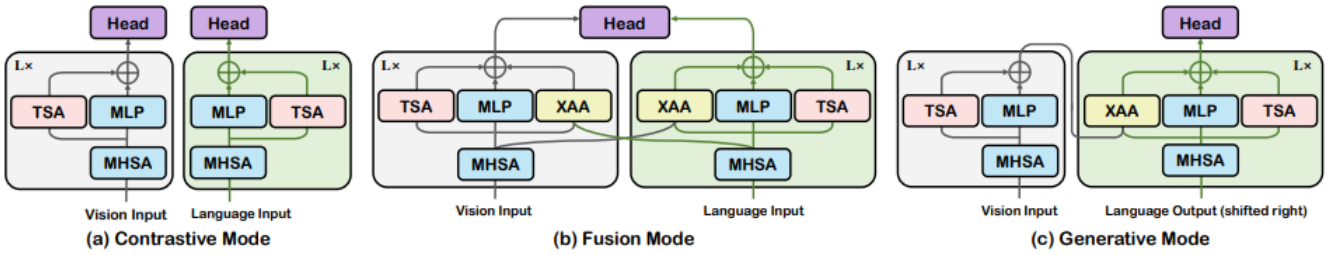
$$z_l^{xaa} = s \cdot AdaptMLP(LN(MHXA(z'_l, y_l)))$$

где y_l выход self-attention слоя части сети для другой модальности.
Далее полученные z_l^{tsa} и z_l^{xaa} аггрегируются с обычным выходом следующим образом:

$$z_l = MLP(LN(z'_l)) + z'_l + z_l^{tsa} + \epsilon \cdot z_l^{xaa},\ \epsilon \in \{0, 1\}$$

ϵ – барьерный множитель включающий или выключающий определенный адаптер для определенной задачи.

Рассматриваемая архитектура обучается на разные задачи в трех режимах Contrastive, Fusion и Generative с различными используемыми адаптерами и функциями потерь.



Contrastive mode:
Этот режим используется для кросс-модальных рекомендаций (XMR) — задачи выбора наиболее подходящего элемента по текстовому описанию и выбора наиболее подходящего описания элемента на изображении среди данных. Все XAA блоки отключены. Обучение производится на выборках элементов $\mathcal{X} \supseteq (\mathbf{I}, \mathbf{T}) = \{(I_1, T_1), \ldots, (I_B, T_B)\}$, сначала части сети для соответствующей модальности по отдельности применяются к элементам каждой пары, формируя 2 итоговых унимодальных представления, а потом с помощью контрастной функции потерь [4] производится максимизация схожести получившихся унимодальных представлений.

$$\mathcal{L}_{XMR} = \frac{1}{2}[\mathcal{L}_{InfoNCE}(\mathbf{T}, \mathbf{I}) + \mathcal{L}_{InfoNCE}(\mathbf{I}, \mathbf{T})]$$

$$\mathcal{L}_{InfoNCE} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(s(X_i, Y_i)/\tau)}{\sum_{j=1}^{B} \exp(s(X_i, Y_j)/\tau)}$$

где τ – обучаемая (???) температура. s – симметричная функция схожести $s(I_i, T_j) = f_\theta^{[c]}(I_i)^T \cdot f_\theta^{[c]}(T_j)$, где $f_\theta^{[c]}$ – собственно нейросеть.

Fusion mode:
Используется для субкатегориального распознавания (SCR)и направляемых текстом рекомендаций (TGIR). И XAA, и TSA блоки включены.
Задача SCR – предсказание подкатегории для данного товара, основываясь на тексте и картинке. Исходя из специфики задачи, к выходу сети дополнительно добавляется классификатор, cross-entropy-loss которого и минимизируется:

$$\mathcal{L}_{SCR} = -\mathbb{E}_{(I,T)\sim D} \log P\left(f_\theta^{[f]}(I, T)\right)$$

Для TGIR –нахождения элементов похожих по изображению на данный и соответствующих текстовому описанию – подход немного другой, поскольку необходимо получить представления отдельно для исходной картинки с текстовым описанием и целевой картинки, поэтому для $(\mathbf{I^r}, \mathbf{T})$ соответственно исходных картинок и текстовых запросов сеть запускается в $fusion$ режиме, а для целевых изображений I^t в $contrastive$ режиме. Далее считается контрастная функция потерь вида

$$\mathcal{L}_{TGIR} = \mathcal{L}_{InfoNCE}((I^r, T), I^t)$$

Generative mode:
Используется, например, для генерации описания к изображениям элементов. TSA блоки включены в обеих модальностях, XAA – только image-to-text. Причем часть обрабатывающая входное изображение используется как энкодер, а вторая – как декодер в авторегрессионном режиме.

1.4 Теоретическая часть

Отличительной особенностью рассмотренных выше подходов является, попытка в том или ином виде моделировать взаимодействие многие ко многим в полном графе из частей внутренней структуры образов (элементов или признаков). Для моделирования взаимодействия между составными частями объекта или образа, нужно, для начала, получить некоторое их векторное представление.

В работе не будем сравнивать между собой способы векторизовать текстовые и графические представления и заранее их зафиксируем. Более того, сконцентрируемся на представлении образа как множества элементов без внутренней структуры и оставим вопрос разбиения объектов на признаки и рассмотрения их взаимодействия за рамками данной работы.

Для получения представлений текстового и графического описания каждого элемента будем использовать предобученную модель «OUTFIT TRANSFORMER???». Кроме того подробнее будем рассматривать именно задачи дополнения и генерации образов. Для решения этих задач, как было сказано выше, необходимо знать функцию совместимости. Оставим восстановление этой функции за рамками текущей работы (ТОЧНО?) и снова будем пользоваться существующим решением – предобученной моделью из [9] (ПОИСКАТЬ ВАРИАНТЫ ЛУЧШЕ И НА ДРУГИХ ДАТАСЕТАХ!), для действия этой модели на образ будем использовать обозначение введенное выше для функции оценки — $\mathcal{S}(\cdot)$.

Для начала рассмотрим задачу дополнения образа. Будем рассматривать случай пустого описания недостающих элементов, заранее не определенных категорий. Таким образом, учитывая постановку определенную ранее получаем задачу следующего вида:

- **Дано:**
  $O_n \in \mathcal{O},\ |O| = n$ — исходный образ
  $k \in \mathbb{N},\ k > n$ — количество недостающих элементов

- **Задача:**

$$\{\hat{X}_i\}_{i=1}^k = \underset{\{X_i\}_{i=1}^k \in \mathcal{X}}{\operatorname{argmax}} \mathcal{S}\left(O_n \cup \{X_i\}_{i=1}^k\right)$$

Для $k > 1$ и большого количества доступных элементов решение задачи полным перебором становится вычислительно невозможным (сложность алгоритма полного перебора $|\mathcal{X}|^k$), поэтому необходимо рассмотреть возможные способы приближенного решения.

В качестве бейзлайна будем рассматривать два следующих жадных алгоритма:

1.
   - Подсчет $\mathcal{S}(O_n \cup X)$ для всех элементов $X \in \mathcal{X}$
   - Выбор элементов $X_i,\ i = \overline{1,k}$ соответствующих k максимальным значениям $\mathcal{S}(O_n \cup X)$
2.
   - Подсчет $\mathcal{S}(O_n \cup X)$ для всех элементов $X \in \mathcal{X}$
   - Выбор элемента X_1 соответствующего максимальному значению $\mathcal{S}(O_n \cup X)$
   - Подсчет $\mathcal{S}(O_n \cup X_0 \cup X)$ для всех элементов $X \in \mathcal{X}$
   - Продолжение последовательных итерации до X_k

(ОФОРМИТЬ АЛГОРИТМЫ НОРМАЛЬНО!)

1.4.1 Дискретный подход

Заметим, что математически задача выбора k наиболее подходящих к образу элементов может быть сведена к задаче поиска наилучшего пути длины k в полном графе, множество вершин которого есть

$$\mathcal{X} \setminus O_n \cup \{X_{init}\},$$

где X_{init} – дополнительная начальная вершина. Для некоторого пути $X_{init}, X_1, X_2 \dots X_j$ можно рассчитать его *оценку* (ПЛОХОЕ СЛОВО, НАДО ПРИДУМАТЬ ЧТО-ТО ЕЩЕ) как оценку образа $O_n \cup \{X_1 \dots X_j\}$ тогда понятно, что поиск пути с максимальной оценкой эквивалентен решению поставленной выше задачи дополнения образа.

Такая переформулировка позволяет напрямую воспользоваться алгоритмами поиска в графе. В частности, перебор всех возможных путей с помощью классических поиска в ширину (BFS) и поиска в глубину (DFS) есть по сути полный перебор возможных дополнений образа, а значит решает задачу точно. Однако, использование этих алгоритмов не улучшает асимптотику, поскольку не решает проблему необходимости полного перебора, а лишь задает его порядок.

Таким образом, для практической вычислимости, необходимо потребовать, чтобы решение задачи поиска в графе также было приближенным. Предлагается воспользоваться широко используемым например в языковых генеративных моделях алгоритмом beam-search. В крайних случаях этот алгоритм либо вырождается в полный перебор, либо становится жадным алгоритмом номер 2 (ГИПЕРССЫЛКА НОРМАЛЬНО?), предложенным выше. (МАТЕМАТИКА??)

1.4.2 Непрерывный подход

Другой возможный поход — научиться генерировать k латентных представлений из непрерывного множества, таких, что будучи подставленными вместо представлений элементов дополнения $\{X_1 \dots X_k\}$ они решают задачу максимизации функции оценки $\mathcal{S}(O_n \cup \{X_1 \dots X_k\})$ в непрерывном пространстве элементов $\mathbb{R}^d$. Далее необходимо лишь выбрать в некотором смысле ближайшие к ним из конечном множестве $\mathcal{X} \subset \mathbb{R}^d$.

Поскольку мы пользуемся предобученной моделью для подсчета функции оценки, нам доступны не только ее значения для любого образа, но и градиент в любой точке. А значит

искомые представления можно получить воспользовавшись одной из модификаций градиентного спуска.

Список литературы

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[2] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[3] Xiao Han, Xiatian Zhu, Licheng Yu, Li Zhang, Yi-Zhe Song, and Tao Xiang. Fame-vil: Multi-tasking vision-language model for heterogeneous fashion tasks, 2023.

[4] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[6] Zhanghui Kuang, Yiming Gao, Guanbin Li, Ping Luo, Yimin Chen, Liang Lin, and Wayne Zhang. Fashion retrieval via graph reasoning networks on a similarity pyramid, 2019.

[7] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. Hierarchical fashion graph network for personalized outfit recommendation, 2020.

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[9] Rohan Sarkar, Navaneeth Bodla, Mariya I. Vasileva, Yen-Liang Lin, Anurag Beniwal, Alan Lu, and Gerard Medioni. Outfittransformer: Learning outfit representations for fashion recommendation, 2022.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[12] Petar Veli?kovi?, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Li?, and Yoshua Bengio. Graph attention networks, 2018.