СОДЕРЖАНИЕ

1 Практическая часть

Abstract

1.1 Цель

С момента своей публикации, модель Трансформера [4] завоевала огромное призвание. Однако у нее есть несколько серьезных проблем, которые усложняют работу с длинными временным последовательностями (LSTF). Последующие исследования предложили различные методы решения данных и связующих проблем (см. Informer [6], Performer [1], Autoformer [5], PatchTST [3], TFT [2] и др.). В данной работе мы сфокусируемся на первых трех.

В данной работе, мы предлагаем заменить слой эмбеддинга в модели Informer компактным двухслойным сверточным блоком с целью повышения эффективности извлечения локальных паттернов, внедрить модуль декомпозиции ряда из Autoformer для явного разделения трендовых и сезонных компонент и заменить ProbSparse-внимание на линейное FAVOR+ из Performer для учёта глобальных зависимостей при низких вычислительных затратах.

1.2 Используемые данные

Прежде чем перейти к методологии нашей работы, рассмотрим модели и понятия, которыми далее будем пользоваться.

1.2.1 Informer

В 2020 году, Zhou et al., опубликовали свою статью «**Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting**», в которой представили новую, основанную на Трансформере [4] модель под названием **Informer** [6].

Informer был создан для решения задачи **Long-sequence time-series forecasting (LSTF)**. Zhou et al. поставили перед собой следующий вопрос: Можем ли мы построить модель, основанную на трансформере, которая

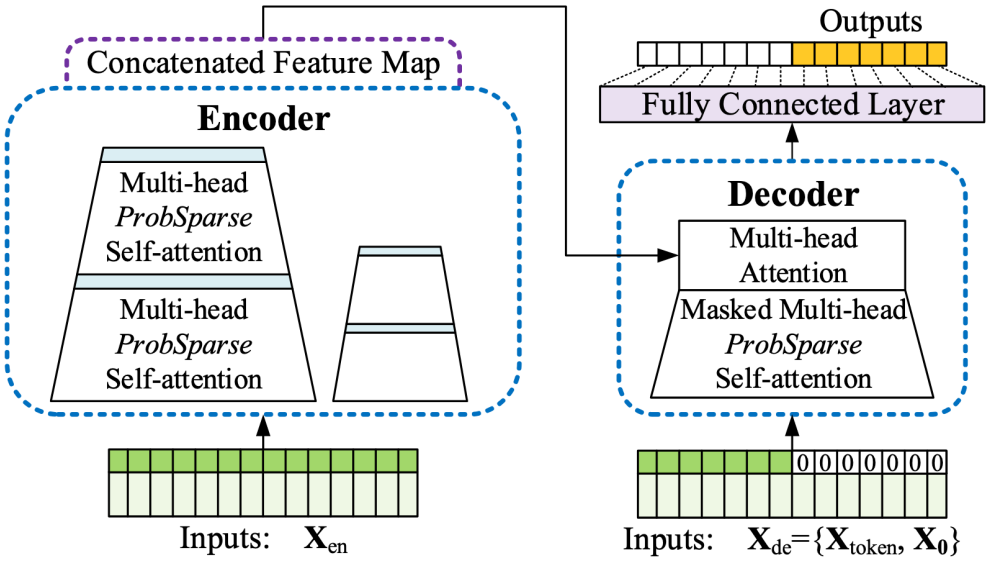

Рис. 1.1: Обзор модели Informer [6].

(a) захватывает очень длинные зависимости

(b) эффективно работает для тысячи временных шагов

Ключевые рассматриваемые слабости трансформера:

- Квадратичная стоимость механизма само-внимания (self-attention) для последовательностей длины L.
- Накладывание слоев умножает эту стоимость, достигая ограничений по памяти.
- Пошаговое (динамичное) декодирование работает медленно и накапливает ошибки.

Информер отвечает на каждый из этих вопросов, реконструируя механизм внимания, кодировщик и декодер.

1.2.1.1 ProbSparse Self-Attention

Зачастую, в длинных временных рядах, большинство скалярных произведений запросов с ключами пренебрежимо малы и лишь некоторые из них достаточно больши. Вместо того, чтобы считать все возможные пары запрос-ключ, informer предлагает следующее:

1. Измерить разброс каждого запроса q_i:

$$M(\boldsymbol{q}_i, \boldsymbol{K}) = \max_j \frac{\boldsymbol{q}_i \boldsymbol{k}_j^T}{\sqrt{d}} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\boldsymbol{q}_i \boldsymbol{k}_j^T}{\sqrt{d}},$$

2. Выбрать u лучших запросов, исходя из $M(\boldsymbol{q}_i, \boldsymbol{K})$, где $u \propto \ln L$.

3. Вычислить полное внимание только для этих u рядов и аппроксимировать остальные через среднее значение.

Что сокращает как время, так и память с $O(L^2)$ до $O(L \log L)$, при этом сохраняя всю важную информацию.

1.2.1.2 Self-Attention Distilling & Кодировщик

Даже после предыдщей операции, каждый слой все равно производит карты признаков длины L, многие из которых повторяют похожие паттерны. Мы можем «дистиллировать» сильнейшие сигналы и сократить последовательность по мере продвижения.

- После каждого блока с механизмом внимания, применяем:

    1. 1-D свертку + ELU активацию,
    2. max-pool со страйдом 2

Что уменьшает размерность в два раза на каждом слое, результируя в пирамиде стэков, чьи выходы в конечном итоге конкатенируют.

Self-attention distilling фокусируется на доминирующих паттернах, при этом сокращая память до $O((2-\varepsilon) L \log L)$.

1.2.1.3 Генеративный декодер

Вместо того, чтобы генерировать генерировать токены по очереди один за одним, заимствуем трюк с начальными токеном из NLP:

- Взять срез ивестной истории (например 5 дней перед целевыми 7ю днями) в качесвте начального токена

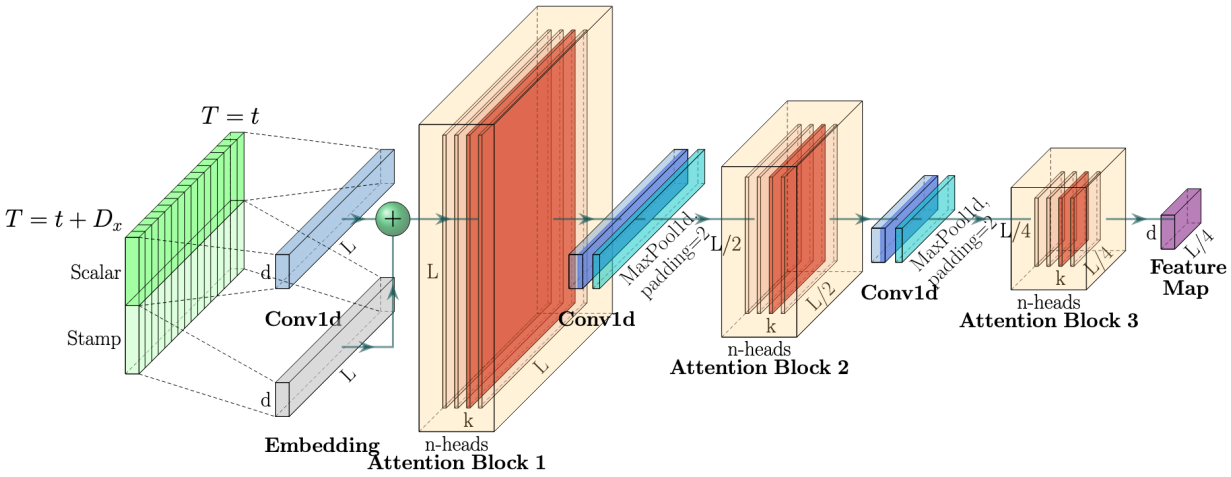

Рис. 1.2: Обзор одного стэка в кодировщике Informer-а [6].

- Разместить плейсхолдеры для всех L_y будущих значение (используя только их временные метки для позиционного контекста)
- За один прямой проход одновременно заполнить все L_y выходов через маскированное ProbSparse внимание.

Что избегает накапливания ошибки и работает гораздо быстрее.

1.2.2 Performer

В 2020 году, Choromanski et al., опубликовали статью «**Rethinking Attention with Performers**», в которой переработали устройство механизма внимания в классических Трансформерах [1].

Как уже было сказано в главе выше, одной из ключевых проблем Трансформера являеется квадратичная сложность расчета внимания: $O(L^2)$. В своей работе, Choromanski et al. задались вопросом:

Можем ли мы добиться той же гибкости глобального внимания, но за линейное время и память?

1.2.2.1 FAVOR+: Fast Attention via positive Orthogonal Random features

Вспомним, что в классическом Трансформере [4] внимание рассчитывается по следующей формуле (без учета нормирующей константы):

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V$$

Заметим, что

$$\exp(q_i^T k_j) = \kappa(q_i, k_j),$$

где κ - softmax ядро.

Согласно «**ядерному трюку**», мы можем представить κ в виде:

$$\kappa(q_i, k_j) = \langle \phi(q_i), \phi(k_j) \rangle,$$

где ϕ - некоторое (возможно, высоко- или бесконечно-мерное) отображение признаков.

Что, в свою очередь, мы можем аппроксимировать согласно **random features**:

$$\kappa(q_i, k_j) \approx \langle \widetilde{\phi}(q_i), \widetilde{\phi}(k_j) \rangle$$

Performer предлагает **FAVOR+**, объединяющий в себе две идеи:

1. **Positive Random Features (PRF)**: Вместо классических тригонометрических random features (которые могут принимать отрицательны значения), предлагается использовать отображение, основанное на экспоненте (таким образом все зачения будут неотрицательными).

2. **Orthogonal Random Features (ORF)**: Выбирать случайные проекции так, чтобы они были взаимно ортогональными, а не независимыми (что уменьшает дисперсию).

Откуда мы получаем выражение для ϕ:

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{m}} \exp\left(\boldsymbol{W}\boldsymbol{x} - \frac{1}{2}||\boldsymbol{x}||^2\right),$$

где $\omega_i \sim \mathrm{N}(0, \boldsymbol{I}_d)$ - случайные ортогональные Гауссовы вектора (строки матрицы $\boldsymbol{W} = [\omega_1^T\ \omega_2^T ... \omega_m^T]^T$), m - кол-во случайных признаков (размерность $\phi(\boldsymbol{x})$).

Обозначив $\Phi_Q = \phi(Q) \in \mathbb{R}^{L \times m}$ и $\Phi_K = \phi(K) \in \mathbb{R}^{m \times d}$, получаем:

$$\text{Attention}(Q, K, V) = \Phi_Q(\Phi_K^T V) \in \mathbb{R}^{L \times d},$$

где расчет Φ_Q и Φ_K занимает $O(Lmd)$, расчет $\Phi_K^T V$ занимает $O(Lmd)$, перемножение с Φ_Q занимает $O(Lmd)$.

Итого имеем линейную сложность для расчета внимания (памяти также требуются только эти три матрицы).

1.2.3 Autoformer

В 2021 году, Wu et al., опубликовали статью «**Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting**», в которой представили новую модель, основанную на Трансформере - Autoformer с целью долгосрочного прогнозирования временных рядов. Auformer объединяет в себе автоматизированную декомпозицию временного ряда и новый механизм внимания, основанный на автокорреляции [5].

В данной работе мы рассмотрим только часть с декомпозицией.

1.2.3.1 Декомпозиция временого ряда

В классическом анализе временных рядов (например декомпозиция по сезонным трендам с помощью LOESS), декомпозиция обычно рассматривается как процесс предобработки. Часто применяется скользящее среднее (здесь и далее под скользящим средним подразумеваем Simple Moving Average (SMA)) для извлечения тренда и сезонности из прошлого ряда, после чего остаток подается на вход модели. Однако после предобработки мы теряем возможность позволить модели уточнить или скорректировать эту декомпозицию по мере обработки более глубоких слоев.

Autoformer предлагает внедрить механизм декомпозиции в саму модель. Таким образом она сможет самостоятельно разделять тренд и сезонные компоненты ряда в каждом слое.

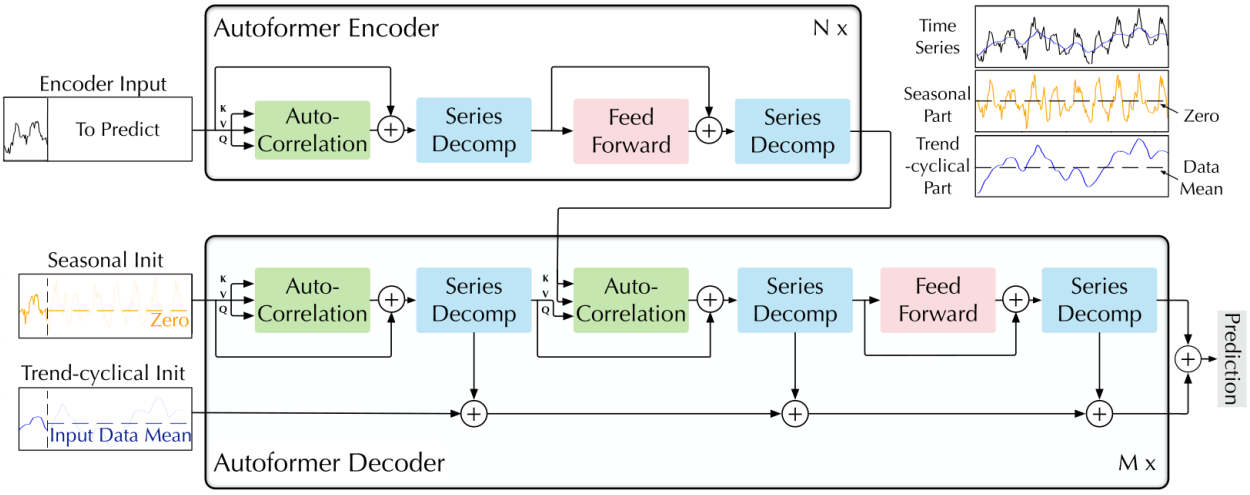

Рис. 1.3: Архитектура Autoformer-а [5].

На каждом слое l берется входная последовательность (которая уже могла быть частитично обработана предыдущими слоями), к которой, внутри слоя, применяется скользящее среднее, что далее разделяется на две части:

- $\text{Trend}^{(l)} = \text{SMA}(\text{Input}^{(l)})$
- $\text{Seasonal}^{(l)} = \text{Input}^{(l)} - \text{Trend}^{(l)}$

После чего мы передаем $\text{Seasonal}^{(l)}$ в наш механизм внимания, а $\text{Trend}^{(l)}$ направляем на вход в следующий слой (via residual connection). Схематично это можно изобразить следующим образом:

Input: $\text{H}^{(l-1)}$

(1) [SeriesDecomp] $\rightarrow \text{Trend}^{(l)}$, $\text{Seasonal}^{(l)}$

where $\text{Trend}^{(l)} = \text{SMA}(\text{H}^{(l-1)})$

$\text{Seasonal}^{(l)} = \text{H}^{(l-1)} - \text{Trend}^{(l)}$

(2) [selft-attention mechanism] on $\text{Seasonal}^{(l)}$

$\rightarrow \text{Seasonal}'^{(l)}$ (+ residual connection, normalization, etc.)

(3) [Feed-Forward] on $\text{Seasonal}'^{(l)} \rightarrow \widetilde{\text{Seasonal}}^{(l)}$

(4) Re-compose $\text{H}^{(l)} = \widetilde{\text{Seasonal}}^{(l)} + \text{Trend}^{(l)}$

1.3 Методология

1.3.1 Повышение эффективности извлечения локальных паттернов

1.3.2 Заимствование механизма внимания из Performer

1.3.3 Внедрение модуля декомпозиции ряда из Autoformer

1.4 Эксперимент

1.4.1 Датасет

1.4.2 Детали

1.4.2.1 Training

1.4.2.2 Baselines

1.4.2.3 Hyper-parameter tuning

1.4.2.4 Setup

1.4.2.5 Metrics

1.4.2.6 Platform

1.5 Ablations

1.6 Результаты

1.7 Заключение

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Krzysztof Choromanski, Valentin Likhosherstov, David Dohan, Xingyou Song, Alex Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In Proceedings of the International Conference on Learning Representations (ICLR), 2020. arXiv:2009.14794.

[2] Bryan Lim, Sercan Ö. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. arXiv:1912.09363.

[3] Wenjie Nie, Di He, Tao Qin, Ming Zhou, and Tie-Yan Liu. A time series is worth 64 words: Long-term forecasting with transformers. In Proceedings of the International Conference on Learning Representations (ICLR), 2023. arXiv:2211.14730.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems (NeurIPS), volume 30, 2017.

[5] Haixu Wu, Yao Xu, Jindong Wang, Guodong Long, Xingquan Jiang, Chengqi Zhang, and Lina Yao. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Advances in Neural Information Processing Systems (NeurIPS), volume 34, pages 22419–22430, 2021.

[6] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 11106–11115, 2020.