Online annex: The attention mechanism – a dynamic weighting scheme

The attention mechanism underpins the transformer architecture of large language models (LLMs) (Vaswani et al (2017)). In a nutshell, it weighs the elements of an input sequence in a way that depends on the context. This is very different from a standard autoregressive model where the coefficients linking past observations to forecasts are fixed. The attention mechanism replaces these fixed coefficients with context-dependent weights, assigning greater weight to what is deemed to be more salient.

The attention mechanism was originally proposed for machine translation. In that context, a single set of weights sufficed to determine which source words mattered for each translated word. Vaswani et al (2017) find that a single attention head was insufficient for longer and more complex texts.[1] They propose running several attention mechanisms in parallel – known as multi-head attention – so that each head can capture a different type of dependency (eg syntactic, semantic or positional). The transformer architecture stacks multiple layers of multi-head attention, with each layer refining the embeddings produced by the previous one.[2]

Formally, a single attention head is achieved via three learned linear projections, termed query, key and value. The query asks, "What information do I need?"; the key signals, "What information do I carry?"; and the value delivers the actual content. For a given element in the sequence (eg the *i*th word), the model computes the vector inner product (ie the dot product) between that element's query vector and the key vectors of all preceding elements. A larger inner product signals a stronger dependency. These raw scores are processed by a function, which normalises them to sum to 1, yielding a valid set of weights. The contextualised representation of the *i*th element is the weighted sum of the value vectors, computed using these data-dependent weights. The result is a dynamic weighting scheme that can capture long-range dependencies and structural breaks without a rigid, predefined lag structure.

This procedure is akin to a kernel regression or a time-varying parameter model, with an important distinction. In conventional time-varying parameter models in econometrics, the weights are estimated from historical data and remain fixed once the estimation step is completed. In contrast, the attention mechanism recomputes the weights for every new input. The model therefore adapts to structural breaks and shifting relationships without re-estimation, which is a valuable property for macroeconomic forecasting, where regimes can change abruptly.

More formally, each input element w_i is first mapped to a numerical vector through a linear projection, $\boldsymbol{x}_i = \boldsymbol{W}\mathrm{w}_i$. From this initial embedding, three vectors are derived through separate linear transformations: the query, $\boldsymbol{q}_i = \boldsymbol{W}^q\boldsymbol{x}_i$, the key,

[1] In the context of transformer neural networks, an attention head refers to the process of computing attention weights and contextualised embeddings. When we refer to multi-headed attention, we mean that several such computations occur in parallel, with each computation utilising a distinct set of weights for the key, query and value matrices.

[2] Alammar (2018) provides a step-by-step visual guide to the transformer, with clear examples and illustrations that offer a useful starting point.

$\boldsymbol{k}_i = \boldsymbol{W}^k \boldsymbol{x}_i$, and the value, $\boldsymbol{v}_i = \boldsymbol{W}^v \boldsymbol{x}_i$. The attention weight that element *j* exerts on element *i* is then:

$$a_{ij}(\boldsymbol{q}_i, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_i) = \frac{\exp(\boldsymbol{q}_i^T \boldsymbol{k}_j)}{\sum_{l=1}^{i} \exp(\boldsymbol{q}_i^T \boldsymbol{k}_l)}.$$

When a_{ij} is large, element *j* carries significant information for element *i*. When a_{ij} is close to zero, element *j* contributes little to *i*th element. The normalising function – the so-called SoftMax – ensures the weights are non-negative and sum to one. The contextualised embedding is then the weighted sum of the value vectors:

$$\boldsymbol{x}_i = \sum_{j=1}^{i} a_{ij}(\boldsymbol{q}_i, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_i)\boldsymbol{v}_j$$

The contextualised embedding thus depends on all preceding elements and defines a single attention head. The three projections divide the workload: the value vector carries the information content, while the key and query vectors determine how much of that content is relevant. Each new input produces a different set of weights and, consequently, a different embedding for every element. This context sensitivity is what gives transformer models their adaptability.

The attention mechanism in MOIRAI (Woo et al (2024)), which underpins BISTRO, follows the procedure described so far, with adaptations for time series data. Consider *R* time series, each with *n* observations: s_{rj}, *i* runs from 1 to *R* and *j* from 1 to *n*. One could draw a direct analogy between individual observations s_{rj} (or equivalently w_i) and individual words. But just as language models process words rather than individual characters, the MOIRAI model groups *p* consecutive observations into a fixed-length patch: $\boldsymbol{p}_{ri} = [s_{r,p*(i-1)+1}, s_{r,p*(i-1)+2}, \ldots, s_{r,p*i}]$. Each patch serves as the minimal unit processed by the transformer, which is analogous to a single word in language modelling. MOIRAI and BISTRO process *p* samples of the time series as a unit, and their predictions are grouped into multiple consecutive samples of size *p*.

The attention mechanism must accommodate two indices. First, each input element $\boldsymbol{p}_{ri}$ is mapped to a numerical vector through a linear projection, $\boldsymbol{x}_{ri} = \boldsymbol{W}\boldsymbol{p}_{ri}$. Next, one computes the query $\boldsymbol{q}_{ri} = \boldsymbol{W}^q \boldsymbol{x}_{ri}$, the key $\boldsymbol{k}_{ri} = \boldsymbol{W}^k \boldsymbol{x}_{ri}$ and the value $\boldsymbol{v}_{ri} = \boldsymbol{W}^v \boldsymbol{x}_{ri}$ vectors. In this notation, *r* indexes the time series and *i* the time stamp. Hence, attention weights have four indices:

$$a_{rs,ij} = \frac{\exp(\boldsymbol{q}_{ri}^T \boldsymbol{R}_{i-j} \boldsymbol{k}_{sj} + u_{r=s}^{(1)} + u_{r\neq s}^{(2)})}{\sum_{o,l} \exp(\boldsymbol{q}_{ri}^T \boldsymbol{R}_{i-l} \boldsymbol{k}_{ol} + u_{r=o}^{(1)} + u_{r\neq o}^{(2)})}$$

The attention mechanism employed by MOIRAI differs from the original attention mechanism of text-oriented LLMs in three key elements:

1. Two scalars were added to distinguish between same-series and cross-series observations: $u_{r=s}^{(1)}$ and $u_{r\neq s}^{(2)}$, where *r* is the index of the query vector and *s* the index for the key vector. These scalars allow the model to weight its own series history differently from cross-series information. Using only two constants, regardless of the number of series, means the model is not constrained in how many series it can process. It also ensures that reordering the series does not change the forecast.

2. $\boldsymbol{R}_{i-j}$ is a rotation matrix that encodes the temporal distance between two patches. The index *i* represents the time index for the query vector, and *j* represents the time index for the key vector. $\boldsymbol{R}_{i-j}$ indicates to the model the

temporal separation between the two observations, which is essential for capturing lead-lag relationships.

3. The normalisation in the denominator runs over both time and series dimensions for the key vector, so that the attention weights sum to one across all patches and all series. This enables the model to draw on both the history of the target variable and the evolution of covariates when forming its prediction.

The new embedding for MOIRAI samples is computed as usual:

$$\boldsymbol{x}_{ri} = \sum_{s,j} a_{rs,ij} \boldsymbol{v}_{sj}$$

The rest of the transformer architecture is analogous to the transformer for text, as explained by Alammar (2018).