Appendix A: Appendix for Proofs

Throughout the appendix we work with fixed-size, padded representations as discussed in Section 4.3. Let $S_{\max}$ and $C_{\max}$ be such that $|\mathcal{S}| \le S_{\max}$ and $|C| \le C_{\max}$. Let $\mathcal{X} \subset \mathbb{R}^d$ denote the (compact) item-feature domain.

Given a state $(C, \mathcal{S})$, choose an arbitrary ordering of the items in $\mathcal{S}$ and form the padded matrix $X^{\mathcal{S}} \in \mathbb{R}^{S_{\max} \times d}$ whose first $|\mathcal{S}|$ rows are the corresponding feature vectors and whose remaining rows are zeros. Let $m^{\mathcal{S}} \in \{0,1\}^{S_{\max}}$ be the mask with $m^{\mathcal{S}}_j = 1$ iff row j of $X^{\mathcal{S}}$ is a real item (not padding). Similarly, form $X^C \in \mathbb{R}^{C_{\max} \times d}$ and $m^C \in \{0,1\}^{C_{\max}}$ for the candidate set.

For the expressiveness proof it is convenient to also mark which assortment rows are currently in C. Define $a^{\mathcal{S}} \in \{0,1\}^{S_{\max}}$ by $a^{\mathcal{S}}_j = 1$ iff the item in row j of $X^{\mathcal{S}}$ lies in C (and $a^{\mathcal{S}}_j = 0$ for already-chosen items and padding). We incorporate masks as ordinary features by augmenting each row:

$$\bar{x}^{\mathcal{S}}_j := (x^{\mathcal{S}}_j,\, a^{\mathcal{S}}_j,\, m^{\mathcal{S}}_j) \in \mathbb{R}^{d+2}, \qquad \bar{x}^C_i := (x^C_i,\, m^C_i,\, m^C_i) \in \mathbb{R}^{d+2},$$

and we write $\bar{X}^{\mathcal{S}}, \bar{X}^C$ for the matrices of augmented rows. Padded rows satisfy $\bar{x} = (0,0,0)$. In the universality construction below we will not rely on additive $-\infty$ masks inside the softmax; instead, we will enforce that *value vectors corresponding to padded rows can be made arbitrarily close to zero* (using the mask bit m), so that including padded rows in attention averages has no effect. Below we first introduce an ancillary lemma for the later proofs.

Lemma 2 (Softmax is Lipschitz) *Let* $\mathrm{softmax} : \mathbb{R}^m \to \Delta^{m-1}$ *be* $\mathrm{softmax}(a)_i = \exp(a_i)/\sum_{j=1}^m \exp(a_j)$. *Then for all* $a, b \in \mathbb{R}^m$,

$$\|\mathrm{softmax}(a) - \mathrm{softmax}(b)\|_\infty \le \tfrac{1}{2}\|a-b\|_\infty, \qquad \|\mathrm{softmax}(a) - \mathrm{softmax}(b)\|_1 \le \tfrac{m}{2}\|a-b\|_\infty.$$

Proof. Fix $a \in \mathbb{R}^m$ and write $s = \mathrm{softmax}(a)$. The Jacobian $J(a) \in \mathbb{R}^{m \times m}$ has entries $J_{ij}(a) = \partial s_i / \partial a_j = s_i(\mathbf{1}\{i = j\} - s_j)$. For each row i,

$$\sum_{j=1}^m |J_{ij}(a)| = s_i(1-s_i) + \sum_{j \ne i} s_i s_j = s_i(1-s_i) + s_i(1-s_i) = 2s_i(1-s_i) \le \tfrac{1}{2},$$

since $0 \le s_i \le 1$ and the function $t \mapsto 2t(1-t)$ is maximized at $t = \frac{1}{2}$ with value $\frac{1}{2}$. Therefore $\|J(a)\|_{\infty \to \infty} = \max_i \sum_j |J_{ij}(a)| \le \frac{1}{2}$. Thus, for some point ξ on the segment between a and b,

$$\|\mathrm{softmax}(a) - \mathrm{softmax}(b)\|_\infty \le \|J(\xi)\|_{\infty \to \infty} \|a-b\|_\infty \le \tfrac{1}{2}\|a-b\|_\infty.$$

Finally, $\|v\|_1 \le m\|v\|_\infty$ for $v \in \mathbb{R}^m$, which yields the ℓ_1 bound. □

The next lemma is a standard equivariant analogue of the DeepSets representation (e.g., Zaheer et al. 2017).

Lemma 3 (Equivariant DeepSets approximation) *Let* $\bar{\mathcal{X}} \subset \mathbb{R}^{d'}$ *be compact and let* $U : \bar{\mathcal{X}}^n \to \mathbb{R}^n$ *be continuous and permutation equivariant. Then for any* $\eta > 0$ *there exist continuous maps* $\phi : \bar{\mathcal{X}} \to \mathbb{R}^p$ *and* $\rho : \bar{\mathcal{X}} \times \mathbb{R}^p \to \mathbb{R}$ *(for some* p*) such that the function* $\tilde{U}$ *defined by*

$$\tilde{U}_i(\bar{X}) := \rho\left(\bar{x}_i,\, \sum_{j=1}^n \phi(\bar{x}_j)\right) \quad (i = 1, \dots, n)$$

approximates U *uniformly:* $\sup_{\bar{X} \in \bar{\mathcal{X}}^n} \|\tilde{U}(\bar{X}) - U(\bar{X})\|_\infty \le \eta$.

Proof. Write $\bar{X} = (\bar{x}_1, \ldots, \bar{x}_n)$ and let $U_1(\bar{X})$ denote the first coordinate of $U(\bar{X})$. By permutation equivariance of U, the scalar function $U_1(\bar{x}_1, \ldots, \bar{x}_n)$ is invariant under permutations of $(\bar{x}_2, \ldots, \bar{x}_n)$.

Introduce a tagged feature space

$$\tilde{\mathcal{X}} := \bar{\mathcal{X}} \times \{0, 1\} \subset \mathbb{R}^{d'+1},$$

which is compact. For $z = (\bar{x}, b) \in \tilde{\mathcal{X}}$ write $\mathrm{feat}(z) = \bar{x}$ and $\mathrm{tag}(z) = b$. Define a function $F : \tilde{\mathcal{X}}^n \to \mathbb{R}$ as follows. Given $(z_1, \ldots, z_n)$, if there is a unique index $i^\star$ with $\mathrm{tag}(z_{i^\star}) = 1$, set

$$F(z_1, \ldots, z_n) := U_1\big(\bar{x}_{i^\star}, \bar{x}_{i_2}, \ldots, \bar{x}_{i_n}\big),$$

where $\{\bar{x}_k\}_{k=1}^n$ are the features $\bar{x}_k = \mathrm{feat}(z_k)$ and $\{i_2, \ldots, i_n\} = \{1, \ldots, n\} \setminus \{i^\star\}$ in any order. (Well-definedness holds because U_1 is invariant to permutations of its last $n-1$ arguments.) If there is not exactly one tag equal to 1, define $F(z_1, \ldots, z_n) := 0$.

Then F is permutation invariant in its n inputs (permuting the tuple does not change the tagged element, and U_1 is symmetric in the remaining elements). Moreover, F is continuous on $\tilde{\mathcal{X}}^n$ since the tag space $\{0, 1\}^n$ is discrete and U_1 is continuous on $\bar{\mathcal{X}}^n$.

By Theorem 9 in Appendix A.2 of Zaheer et al. 2017, for any $\eta > 0$ there exist a dimension p and continuous maps $\tilde{\phi} : \tilde{\mathcal{X}} \to \mathbb{R}^p$ and $\tilde{\rho} : \mathbb{R}^p \to \mathbb{R}$ such that

$$\sup_{(z_1, \ldots, z_n) \in \tilde{\mathcal{X}}^n} \left| F(z_1, \ldots, z_n) - \tilde{\rho}\left(\sum_{j=1}^n \tilde{\phi}(z_j) \right) \right| \le \eta.$$

Define $\phi(\bar{x}) := \tilde{\phi}((\bar{x}, 0))$ and

$$\psi(\bar{x}, s) := \tilde{\rho}\big(\tilde{\phi}((\bar{x}, 1)) + s\big), \qquad (\bar{x}, s) \in \bar{\mathcal{X}} \times \mathbb{R}^p.$$

Evaluating the above bound on the tagged tuple $z_1 = (\bar{x}_1, 1)$ and $z_j = (\bar{x}_j, 0)$ for $j \ge 2$ yields

$$\sup_{\bar{X} \in \bar{\mathcal{X}}^n} \left| U_1(\bar{X}) - \psi\left(\bar{x}_1, \sum_{j=2}^n \phi(\bar{x}_j) \right) \right| \le \eta.$$

Now define

$$\rho(\bar{x}, s) := \psi(\bar{x},\ s - \phi(\bar{x})).$$

Then

$$\rho\left(\bar{x}_1, \sum_{j=1}^n \phi(\bar{x}_j) \right) = \psi\left(\bar{x}_1, \sum_{j=2}^n \phi(\bar{x}_j) \right),$$

so the same uniform η-approximation holds for coordinate 1 with the full sum.

Finally, since both U and $\tilde{U}$ are permutation equivariant, for any $i \in \{1, \ldots, n\}$ and any $\bar{X}$, choose a permutation π that swaps indices 1 and i. Then

$$|U_i(\bar{X}) - \tilde{U}_i(\bar{X})| = |U_1(\pi\bar{X}) - \tilde{U}_1(\pi\bar{X})| \le \eta.$$

Taking the maximum over i and the supremum over $\bar{X}$ gives $\sup_{\bar{X} \in \bar{\mathcal{X}}^n} \|\tilde{U}(\bar{X}) - U(\bar{X})\|_\infty \le \eta$, as claimed. □

A.1. Proof of Lemma 1.

For $i \in C = \mathcal{S} \setminus A$, the event that i is chosen next under a static-utility RUM is

$$\{U_i \geq \max_{j \in C \setminus \{i\}} U_j\}.$$

Conditioning on the history only affects the conditioning event that every already-chosen item $a \in A$ had utility at least as large as the best remaining candidate at the time it was chosen; equivalently, after A has been selected, each $a \in A$ must satisfy $U_a \geq \max_{j \in C} U_j$. This conditioning depends on the history only through the set A (and thus through $(C, \mathcal{S})$), not on the order in which items in A were selected. Therefore,

$$\mathbb{P}(i \mid C, \mathcal{S}) = \mathbb{P}\left(U_i \geq \max_{j \in C \setminus \{i\}} U_j \,\middle|\, U_a \geq \max_{j \in C} U_j \;\; \forall a \in A\right),$$

as stated. □

A.2. Proof of Theorem 2.

Fix $\varepsilon > 0$. Recall $q(C, \mathcal{S}) = |C|$ (or $|C| + 1$ if stop is included) and recall that $q(C, \mathcal{S}) \leq C_{\max} + 1$. Let $u^{C,\mathcal{S}} \in \mathbb{R}^{q(C,\mathcal{S})}$ be the target utility vector and $\mathbb{P}(\cdot \mid C, \mathcal{S}) = \mathrm{softmax}(u^{C,\mathcal{S}})$ be the corresponding choice probabilities (as in (10)).

Let $\tilde{u}^{C,\mathcal{S}}$ be any approximate utility vector of the same dimension. By Lemma 2,

$$\|\mathrm{softmax}(\tilde{u}^{C,\mathcal{S}}) - \mathrm{softmax}(u^{C,\mathcal{S}})\|_1 \leq \frac{q(C,\mathcal{S})}{2} \|\tilde{u}^{C,\mathcal{S}} - u^{C,\mathcal{S}}\|_\infty \leq \frac{C_{\max} + 1}{2} \|\tilde{u}^{C,\mathcal{S}} - u^{C,\mathcal{S}}\|_\infty.$$

Hence it suffices to construct a TCNet whose utilities satisfy $\sup_{(C,\mathcal{S})} \|\tilde{u}^{C,\mathcal{S}} - u^{C,\mathcal{S}}\|_\infty \leq \delta$ with $\delta := 2\varepsilon/(C_{\max} + 1)$.

Under Assumption 2, after padding/masking we may view the utility map as a continuous permutation-equivariant function of the augmented assortment matrix $\bar{X}^{\mathcal{S}}$. Formally, define $n := S_{\max}$ and let $\bar{\mathcal{X}} \subset \mathbb{R}^{d+2}$ be the compact set of all augmented rows (x, a, m) that can arise. The padded utility map $U : \bar{\mathcal{X}}^n \to \mathbb{R}^n$ is continuous and permutation equivariant. Applying Lemma 3 with $\eta = \delta/2$ yields continuous maps $\phi : \bar{\mathcal{X}} \to \mathbb{R}^p$ and $\rho : \bar{\mathcal{X}} \times \mathbb{R}^p \to \mathbb{R}$ such that for every padded state and every index $i \in \{1, \ldots, n\}$,

$$\left| U_i(\bar{X}^{\mathcal{S}}) - \rho\left(\bar{x}_i^{\mathcal{S}}, \sum_{j=1}^{n} \phi(\bar{x}_j^{\mathcal{S}})\right) \right| \leq \delta/2.$$

Restricting to the active candidate rows (those with $a_i^{\mathcal{S}} = 1$) gives an approximation of the desired utility vector $u^{C,\mathcal{S}}$ to within $\delta/2$ in $\|\cdot\|_\infty$.

We now show that a TCNet can approximate the map $\bar{x}_i \mapsto \rho\big(\bar{x}_i, \sum_j \phi(\bar{x}_j)\big)$ uniformly. The argument is constructive and uses only standard ingredients of the TCNet architecture.

(a) Approximate ϕ itemwise on the assortment side. Because ϕ is continuous on a compact set, standard universal approximation for feed-forward networks implies that for any $\gamma > 0$ there exists an MLP (realizable by the FFN sublayers of the assortment encoder) whose output $\widehat{\phi}$ satisfies $\sup_{\bar{x} \in \bar{\mathcal{X}}} \|\widehat{\phi}(\bar{x}) - \phi(\bar{x})\|_\infty \leq \gamma$. We moreover enforce (using the mask bit m) that the value vector can be made arbitrarily small (and very close to zero) on padded rows: define the value representation

$$v(\bar{x}) := m \cdot (0, \widehat{\phi}(\bar{x})) \in \mathbb{R}^{d+2+p}.$$

Then $v(\bar{x}) = 0$ whenever $m = 0$.

(b) Pool $\sum_j \widehat{\phi}(\bar{x}_j^{\mathcal{S}})$ by uniform cross-attention. Consider a single-head cross-attention map with queries from the candidate side and keys/values from the assortment side. Choose parameters so that all attention scores are identically zero: for instance set $W_Q = W_K = 0$ and all associated biases to zero (and do not add an external $-\infty$ mask). Then the attention weights are uniform, i.e., each query attends to each assortment row with weight $1/n$. The attention output for any query is therefore the average $(1/n)\sum_{j=1}^{n} v(\bar{x}_j^{\mathcal{S}})$. Since padded rows have $v(\bar{x}_j^{\mathcal{S}}) = 0$, this average equals $(1/n)\sum_{j:m_j^{\mathcal{S}}=1} v(\bar{x}_j^{\mathcal{S}})$. Finally, a linear map can multiply the second block by the constant $n = S_{\max}$, yielding the pooled vector $\sum_{j=1}^{n}(0, \widehat{\phi}(\bar{x}_j^{\mathcal{S}})) = (0, \sum_{j=1}^{n}\widehat{\phi}(\bar{x}_j^{\mathcal{S}}))$.

(c) Concatenate the candidate row feature with the pooled sum using a residual connection. On the candidate side, use the embedding/FFN sublayers to map each candidate row feature $\bar{x}_i^{C}$ to $h_i := (\bar{x}_i^{C}, 0) \in \mathbb{R}^{d+2+p}$ (i.e., place $\bar{x}_i^{C}$ in the first block and zeros in the pooled block). Adding the pooled vector from (b) through the standard Transformer residual connection produces

$$\tilde{h}_i \approx \Big(\bar{x}_i^{C},\ \sum_{j=1}^{n}\widehat{\phi}(\bar{x}_j^{\mathcal{S}})\Big).$$

(d) Decode ρ. Finally, since ρ is continuous on a compact domain, an MLP (realizable by the utility decoder) can approximate ρ: there exists $\widehat{\rho}$ such that $\sup_{\bar{x},s}\big|\widehat{\rho}(\bar{x},s) - \rho(\bar{x},s)\big| \le \delta/2$ on the relevant compact set. Applying $\widehat{\rho}$ row-wise to $\tilde{h}_i$ yields logits $\tilde{u}_i^{C,\mathcal{S}}$ satisfying $\|\tilde{u}^{C,\mathcal{S}} - u^{C,\mathcal{S}}\|_\infty \le \delta$ for all states $(C,\mathcal{S})$ provided γ above is chosen small enough.

The resulting probability vectors satisfy $\|\tilde{\mathbb{P}}(\cdot \mid C,\mathcal{S}) - \mathbb{P}(\cdot \mid C,\mathcal{S})\|_1 \le \varepsilon$ uniformly over all $(C,\mathcal{S})$, which proves the theorem. □

A.3. Lemma 4 and its proof

Lemma 4 *Let $\ell(u,y) = -u_y + \log\big(\sum_{j=1}^{m} e^{u_j}\big)$ be the multiclass cross-entropy loss for $u \in \mathbb{R}^m$ and label $y \in \{1,\dots,m\}$. Then for all $u, v \in \mathbb{R}^m$ and all y,*

$$|\ell(u,y) - \ell(v,y)| \le 2\|u - v\|_\infty.$$

Proof. Fix y and set $p := \mathrm{softmax}(u) \in \Delta^{m-1}$. Then $\nabla_u \ell(u,y) = p - e_y$, where e_y is the yth standard basis vector. Therefore,

$$\|\nabla_u \ell(u,y)\|_1 = \sum_{j=1}^{m} |p_j - \mathbf{1}\{j = y\}| = (1 - p_y) + \sum_{j \ne y} p_j = 2(1 - p_y) \le 2.$$

Therefore $\ell(\cdot, y)$ is 2-Lipschitz with respect to $\|\cdot\|_\infty$ (duality between ℓ_1 and ℓ_∞), and the claimed inequality follows. □

A.4. Proof of Theorem 3.

Recall the population and empirical cross-entropy risks $\mathcal{L}_{\mathcal{D}}(\theta)$ and $L_{\mathrm{CE}}(\theta)$ from the generalization analysis section, and let $\hat{\theta}$ be any empirical risk minimizer over $\mathcal{F}_M$. Throughout this proof, every masked attention or output softmax is understood in the equivalent restricted-softmax sense: padded coordinates are removed before the softmax is applied, so no $-\infty$ entries appear in the analyzed forward map.

Let $U_{\max} = \sup_{\theta \in \mathcal{F}_M} \sup_{(C,\mathcal{S})} \|u_\theta^{C,\mathcal{S}}\|_\infty$, the forward map $(\theta, \bar{X}^{\mathcal{S}}, \bar{X}^{C}) \mapsto u_\theta^{C,\mathcal{S}}$ is continuous (it is a composition of affine maps, continuous nonlinearities, and softmax operations), and its domain is compact (finite-dimensional, closed, and bounded) because $\mathcal{F}_M$ is a product of closed norm-balls and the input feature domain is compact. Hence $U_{\max} < \infty$. For any state $(C,\mathcal{S})$ and label $i \in C$,

$$\ell(u_\theta^{C,\mathcal{S}}, i) = -u_{\theta,i}^{C,\mathcal{S}} + \log\Big(\sum_{j \in C \cup \{\mathsf{stop}\}} e^{u_{\theta,j}^{C,\mathcal{S}}}\Big) \le U_{\max} + \log\big((C_{\max}+1)e^{U_{\max}}\big) = \log(C_{\max}+1) + 2U_{\max}.$$

Thus the loss is bounded in $[0, B]$ with $B := \log(C_{\max} + 1) + 2U_{\max}$.

Define the loss class $\mathcal{L}_M := \{(C, \mathcal{S}, i) \mapsto \ell(u_\theta^{C,\mathcal{S}}, i) : \theta \in \mathcal{F}_M\}$. A standard symmetrization and McDiarmid's inequality argument (see Shalev-Shwartz and Ben-David (2014)) gives that with probability at least $1 - \delta$ over the sample,

$$\sup_{\theta \in \mathcal{F}_M} \big(\mathcal{L}_{\mathcal{D}}(\theta) - L_{\mathrm{CE}}(\theta)\big) \le 2\,\mathfrak{R}_m(\mathcal{L}_M) + B\sqrt{\frac{\log(1/\delta)}{2m}}.$$

By Lemma 4, for each label i the map $u \mapsto \ell(u, i)$ is 2-Lipschitz in $\|\cdot\|_\infty$. By the vector contraction inequality for Rademacher complexity,

$$\mathfrak{R}_m(\mathcal{L}_M) \le 2\,\mathfrak{R}_m(\mathcal{U}_M),$$

where $\mathcal{U}_M$ is the corresponding logit class and $\mathfrak{R}_m(\mathcal{U}_M)$ is defined in the main text.

Since $\mathcal{L}_{\mathcal{D}}(\hat{\theta}) - L_{\mathrm{CE}}(\hat{\theta}) \le \sup_{\theta \in \mathcal{F}_M}(\mathcal{L}_{\mathcal{D}}(\theta) - L_{\mathrm{CE}}(\theta))$, we have

$$\mathcal{L}_{\mathcal{D}}(\hat{\theta}) \le L_{\mathrm{CE}}(\hat{\theta}) + 4\,\mathfrak{R}_m(\mathcal{U}_M) + B\sqrt{\frac{\log(1/\delta)}{2m}},$$

□

Appendix B: Appendix for Section 6.1

This appendix provides detailed information about the datasets and their preprocessing steps, model implementations, training processes, and hyperparameter tuning methods referenced in §6.1.

B.1. Dataset

	Single Choice								Multiple Choice	
	Car	Sushi	Expedia(book)	Hotel	SFwork	SFshop	Flight	Retail	Bakery	Expedia(click)
# Samples	4,652	5,000	386,558	9,330	5,029	3,157	90,635	17,200	20,000	8,041
# Features	21	7	13	0	0	0	0	0	4	13
Largest Assortment Size	6	10	39	17	6	8	26	21	50	38
Mean Assortment Size	6	10	25.90	10.99	4.38	7.20	10.58	17.20	50	26.85
Largest Chosen Subset Size	1	1	1	1	1	1	1	1	8	10
Mean Chosen Subset Size	1	1	1	1	1	1	1	1	3.55	4.95

Table 3 Summary statistics of datasets.

Table 3 summarizes the datasets used in the experiments and the details are as follows:

• Car: The data is available at http://qed.econ.queensu.ca/jae/2000-v15.5/mcfadden-train/. Each observation contains the single choice from six cars with varying features. We use 21 item features: *Prices divided by* $\log(income)$*, Range, Acceleration, Top speed, Pollution, Size, "Big enough", Luggage space, Operating cost, Station availability, Sports utility vehicle (SUV), Sports car, Station wagon, Truck, Van, Electric Vehicle (EV), Commute ¡ 5 times for EV, College times EV, Constant for Compressed Natural Gas (CNG) ,Constant for methanol, College times methanol*. The details of the features can be found in (McFadden and Train 2000).

• Sushi: The data is available at https://www.kamishima.net/sushi/. We use the *sushi3b.5000.10.order* dataset, which shows the top-10 ranked sushis for each customer. We use these top-10 sushis as the assortment and the top-1 as the final choice. We use 7 item features *Style, Major-group, Minor-group, Heaviness, Frequency-eaten, Normalized-price, Frequency-sold*, and 6 customer features *Gender, Age, Time to fill the survey, Most Long-Lived East/West ID until Age 15, Current East/West ID, Match of Long-Lived and Current Region*. The details of the features can be found in (Kamishima 2003).

• Hotel: The data is available at https://pubsonline.informs.org/doi/abs/10.1287/msom.1080.0231 and from Bodea et al. (2009). It is originally collected from five U.S. properties of a major hotel chain. Each observation contains different room types to be chosen and the final choice. We use the Hotel 1's dataset. We pre-process the data as in Şimşek and Topaloglu (2018), removing purchases without matched room type in the assortment, room types with few purchases (less than 10), and adding an auxiliary no-purchase option (create four no-purchase records for each purchase record).

• SFwork and SFshop: The raw data is available at https://github.com/tfresource/modechoice and originally from Koppelman and Bhat (2006). We use the cleaned version from https://github.com/arjunsesh/cdm-icml. The datasets contain the choices of transportation to commute to shop or work around the San Francisco. Dataset features are not used in our study.

• Flight and Retail: These are two private real datasets without features. The airline data contains the offered choices of offered bundles, i.e., bundle of seat, food etc. The offered assortments are decided by both the selling strategy and remaining seats. The retailing data is collected from a fashion-retailing company, where the items are clothes (from a same category and thus most customers purchase at most one item) and the assortments are decided by the inventories. Both datasets contain records of purchase transactions only, so we add no-purchases following the same way as Hotel dataset.

• Bakery: the data is available at http://users.csc.calpoly.edu/~dekhtyar/466-Spring2018/labs/lab01.html. All observations share the same offered assortment containing 50 items from a bakery and the final (multiple) chosen items. We also encode each item with a unique one-hot encoder as the feature. We use the *20000* folder's dataset with 4 item features: *Flavor, Food Type, Price, Food or Drink*. To construct the sequential choice dataset, we randomly sample one item from the chosen subset as the final choice, and add other unchosen items as the candidates (and the assortment are still all 50 items).

• Expedia: the data is available at https://www.kaggle.com/c/expedia-hotel-recommendations/overview. It contains the ordered list of hotels according to the user's search, and the clicked and final booked (if any) hotel of each search session. We clean the data by dropping the searches with outlier prices ($> \$1,000$) and days of booking in advance (gap between booking date and check-in date > 365) and also the features with missing values. We use 7 item features *Hotel star rating, Hotel chain or not, Location score (by Expedia), Historical prices, Price, Promotion or not, Randomly ranked or not (by Expedia)* and 6 customer features *# nights stay, # days booking in advance,# adults, # children,# rooms,Saturday included or not* . We also use the rankings (in the website) of each item as a feature. To construct the single choice dataset, we choose the booked hotel as the final single choice, and add the no-purchase option if there is no booking. For the multiple choice dataset, we use the sessions with total clicks in the range of 2 to 10, and use clicked hotels as the final choices. For the sequential choice dataset we use the same construction method as in the Bakery dataset.

B.2. Models, Hyperparameters, and Training

• TCNet: Our implementation of the Transformer Choice Net uses (masked) attention to handle variable-cardinality assortments and candidate sets: within each minibatch, we pad $\mathcal{S}$ and C to the maximum sizes in that minibatch (or a fixed cap) and apply an attention mask so that padded rows neither attend to nor are attended by active items. The candidates encoder and assortment encoder each use a single Transformer layer in our experiments (with multi-head attention, layer normalization, residual connections, and dropout as in Vaswani et al. 2017). The utility decoder is a shared item-wise feed-forward layer (no activation) followed by a softmax. For multi-choice prediction with TCNet, we additionally include a virtual stop item during training as discussed in §3.2.

• DeepMNL: We modify the architecture by Wang et al. (2020), Sifringer et al. (2020). We use a neural network g to model the (mean) utility of each item i as a function of its features x_i: $u_i^{\mathcal{S}} = u_i = g(x_i)$ for all $i \in \mathcal{N}$ and $\mathcal{S} \subseteq \mathcal{N}$. The architecture we implement has two layers for the NN. This design is specific to scenarios with observed features; otherwise, it reduces to the MNL model.

• AssortNet: We implement the AssortNet architecture by Wang et al. (2023). In the absence of features, AssortNet uses a neural network g as an assortment encoder, which directly maps the assortment $\mathcal{S}$ into the assortment-specific utilities. Specifically, for any assortment $\mathcal{S}$, the output $g(\mathcal{S}) \in \mathbb{R}^n$ models $u_i^{\mathcal{S}} = g_i(\mathcal{S})$ for $i \in \mathcal{S}$ and $u_i^{\mathcal{S}} = -\infty$ for $i \notin \mathcal{S}$. When features are present, feature encoders are employed to embed these features. The architectural implementation comprises a single layer for the assortment encoder and two layers for the feature encoders (if needed).

• SDANet: We implement the Set-Dependent Aggregation Net (SDANet) by Rosenfeld et al. (2020), which uses two neural networks w and ϕ to model the assortment-specific utilities: $u_i^{\mathcal{S}} = w(\mathcal{S}) \cdot \phi(x_i, \mathcal{S})$. The exact implemented architecture follows the default setting of the posted code. https://drive.google.com/file/d/1KZVbqfVR6QNIpv38y4e8ptzVdbi_GQH4/view.

• FATENet: We implement the First Aggregate Then Evaluate Net (FATENet) by Pfannschmidt et al. (2022), which uses two neural networks U' and ϕ to model the assortment-specific utilities: $u_i^{\mathcal{S}} = U'\left(x_i, \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \phi(x_j)\right)$. The exact implemented architecture follows the default setting of their posted code. https://github.com/kiudee/cs-ranking.

• Mixed-MNL: We train a featured mixed-MNL model (McFadden and Train 2000) by setting the number of customer types the same as the number of item features, and each customer type's utility on item i is a linear function of its features. This follows the setting in (Tomlinson and Benson 2021).

• DLCL: We implement decomposed linear context logit (DLCL) by Tomlinson and Benson (2021), which is an extension of Mixed-MNL such that each customer type's utility on item i is a linear function of both the item i itself and the averaged features over all items in the assortment. The number of customer types is still set by the number of item features, which follows the setting in (Tomlinson and Benson 2021).

B.3. Training Process and Hyperparameters Tuning

We train all models using the Adam optimizer (Kingma and Ba 2014), targeting the minimization of training loss specified in §6.1. The learning rate decays exponentially at a rate of 0.95 every 10 epochs. The initial learning rate is tuned from the values $0.001, 0.0005$, and 0.0001. All trainable parameters are initialized using the uniform Xavier method (Glorot and Bengio 2010). Training has a total of 100 epochs without early stopping, utilizing a mini-batch size of 256. The model with the lowest validation error over the 100 epochs is chosen for testing. Additionally, for models

DeepMNL, AssortNet, SDANet, FATENet, and TCNet, we tune the width of the hidden layers from 32, 128, and 256. For the number of heads in TCNet, we tune it from 4, 8, 16, and 32. For the DeepMNL, AssortNet, SDANet, FATENet, and TCNet in multiple choices prediction task, we tune the threshold defined in (11) from 0.1, 0.3, 0.5, 0.7, 0.9.

Appendix C: Appendix for Section 6.2

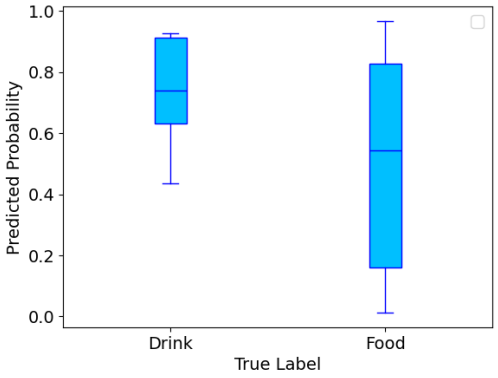

Figure 5 Predicted probability

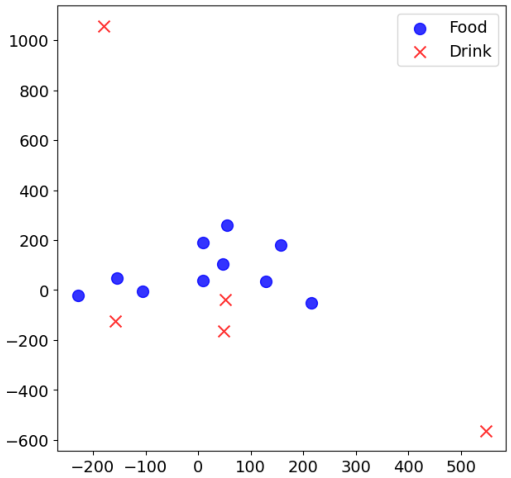

Figure 6 t-SNE visualization of latent features.

Figure 7 Probing the latent representation to identify drink from food.

Experiment details. We train the TCNet with 4 heads on all attention sub-layers, and we choose the initial learning rate as 0.0005 and the hidden dimension as 64. All other hyperparameters, training process, and dataset details are the same as in Appendix B. For training the linear logistic model, we select the cakes Strawberry Cake, Truffle Cake, Napoleon Cake, and Apple Pie (not cake but still more expensive and larger compared to snacks), and the snacks Ganache Cookie, Gongolais Cookie, Raspberry Cookie, and Lemon Cookie as training set. The testing dataset for both Figures 2 and 3 is shown in Figure 3. For training the logistic model in Figure 7, we select the items with indices from 35 to 44 as the training set, where the first 5 items are food and the others are drinks. The testing dataset for Figure 5 contains all other samples, which includes 35 types of food and 5 drinks, and testing dataset for Figure 6 contains the other 5 drinks and the food items with index from 0 to 10 (items information can be found in http://users.csc.calpoly.edu/~dekhtyar/466-Spring2018/labs/lab01.html).

Appendix D: Appendix for Section 6.3

D.1. Experiment Details

We train the TCNet with a single head on all attention sub-layers, and we choose the initial learning rate as 0.001 and the hidden dimension as 32. All other hyperparameters, training process, and dataset details are the same as in Appendix B.

D.2. More Visualization Examples

We use two more examples to showcase how the attention scores in each attention sub-layer help TCNet encode important items.

We construct a sequential choice model for four items $\mathcal{N} = \{A, A', B, L\}$, where B indicates a benchmark item and L indicates the 'leave' option. We assume items A', B and L have constant utility 1 independent of C and $\mathcal{S}$. However, the utility of A can be boosted as follows:

In the first example, the utility of A is equal to 100 if $A' \in C$ and otherwise equals 1. This indicates an (extreme) case that the appearance of A' as a candidate can boost the utility of A. A similar choice behavior can be found in the experiment by (Ariely and Jones 2008).

In the second example, the utility of A is equal to 100 if $A' \in \mathcal{S} \setminus C$ (and otherwise is equal to 1). This indicates an (extreme) case that the utility of A will be boosted when A' has been chosen. This describes the "buy one (A') and get another (A) free" situation.

We train two TCNets based on these two choice models. Figure 12 and 17 output the attention scores of two examples respectively.

Example 1: Figure 12 illustrates the (normalized) attention scores derived from the trained TCNet for two inputs: $(\mathcal{S}_1, C_1)$ and $(\mathcal{S}_2, C_2)$, where $\mathcal{S}_1 = \mathcal{S}_2 = \{A, A', B, L\}, C_1 = \{A, B, L\}, C_2 = \{A, A', B, L\}$. Recall that the attention scores can basically quantify the influence of one item on another. Figure 8 presents the candidates encoder's self-attention score matrix, displaying relatively uniform attention allocated to all three candidates in C_1; In contrast, Figure 9 exhibits a significant increase in attention towards A for both B and L when A' is included among the candidates. This shift can reveal $A' \in C_2$ significantly alters the choice probabilities, unlike in the first scenario where the absence of A' results in fairly balanced attention among the remaining candidates. Figures 10 and 11 represent the candidates encoder's cross attention score matrix. There is a noticeable rise in weights on A for all three items A', B, L as depicted in Figure 11, compared to Figure 10. This elevation in weights demonstrates the substantial impact A has on the choice probability, attributed to its high utility value.

Example 2: Figure 17 shows the attention scores derived from the TCNet again for two inputs same as in Example 1: $(\mathcal{S}_1, C_1)$ and $(\mathcal{S}_2, C_2)$. Figure 13 and Figure 14 display the assortment encoder's self-attention score matrix for two inputs, which are identical since $\mathcal{S}_1 = \mathcal{S}_2$. One observation is the large attention on A' for A, which reveals the potentially large influence of A' since it appears in the assortment (while the assortment encoder is currently unknown whether A' is already chosen or not). Figures 15 and 16 represent the candidates encoder's cross-attention score matrix. There are extreme weights on A for B, L as depicted in Figure 15, compared to Figure 16. This indicates that the large influence of A on the choice probability, due to the fact that $A' \in \mathcal{S}_1 \setminus C_1$, is encoded in 15; While when $A' \notin \mathcal{S}_1 \setminus C_1$, because the (true) choice probabilities are equal among all candidates in Figure 16, the weights are more balanced.

Below are the experiment details:

Training data generation: We generate $24,000$ samples for both training datasets, where all samples' assortments are independently and identically sampled by including each item A, A', B with probability 0.5 and always including L. The candidates are similarly sampled by including the items in assortments with probability 0.5 while always including L. Conditional on the sampled assortment and candidates, the final choice is sampled by the underlying two choice models respectively. TCNet architecture and training process: Both trained TCNets contain a single-layer candidates

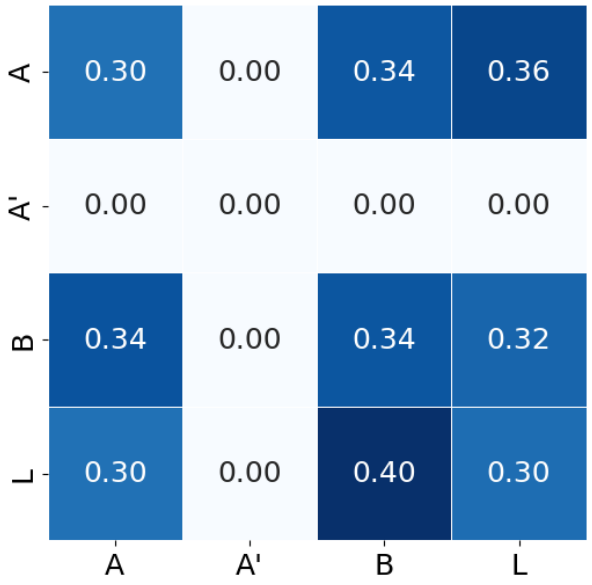

Figure 8 **Candidates attention of $(\mathcal{S}_1, C_1)$.**

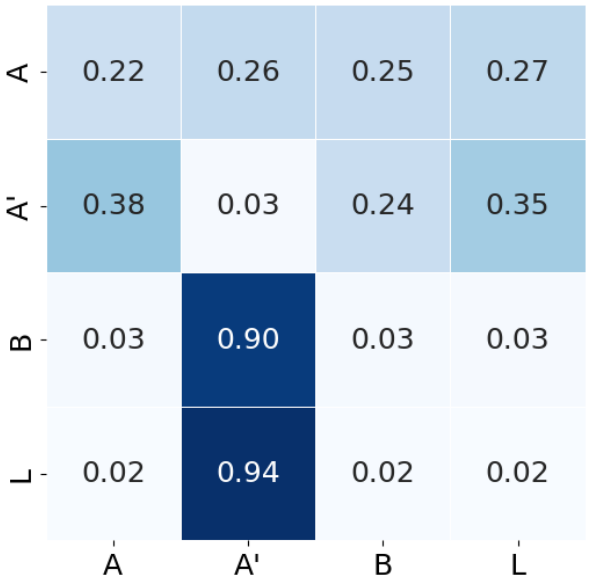

Figure 9 **Candidates attention of $(\mathcal{S}_2, C_2)$.**

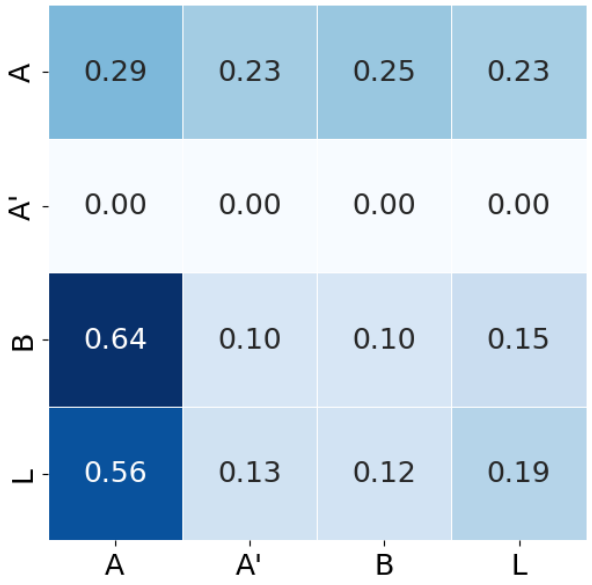

Figure 10 **Cross attention of $(\mathcal{S}_1, C_1)$.**

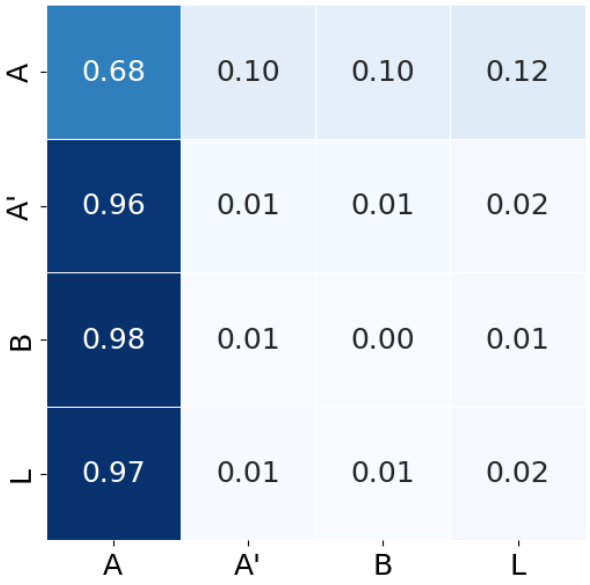

Figure 11 **Cross attention of $(\mathcal{S}_2, C_2)$.**

Figure 12 **First example: candidates encoder's (self) attention and cross attention scores for two inputs $(\mathcal{S}_1, C_1)$ and $(\mathcal{S}_2, C_2)$, where $\mathcal{S}_1 = \mathcal{S}_2 = \{A, A', B, L\}$, $C_1 = \{A, B, L\}$, $C_2 = \{A, A', B, L\}$. Each row indicates the normalized attention scores for the labeled item. If the row's corresponding item is not available, we set its attention scores as 0.**

encoder and an assortment encoder with one embedding sub-layer. The utility decoder also contains a single layer but without an activation function. We employ single-head attention, layer normalization, residual connection, and dropout (with dropout rate 0.1) as described in (Vaswani et al. 2017). The hidden dimensions of all sub-layers (if available) are set as 6. The training process is the same as in Appendix B.3 except there are only 20 training epochs and we apply 0.002 initial learning rate and 0.0005 weight decay on the optimizer.

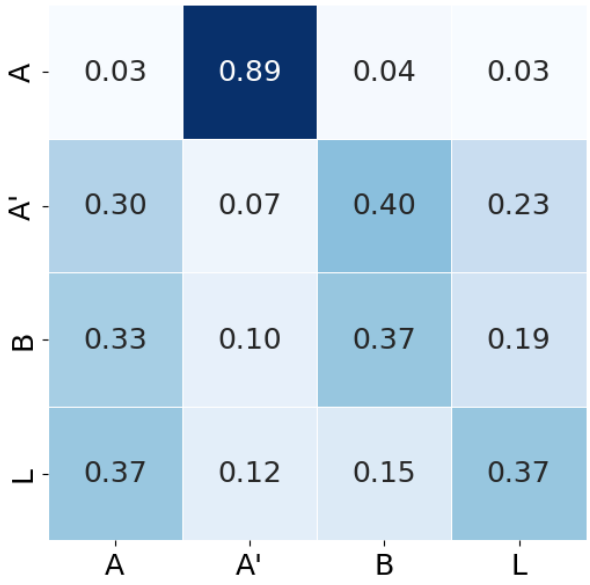

Figure 13 **Assortment attention of $(\mathcal{S}_1, C_1)$.**

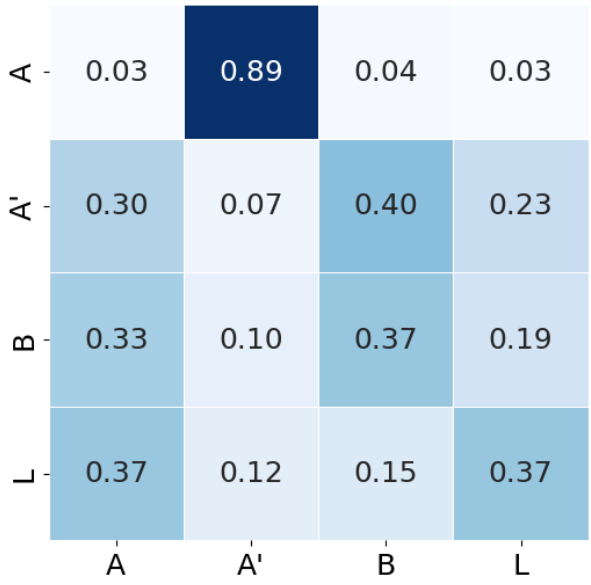

Figure 14 **Assortment attention of $(\mathcal{S}_2, C_2)$.**

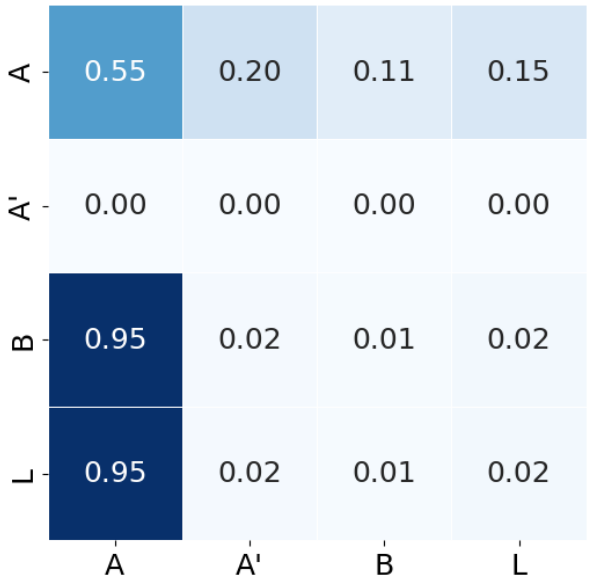

Figure 15 **Cross attention of $(\mathcal{S}_1, C_1)$.**

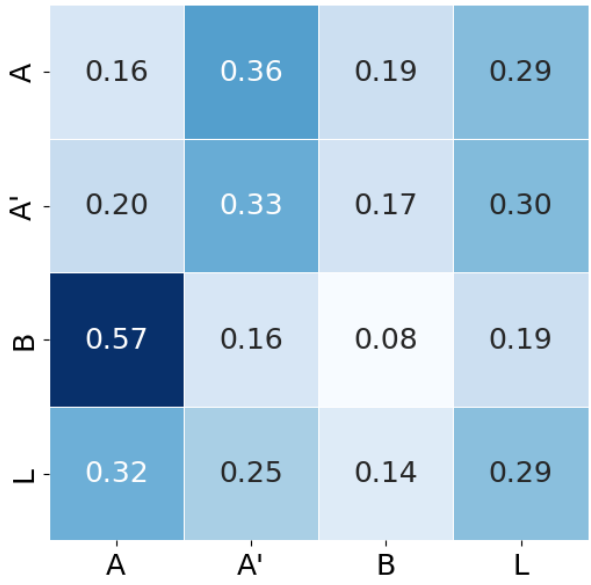

Figure 16 **Cross attention of $(\mathcal{S}_2, C_2)$.**

Figure 17 **Second example: assortment encoder's self-attention and candidates encoder's cross attention scores for two inputs $(\mathcal{S}_1, C_1)$ and $(\mathcal{S}_2, C_2)$, where $\mathcal{S}_1 = \mathcal{S}_2 = \{A, A', B, L\}$, $C_1 = \{A, B, L\}$, $C_2 = \{A, A', B, L\}$. Each row indicates the normalized attention scores for the labeled item. If the row's corresponding item is not available, we set its attention scores as 0.**

Appendix E: Running Time Comparison

We compare the per-epoch training time of TCNet with several deep choice models. Figure 18 reports the time for one epoch of training on 40,000 samples from the Expedia(book) dataset, measured on a MacBook Pro with an Apple M5 chip. We observe that TCNet has a longer per-epoch training time than the other baselines. However, under the

training setup used in our experiments (100 epochs; Appendix B.3), the total training time for TCNet remains within a reasonable time budget (about several minutes).

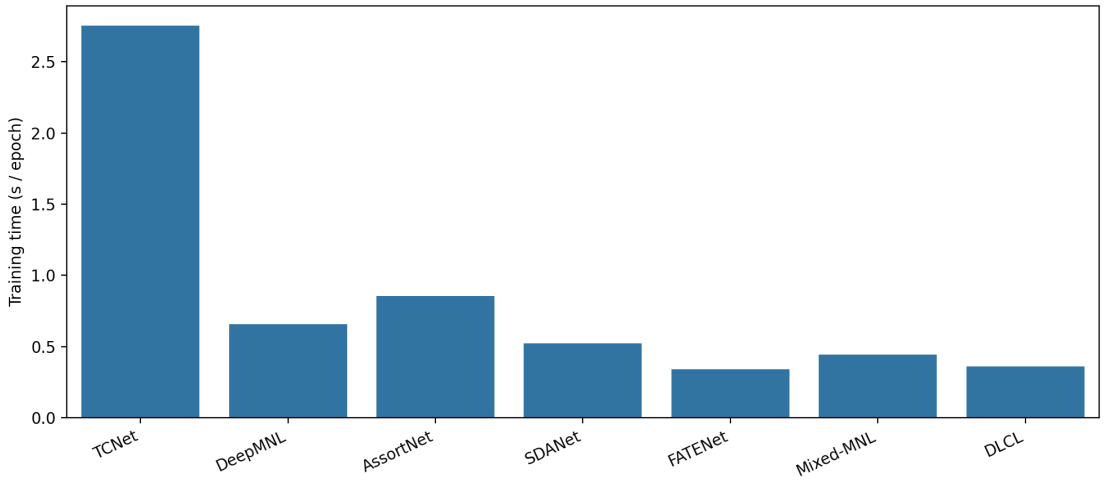

Figure 18 Training time per epoch (seconds) for one epoch of training from the Expedia(book) dataset.