A EXPECTED UTILITY

We leverage the linearity of expectation to simplify the calculation of expected utility. Since the utility of each item is a linear function of the number of occurrences of the item, the expected utility is also a linear function of the probability of each item's occurrence. Specifically,

$$\begin{aligned}
&\mathbb{E}_{\mathbf{X}\sim\boldsymbol{P}}[u(a_i, \boldsymbol{U}, \mathbf{X}_{i,:})] \\
&= \sum_{j=1}^{n} \left(\sum_{(x_1,\ldots,x_n)\in\Omega} x_j \Pr(\mathrm{X}_{i,1} = x_1, \ldots, \mathrm{X}_{i,n} = x_n) \right) U_{i,j} \\
&= \sum_{j=1}^{n} \mathbb{E}_{\mathbf{X}\sim\boldsymbol{P}}[\mathrm{X}_{i,j}] U_{i,j} \\
&= \sum_{j=1}^{n} k P_{i,j} U_{i,j}.
\end{aligned}$$

B DATASETS

Random synthetic data. We generate real-numbered matrices with sizes 10×100, 20×100, 50×100, 100×100, 100×50, 100×20 and 100×10. Each matrix is generated using a truncated normal distribution with lower bound 0., upper bound 1., mean 0.5 and standard deviation 0.1. *Structured synthetic data.* We generate two 20×100 real-numbered matrices with user groups or item groupss. **Item groups (IG)**: for all users, 20% of the items have higher scores in general than the rest of the items. This matrix is generated using two truncated normal distributions, 20% with lower bound 0.2, upper bound 1., mean 0.6, standard deviation 0.05, and 80% with lower bound 0., upper bound 0.6, mean 0.3, standard deviation 0.1. **User groups (UG)**: 20% of the users have in general higher scores towards all the items than their peers. This matrix is generated with the same approach as IG.

Real data. Four real world datasets are used, all output from the same upstream job recommendation model trained with profiles of real job seekers, contents of actual job postings, and behavioral data of those job seekers. **Zhilian**: Scores between 2781 users and 6568 items, sampled from a public dataset for a data mining competition[2] provided by a Chinese online recruitment platform "Zhilian Zhaopin", trained with skills, addresses and interactions. **CareerBuilder**: Scores between 7459 users and 11020 items, obtained from the last day's interaction log from a public dataset[3] provided by "CareerBuilder", trained with addresses and interactions. **VDAB small**: Scores between 1186 users and 8921 items, a random sample of one day's active job seekers and job postings from the private dataset "Vlaamse Dienst voor Arbeidsbemiddeling en Beroepsopleiding"[4], the Flemish labor agency in Belgium, trained with user profiles, job posting contents and interations. **VDAB large**: Scores between 10369 users and 66898 items, a random sample of 6 days' active job seekers and job postings from the same source with the above dataset.

B.1 Zhilian user clustering

We performed Kernel PCA and TSNE on the original score matrix, followed by the clustering method DBSCAN, which gives 22 clusters with various user numbers ranging from 2 to 455. We trained and evaluated the two clusters with the largest user numbers, 455 and 411, respectively.

[2] https://tianchi.aliyun.com/dataset/31623
[3] https://www.kaggle.com/c/job-recommendation
[4] https://www.vdab.be/

C BACKBONE RECOMMENDATION MODEL

The upstream backbone model for predicting the user-item scores of all the real-world datasets is based on [11]. The backbone model training followed the standard practice of temporal split. Hyperparameters were chosen heuristically and tuned based on the validation AUC score.

D IMPLEMENTATION DETAILS

D.1 CA

Let $\boldsymbol{R}$ denote the normalized scores such that $R_{i,j}$ represents the probability of recommending job b_j to job seeker a_i, the optimal recommendation strategy is the solution of the following linear program:

$$\begin{aligned} \arg\max_{\boldsymbol{P}} \quad & \sum_i \sum_j P_{i,j} R_{i,j} - \epsilon P_{i,j} \log P_{i,j} \\ \text{subject to} \quad & \boldsymbol{P} \in \Gamma(m, n), \end{aligned}$$

where $\boldsymbol{P}$ is an $m \times n$ matrix representing an assignment plan relating job seekers to jobs, ϵ is the entropic relaxation term, and $\Gamma(m, n)$ is the set of admissible plans satisfying

$$\Gamma(m, n) := \left\{ \boldsymbol{P}^* \text{ is an } m \times n \text{ matrix} : \forall i, j, P^*_{i,j} \geq 0, \sum_j P^*_{i,j} = \frac{1}{n}, \sum_i P^*_{i,j} = \frac{1}{m} \right\}.$$

We implemented the solution using the Sinkhorn's algorithm base on the code from https://gist.github.com/wohlert/8589045ab544082560cc5f8915cc90bd with different ϵs to achieve different levels of trade-offs and the solutions with less than k positive numbers for any users are discarded as invalid. The values of ϵs for different datasets are shown in Table 3.

Table 3. ϵs for different datasets

Dataset	ϵs
Synthetic VDAB small Zhilian CareerBuilder	0.001, 0.01, 1.9e-02, 2.8e-02, 3.7e-02, 4.6e-02, 5.5e-02, 6.4e-02, 7.3e-02,8.2e-02, 9.1e-02, and evenly spaced values from 0.1 to 5.0 with step size 0.1
VDAB large	1, 0.9, 0.5, 0.3, 0.2, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001

D.2 FEIR

For the synthetic datasets, we use `hardtanh` as the activation function and set the weight of the probability loss to be 1. For the real world datasets, we use `softmax` (with inverse temperature 100) as the activation function and then there is no need for the probability loss. During training, we evaluate the model by deterministic metrics after each training step and save the best parameters with the highest $u(\boldsymbol{a}, \boldsymbol{U}, \boldsymbol{C}) - e(\boldsymbol{a}, \boldsymbol{U}, \boldsymbol{C}) - f(\boldsymbol{a}, \boldsymbol{S}, \boldsymbol{C})$.

The other hyperparameters for each dataset are shown in Table 4.

Table 4. Hyperparameters for different datasets. “sf init t” is the inverse temperature used for initialize the parameters, and “NA” means not appliable. A single combination of weights is an ordered set of four numbers: the weight of envy loss, inferiority loss, utility loss and probability penalty.

Dataset	lr	sf init t	Weights combination	#Sample
Syn 20×100	5e-4	3	Cartesian product of [1, 10, 50], [50, 100, 150, 200], [0,1,2,5], [1]	NA
Syn 100×100	1e-4	4	Cartesian product of [5], [10, 15, 20, 25, 30], [1, 2,3,4,5,7,10], [1]	NA
Syn 100×20	2e-4	2	Cartesian product of [1,5], [1,3,5], [1,2,3,4,5,10], [1]	NA
IG	25e-5, 5e-4	2	[1, 200, 1,1],[1, 150, 1,1],[1, 100, 1,1,],[1, 50, 1,1,], [1, 10, 1, 1],[10, 1, 1, 1],[50, 1, 1, 1],[100, 1, 1, 1], [200, 1, 1, 1],[1, 1, 1,1],[0, 1, 0, 1]	NA
UG	3e-4	1,5	[1,1,1,1], [1,10,0,1], [1,50,1,1], [1,100,1,1], [1,150,1,1],[1,50,0,1], [1,100,0,1], [1,150,0,1], [0,1,0,1]	NA
VDAB small	0.002	1	Cartesian product of [1, 10, 20, 50 ,100], [1, 10, 20, 50, 100], [1, 10, 20, 50, 100] [0]	NA
Zhilian	5e-4	1	Cartesian product of [1], [10, 20, 50, 100, 150, 200, 250], [10, 20, 50, 100, 200], [0]	400
CareerBuilder	5e-4	1	Cartesian product of [1], [1, 10, 20, 50, 100, 150], [1, 10, 20, 50], [0] for k <50 [1], [1,2,3,4,5,6,7,8,9,10], [1, 10, 20, 30], [0] for k=50 [1], [1,5,10,15,20,25], [5,50],[0] for k=100	250
VDAB large	5e-4	1	Cartesian product of [1], [200,300,500,1000,1200,1500,2000,3000], [200,500,1000,1200,1500,2000, 3000], [0]	400 × 1000

E AUXILIARY EXPERIMENTAL RESULTS

E.1 Extra results of the synthetic datasets

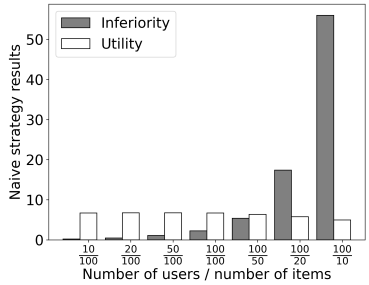
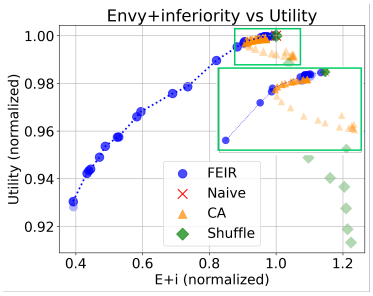
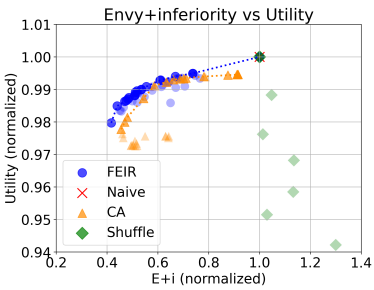


Fig. E.1. Inferiority increases with more users competing for less items under naive recommendation.

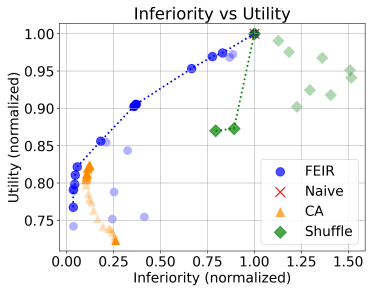


(a) Overall fairness vs utility ($m = n = 100$) with upper right region zoomed in.

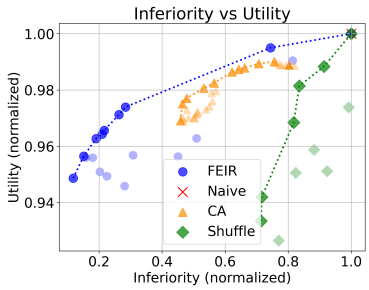


(b) Overall fairness vs utility ($m = 20, n = 100$).

Fig. E.2. Pareto frontiers trading off general fairness with utility for the random synthetic datasets with different user-item ratios ($k = 10$). When the number of items is not greater than the number of users, CA can only cover a small solution region and is dominated by FEIR even in the region they can cover. When the number of items is greater than that of



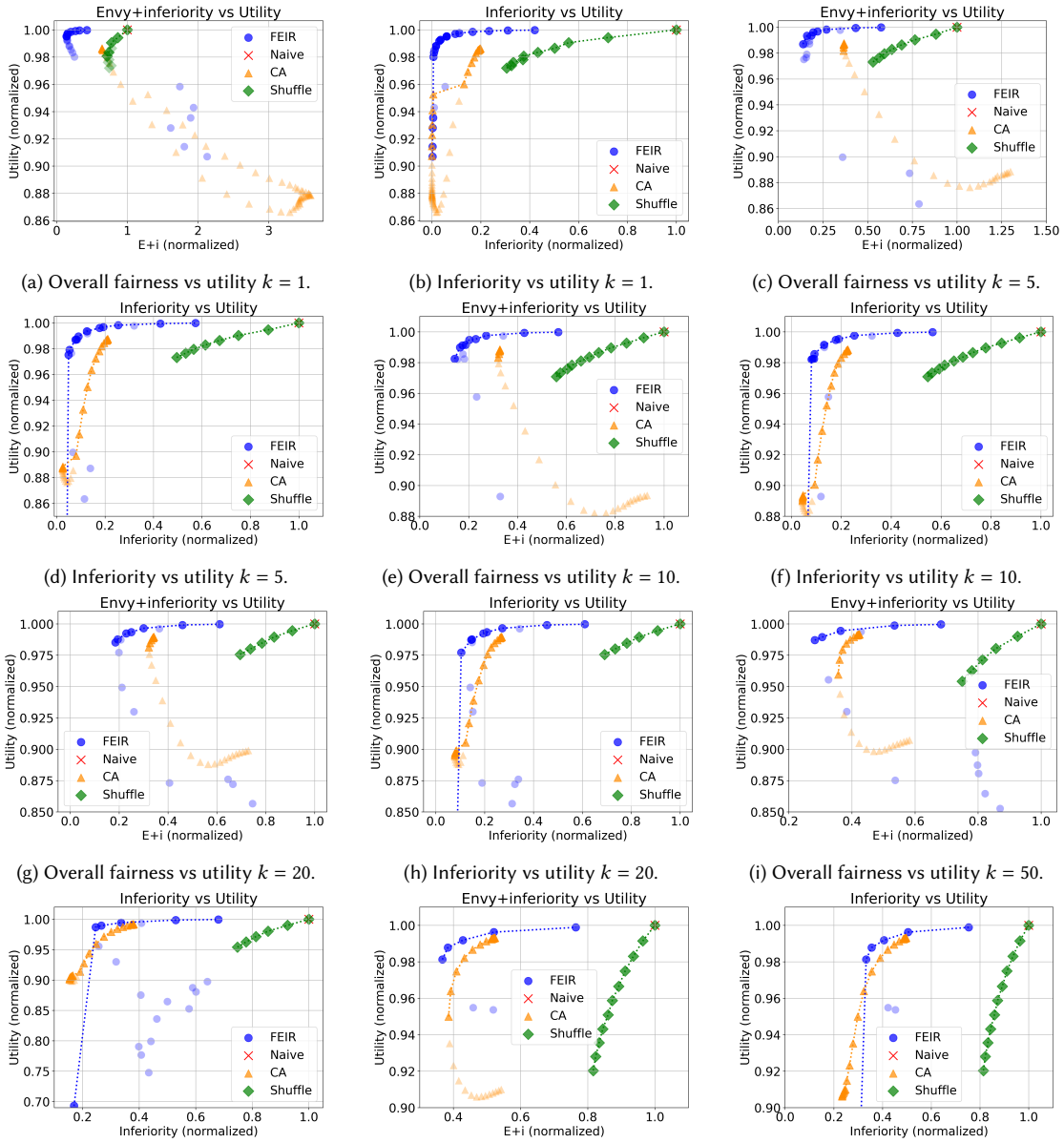
(a) IG: inferiority vs utility.



(b) UG: inferiority vs utility.

Fig. E.3. Pareto frontiers trading off inferiority and utility for the structured synthetic datasets.

E.2 Extra results of the real world datasets



(a) Overall fairness vs utility $k = 1$.
(b) Inferiority vs utility $k = 1$.
(c) Overall fairness vs utility $k = 5$.
(d) Inferiority vs utility $k = 5$.
(e) Overall fairness vs utility $k = 10$.
(f) Inferiority vs utility $k = 10$.
(g) Overall fairness vs utility $k = 20$.
(h) Inferiority vs utility $k = 20$.
(i) Overall fairness vs utility $k = 50$.
(j) Inferiority vs utility $k = 50$.
(k) Overall fairness vs utility $k = 100$.
(l) Inferiority vs utility $k = 100$.

Fig. E.4. CareerBuilder: Pareto frontiers trading-off envy, inferiority and utility.

Table 5. Comparison of the Pareto frontiers trading off fairness metrics with utility for the **VDAB small** dataset with varying ks. The reference point for calculating the HVs being [1, 0.9] (the normalized fairness metric is 1 and the normalized utility 0.9). FEIR is better than CA almost across the board except $\min(i|0.9)$ for $k = 100$. **Better** results are marked **bold**.

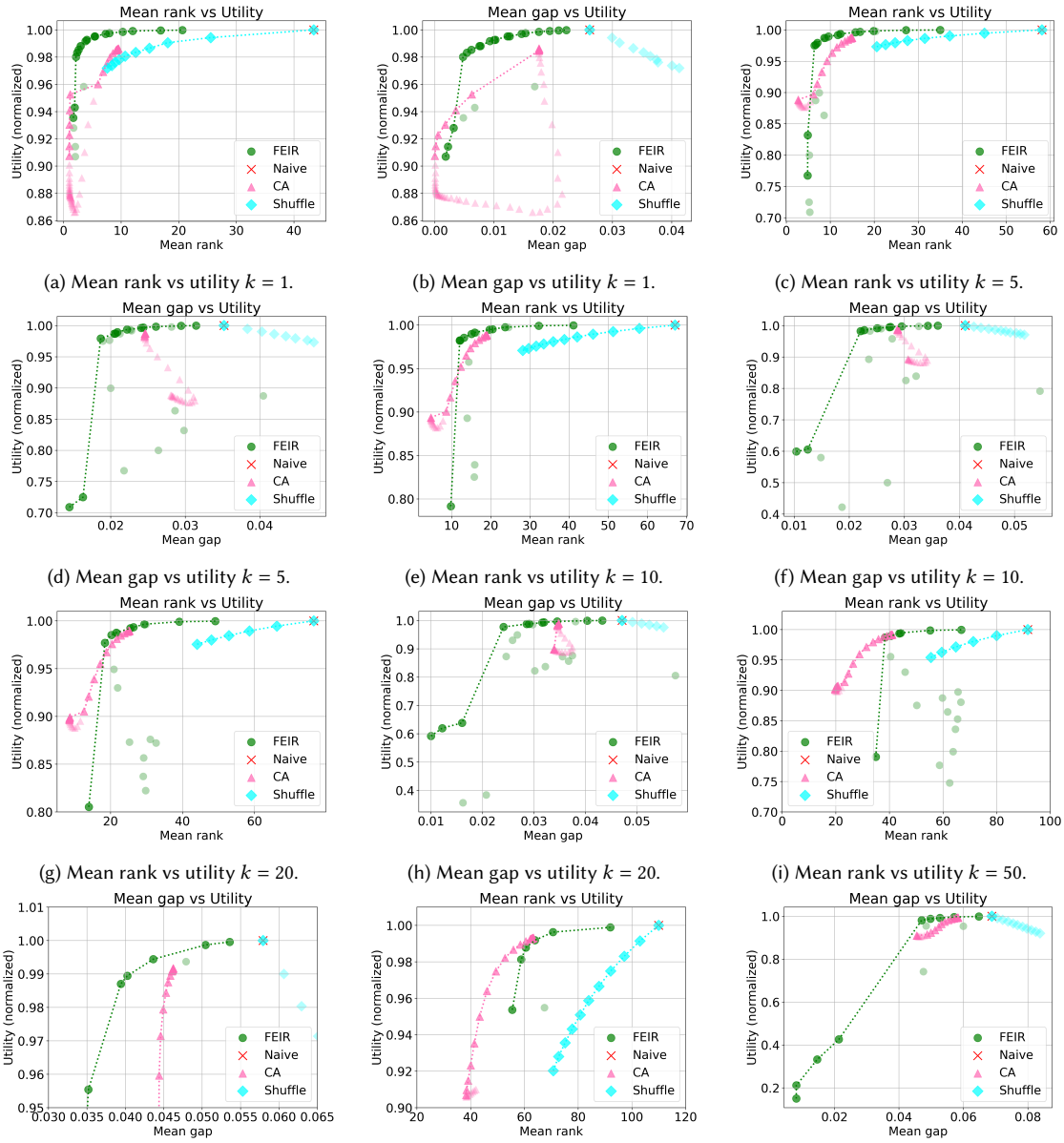
k	HV(g vs u)		HV(i vs u)		$\min(g\|0.9)$		$\min(i\|0.9)$	
	FEIR	CA	FEIR	CA	FEIR	CA	FEIR	CA
1	**0.074**	0.0	**0.097**	0.016	**0.255**	3.454	**0.004**	0.245
5	**0.083**	0.0	**0.091**	0.022	**0.127**	1.120	**0.019**	0.200
10	**0.082**	0.006	**0.088**	0.026	**0.121**	0.798	**0.045**	0.188
20	**0.078**	0.014	**0.083**	0.029	**0.158**	0.624	**0.058**	0.187
50	**0.073**	0.022	**0.076**	0.033	**0.181**	0.475	**0.131**	0.154
100	**0.066**	0.028	**0.069**	0.037	**0.264**	0.397	0.233	**0.16**

Table 6. Comparison of the Pareto frontiers trading off competition metrics with utility for the **CareerBuilder** dataset with varying ks. The reference point for calculating the HV(rank vs u)s is [100, 0.95], for HV(gap vs u)s is [0.07, 0.95]. The better results are marked bold.

k	HV(rank vs u)		HV(gap vs u)		min(rank|0.95)		min(gap|0.95)	
	FEIR	CA	FEIR	CA	FEIR	CA	FEIR	CA
1	**4.818**	3.378	**0.003**	0.002	2.156	**1.157**	**0.005**	0.006
5	**4.527**	3.298	**0.002**	**0.002**	**6.418**	9.277	**0.019**	0.025
10	**4.250**	3.261	**0.002**	**0.002**	**12.075**	12.341	**0.022**	0.029
20	**3.884**	3.156	**0.002**	0.001	18.432	**17.125**	**0.024**	0.035
50	**2.976**	2.790	**0.001**	**0.001**	37.110	**28.798**	**0.038**	0.044
100	1.854	**2.108**	**0.001**	**0.001**	55.491	**46.053**	**0.047**	0.053

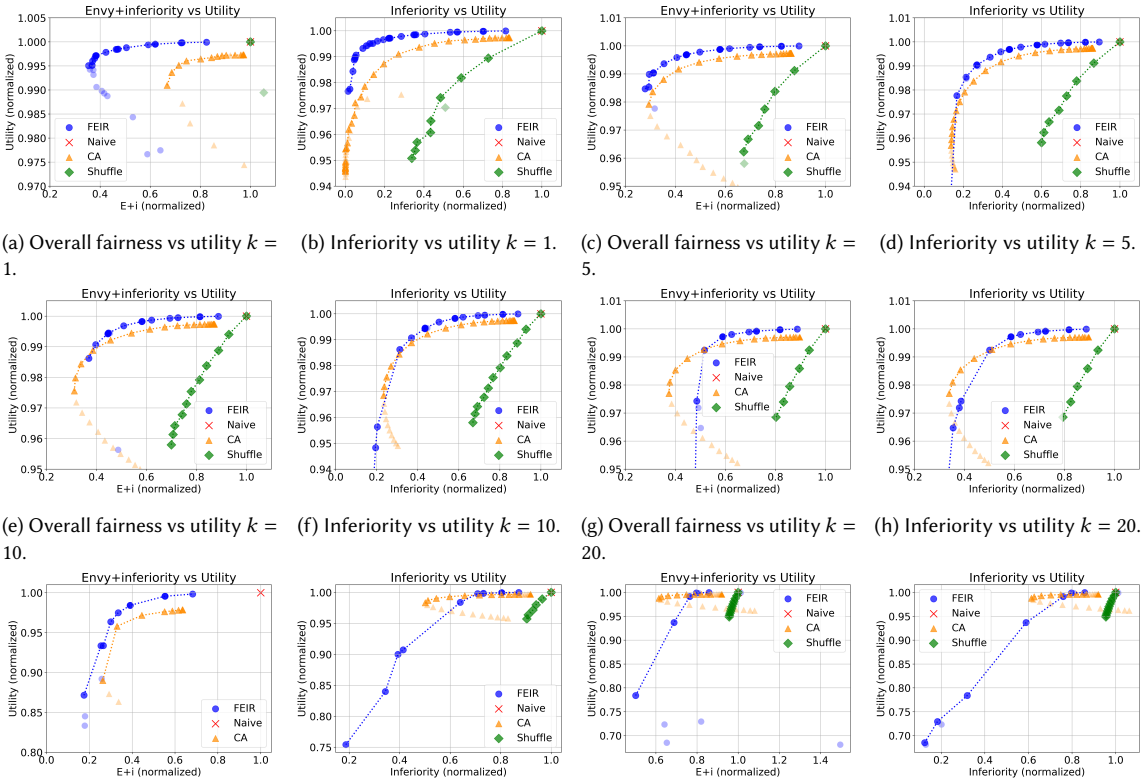
Table 7. Comparison of the Pareto frontiers trading off competition metrics with utility for the **Zhilian** dataset with varying ks. The reference point for calculating the HV(rank vs u)s is [50, 0.95], for HV(gap vs u)s is [0.03, 0.95]. Better results are marked bold.

k	HV(rank vs u)		HV(gap vs u)		min(rank|0.95)		min(gap|0.95)	
	Ours	CA	Ours	CA	Ours	CA	Ours	CA
1	**2.433**	2.303	**0.001**	**0.001**	1.152	**1.001**	0.001	**0.0**
5	**2.356**	2.258	**0.001**	**0.001**	2.661	**1.781**	**0.007**	0.01
10	**2.269**	2.203	**0.001**	**0.001**	4.423	**2.988**	**0.006**	0.011
20	**2.116**	2.084	**0.001**	**0.001**	7.524	5.264	**0.008**	0.01
50	1.743	**1.764**	**0.001**	**0.001**	14.946	**11.934**	**0.011**	**0.011**
100	1.169	**1.239**	**0.001**	**0.001**	26.401	**22.902**	**0.012**	**0.012**



(a) Mean rank vs utility $k = 1$.
(b) Mean gap vs utility $k = 1$.
(c) Mean rank vs utility $k = 5$.
(d) Mean gap vs utility $k = 5$.
(e) Mean rank vs utility $k = 10$.
(f) Mean gap vs utility $k = 10$.
(g) Mean rank vs utility $k = 20$.
(h) Mean gap vs utility $k = 20$.
(i) Mean rank vs utility $k = 50$.
(j) Mean gap vs utility $k = 50$.
(k) Mean rank vs utility $k = 100$.
(l) Mean gap vs utility $k = 100$.

Fig. E.5. CareerBuilder: Pareto frontiers trading-off competition and utility.



(a) Overall fairness vs utility $k = 1$.

(b) Inferiority vs utility $k = 1$.

(c) Overall fairness vs utility $k = 5$.

(d) Inferiority vs utility $k = 5$.

(e) Overall fairness vs utility $k = 10$.

(f) Inferiority vs utility $k = 10$.

(g) Overall fairness vs utility $k = 20$.

(h) Inferiority vs utility $k = 20$.

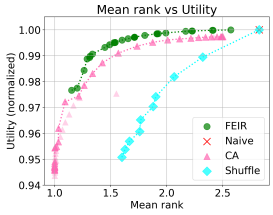
(i) Cluster 1: Overall fairness vs utility $k = 50$.

(j) Inferiority vs utility $k = 50$.

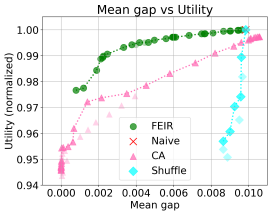
(k) Overall fairness vs utility $k = 100$.

(l) Inferiority vs utility $k = 100$.

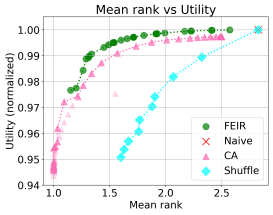
Fig. E.6. Zhilian: Pareto frontiers trading-off envy, inferiority and utility.



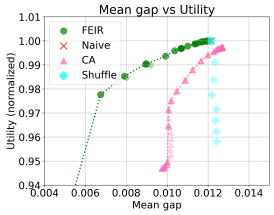
(a) Mean rank vs utility $k = 1$.



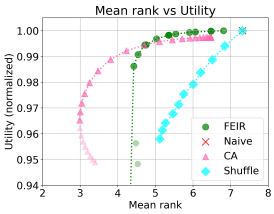
(b) Mean gap vs utility $k = 1$.



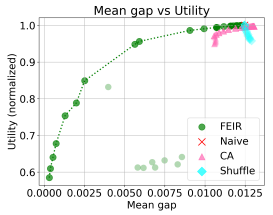
(c) Mean rank vs utility $k = 5$.



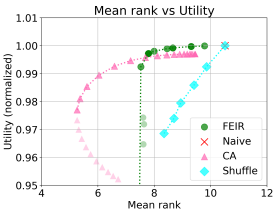
(d) Mean gap vs utility $k = 5$.



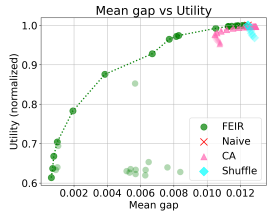
(e) Mean rank vs utility $k = 10$.



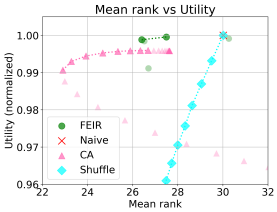
(f) Mean gap vs utility $k = 10$.



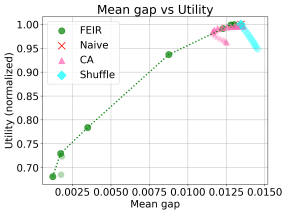
(g) Mean rank vs utility $k = 20$.



(h) Mean gap vs utility $k = 20$.

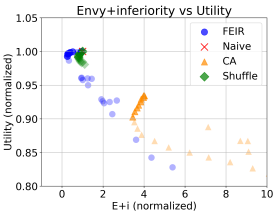


(i) Mean rank vs utility $k = 100$.

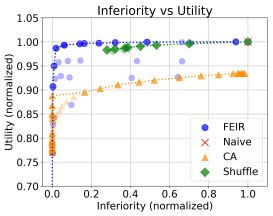


(j) Mean gap vs utility $k = 100$.

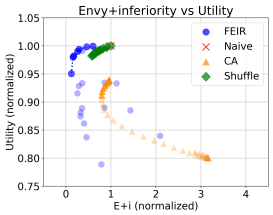
Fig. E.7. Zhilian: Pareto frontiers trading-off competition and utility.



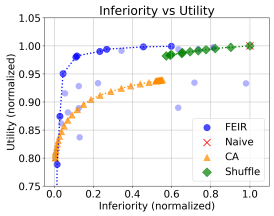
(a) Fairness vs utility k=1.



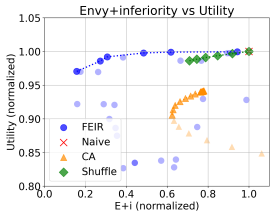
(b) Inferiority vs utility k=1.



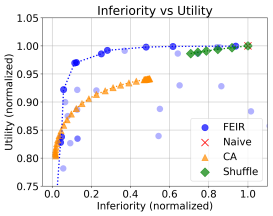
(c) Fairness vs utility k=10.



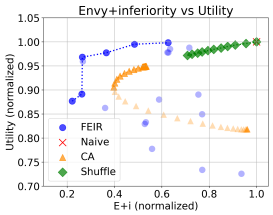
(d) Inferiority vs utility k=10.



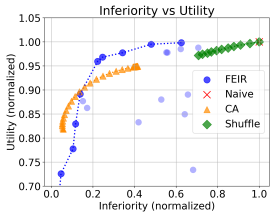
(e) Fairness vs utility k=20.



(f) Inferiority vs utility k=20.

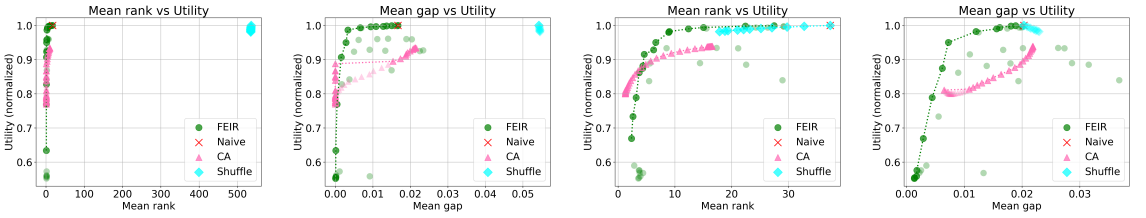


(g) Fairness vs utility k=100.

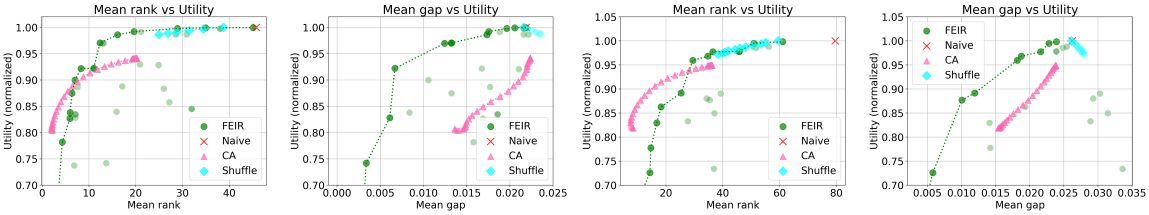


(h) Inferiority vs utility k=100.

Fig. E.8. VDAB small: Pareto frontiers trading-off envy, inferiority and utility.

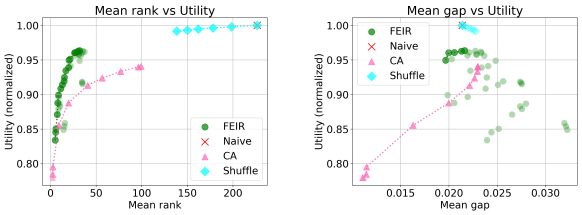


(a) Mean rank vs utility k=1. (b) Mean suitability gap vs utility k=1. (c) Mean rank vs utility k=10. (d) Mean suitability gap vs utility k=10.



(e) Mean rank vs utility k=20. (f) Mean suitability gap vs utility k=20. (g) Mean rank vs utility k=100. (h) Mean suitability gap vs utility k=100.

Fig. E.9. VDAB small: Pareto frontiers trading-off competition and utility.



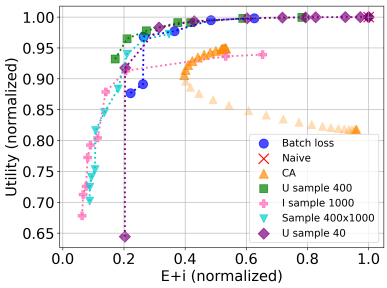
(a) Mean rank vs utility $k = 20$. (b) Mean gap vs utility $k = 20$.

Fig. E.10. VDAB large: Pareto frontiers trading-off competition and utility.

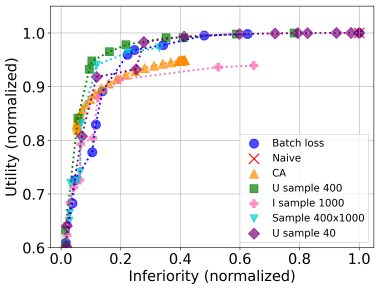
E.3 Scaling-up methods comparison

With respect to the performance (Fig. E.11a and E.11b), all scaling-up methods are close. **User sampling** ("U sample 40" and "U sample 400") seem better at the high utility region, but not good at the low unfairness region. Also, shrinking the sample size does not decrease the performance much. **Item sampling** ("I sample 1000") performs the worst and the reason might be that it destroys the probability distribution which makes the loss estimation faulty and then parameter updates noisy. Surprisingly, **user-item sampling**, i.e., sampling from both sides ("Sample 400×1000") seems better than only sampling from the items, perhaps due to that the noisy updates only affect a smaller number of users at each training step, but a deeper investigation will be future work.

The peak memory usages during training for the VDAB small dataset ($m = 1186$, $n = 8921$, $k = 100$) of different are shown in Table 8 (hardware: NVIDIA GeForce RTX 3090).



(a) VDAB small: General fairness vs utility comparing different scaling-up methods. Points not on the frontiers are omitted for readability.



(b) VDAB small: Inferiority vs utility comparing different scaling-up methods. Points not on the frontiers are omitted for readability.

Method	Memory
Mini batch 128	15.45G
Sample user 400	16.11G
Sample item 1000	15.88G
Sample user 40	0.45G
Sample 400 × 1000	1.93G

Table 8. Peak training memory usage of different methods for the **VDAB small** dataset (1186 × 8921).

E.4 Tuning the loss weights to control the trade-offs.

The desired (or suitable) trade-offs between the fairness notions and utility depend on the the context and stakeholders. In this section we give suggestions by reporting the results of tuning the weight terms.

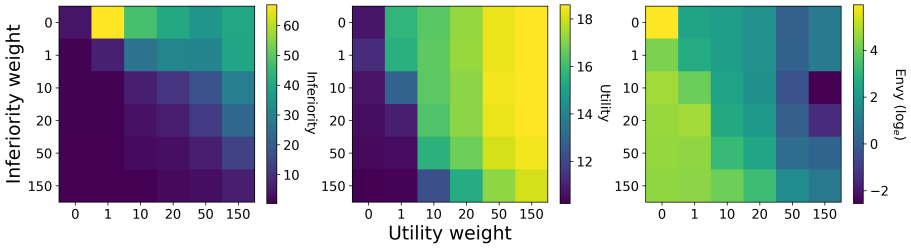
We plot the measurements resulting from different combinations of weights for the VDAB small dataset optimized for the top 20 recommendation. First, when we fix the weight of the envy loss (Fig. E.12b), the greater the utility loss weight, the higher the utility, and the greater the inferiority loss weight, the lower the inferiority. The envy is usually low when the weight of the utility is high, but high when the weight of the inferiority loss is high, showing a trade-off between envy and inferiority as well, but there is more randomness when the other losses are set to 0 or the envy loss

is too high (Fig. E.12e), suggesting a suboptimal solution when optimizing only the envy loss.. When we fix the weight of the utility loss (Fig. E.13b), higher weights of the inferiority loss tend to yield lower inferiority, lower utility, and higher envy. Higher utility weights also decrease envy. However, when the utility weight is high, the weight of the envy loss does not contribute much; high weights of the envy loss sometimes even have opposite effects (Fig. E.13e). Investigating the reasons behind the counter-intuitive results is future work.

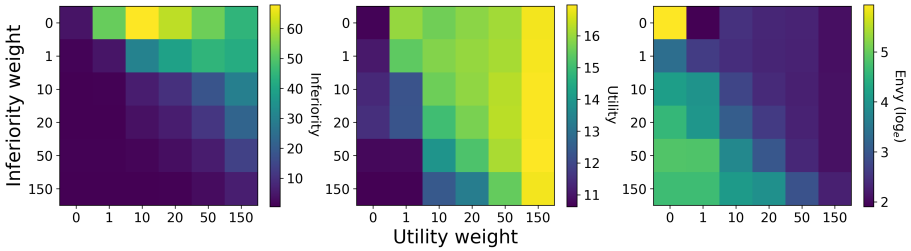
Some heuristic guides to tune the weights. Based on our experience, it is suggested to first find a suitable weight for the envy loss such that the magnitudes of all losses are similar, and then fix this weight for the envy loss and vary the weights of the utility loss and the inferiority loss. Varying the ratio between the utility loss and the inferiority loss tends to give solutions with distinct trade-offs. The reason is that maximizing utility tends to minimize envy, but not vice versa, and the intricate relation between envy and utility makes it tricky to tune the weights of the envy loss and the utility loss at the same time.

E.5 Competition and utility per recommendation with increasing k

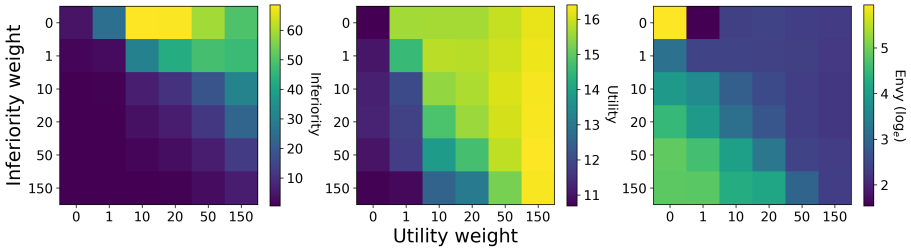
In all real datasets, the inferiority and competition both begin to increase at a fast rate, which slows down as k increases over 20. The utility decreases with a similar rate pattern, as seen in Fig. E.14, E.15, E.16 and E.17.



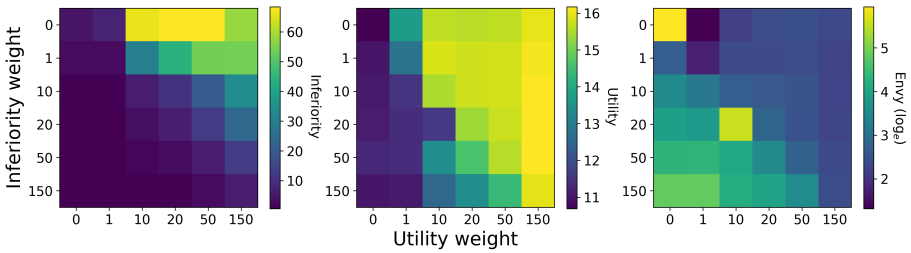
(a) Envy loss weight=1.



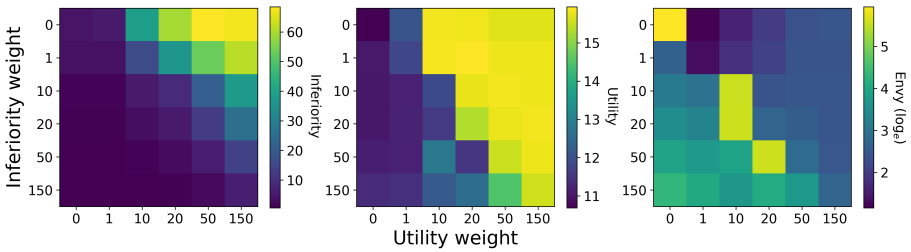
(b) Envy loss weight=10.



(c) Envy loss weight=20.

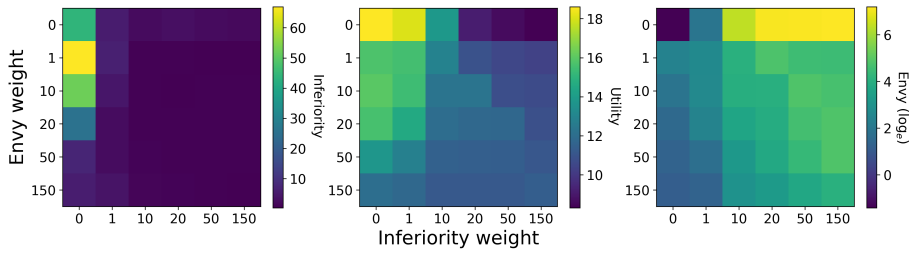


(d) Envy loss weight=50.

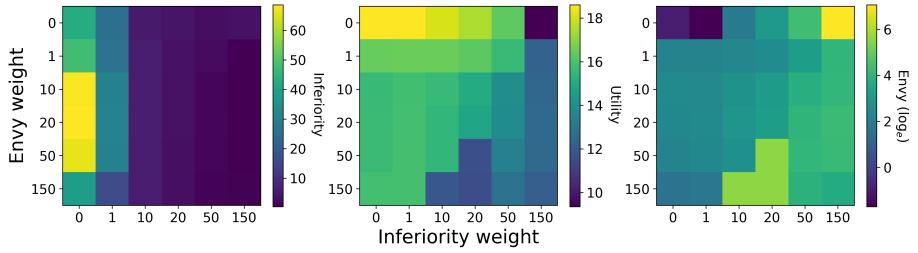


(e) Envy loss weight=150.

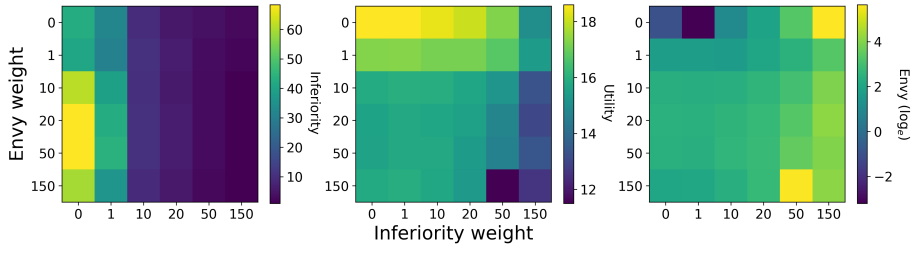
Fig. E.12. Resulting values for inferiority, utility, and envy on the VDAB small dataset when optimizing the combined loss with fixed weights for envy while varying the remaining weights.



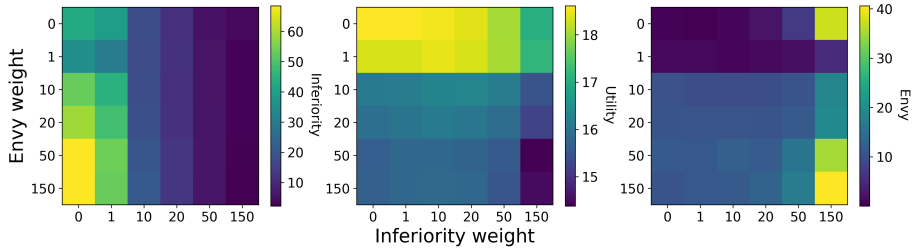
(a) Utility loss weight=1.



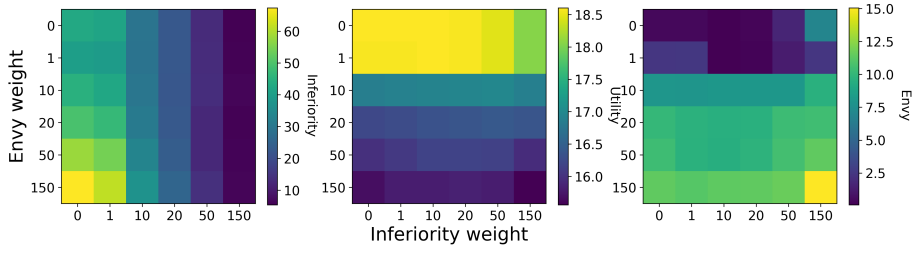
(b) Utility loss weight=10.



(c) Utility loss weight=20.

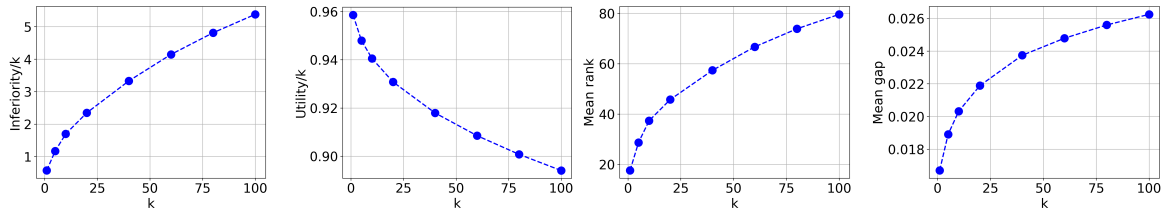


(d) Utility loss weight=50.



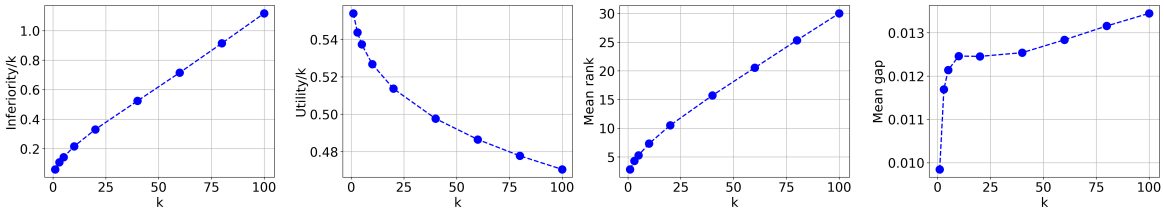
(e) Utility loss weight=150.

Fig. E.13. Resulting values for inferiority, utility, and envy on the VDAB small dataset when optimizing the combined loss with fixed weights for utility while varying the remaining weights.



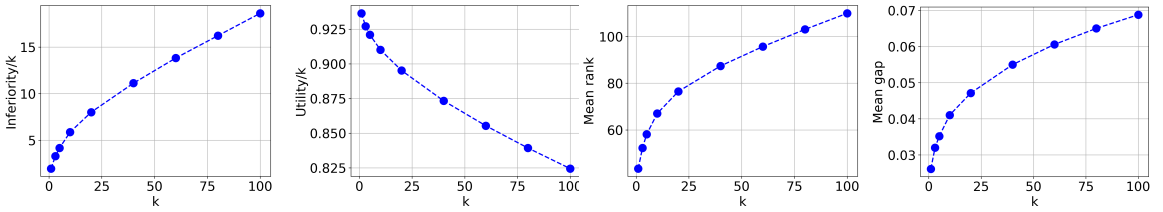
(a) Number of recommended jobs vs inferiority per job.
(b) Number of recommended jobs vs utility per job.
(c) Number of recommended jobs vs mean rank.
(d) Number of recommended jobs vs mean suitability gap.

Fig. E.14. VDAB small: Trends of naive recommendation with increasing k.



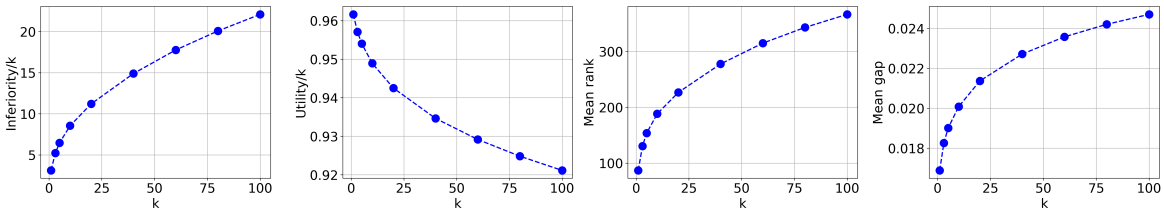
(a) Number of recommended jobs vs inferiority per job.
(b) Number of recommended jobs vs utility per job.
(c) Number of recommended jobs vs mean rank.
(d) Number of recommended jobs vs mean suitability gap.

Fig. E.15. Zhilian: Trends of naive recommendation with increasing k.



(a) Number of recommended jobs vs inferiority per job.
(b) Number of recommended jobs vs utility per job.
(c) Number of recommended jobs vs mean rank.
(d) Number of recommended jobs vs mean suitability gap.

Fig. E.16. CareerBuilder: Trends of naive recommendation with increasing k.



(a) Number of recommended jobs vs inferiority per job.
(b) Number of recommended jobs vs utility per job.
(c) Number of recommended jobs vs mean rank.
(d) Number of recommended jobs vs mean suitability gap.

Fig. E.17. VDAB large: Trends of naive recommendation with increasing k.