Information Systems Research

Online Appendix: Matching Mobile Applications for Cross Promotion

Gene Moo Lee
Sauder School of Business, University of British Columbia
Vancouver, British Columbia, Canada V6T 1Z2, gene.lee@sauder.ubc.ca

Shu He
Department of Operations and Information Management, University of Connecticut
Storrs, Connecticut 06269, shu.he@uconn.edu

Joowon Lee
College of Social Science, Hansung University
Seoul, Korea 136-172, jwlee1015@gmail.com

Andrew B. Whinston
McCombs School of Business, University of Texas at Austin
Austin, Texas 78712, abw@uts.cc.utexas.edu

The mobile app market is one of the most successful software markets. As the platform grows rapidly, with millions of apps and billions of users, search costs are increasing tremendously. The challenge is how app developers can target the right users with their apps and how consumers can find the apps that fit their needs. Cross promotion, advertising a mobile app (target app) in another app (source app), is introduced as a new app promotion framework to alleviate the issue of search costs. In this paper, we model source app user behaviors (downloads and post-download usage) with respect to different target apps in cross-promotion campaigns. We construct a novel app similarity measure using LDA topic modeling on apps' production descriptions, and then analyze how the similarity between the source and target apps influences users' app download and usage decisions. To estimate the model, we use a unique data set from a large-scale random matching experiment conducted by a major mobile advertising company in Korea. The empirical results show that consumers prefer more diversified apps when they are making download decisions compared with their usage decisions, which is supported by the psychology literature on people's variety-seeking behavior. Lastly, we propose an app-matching system based on machine learning models (on app download and usage prediction) and generalized deferred acceptance algorithms. The simulation results show that app analytics capability is essential in building accurate prediction models and in increasing ad effectiveness of cross promotion campaigns, and that, at the expense of privacy, individual user data can further improve the matching performance. The paper has implications on the tradeoff between utility and privacy in the growing mobile economy.

Key words: Mobile applications, cross promotion, matching, search cost, two-sided platform, topic modeling, machine learning, deferred acceptance, algorithm

Appendices

A. Description of Mobile App Samples

A.1. Data Representativeness of Mobile App Samples

Although our paper focuses on the mobile apps that involved with cross promotion (CP) campaigns, it is important to examine the representativeness of the data samples in order to understand the implication of this research on the whole app market. At the data collection period, there were a total of 383,904 apps (`all_apps`) across three major Korean app markets: Apple's App Store, Google's Play Store, and SK Telecom's T-store. Among the apps, 679 of the apps that have adopted the app analytics tool of our data source (`analytics_apps`). Then 128 of the analytics-enabled apps have conducted CP campaigns as target apps (`cp_tgt_apps`) and 137 have conducted mobile display ads (MDA) campaigns (`display_apps`). For each app sample, we report the summary statistics of the following app characteristics: file size, ranking, rating count, average rating, and whether the app is free or has in-app purchase features. In the following analysis, we dropped the app observations with any missing values.

Table 1 shows the summary statistics of the app characteristics in each app sample, where the numbers show the means and standard deviations are in the parentheses. The results show that, comparing to the app samples in our analyses, an app in `all_apps` sample has a slightly larger file size but has less ratings with a lower mean rating (2.38). It also shows that across all samples, almost all apps are free and have in-app purchase features. Comparing to `analytics_apps` sample, the promoted app samples (`cp_tgt_apps` and `display_apps`) are highly ranked, have less ratings, but the mean ratings are higher. Lastly, we do not find significant differences between `cp_tgt_apps` and `display_apps` samples, which suggests that the difference in ad effectiveness between CP and MDA campaigns may not be due to the intrinsic app characteristics.

Table 1 Summary statistics of app samples

	`all_apps`	`analytics_apps`	`cp_tgt_apps`	`display_apps`
App count	383,904	679	128	137
File size (MB)	42.58 (222.87)	30.12 (29.49)	37.23 (35.89)	36.73 (25.14)
Ranking	198.77 (120.89)	247.35 (132.87)	218.08 (99.46)	194.53 (91.62)
# Ratings	303.27 (3865.8)	810.61 (4581.6)	456.42 (2485.4)	390.30 (1396.6)
Avg rating	2.38 (20.08)	4.14 (0.76)	4.21 (0.40)	4.24 (0.55)
Free	99.99%	99.49%	100.0%	100.00%
In-app purchase	99.99%	98.73%	100.0%	99.17%

A.2. Categorical Data

In our empirical analysis, we have categorical variables such as app genres (`cate_i`, `cate_j`), day of week (`weekday_t`), and month (`month_t`), and we describe the categorical variables in distribution charts, rather than summary statistics.

First, we examine the representativeness of 679 analytics-enabled apps by comparing app genres of `analytics_apps` and `all_apps` samples. Figure 1 shows that in both samples, the largest app genre is game. At the same time, we also find that the proportion of game apps in `analytics_apps` sample (67%) is significantly larger than that in `all_apps` sample (20%). Thus we must take caution in applying the empirical findings in our paper to the non-game app genres.

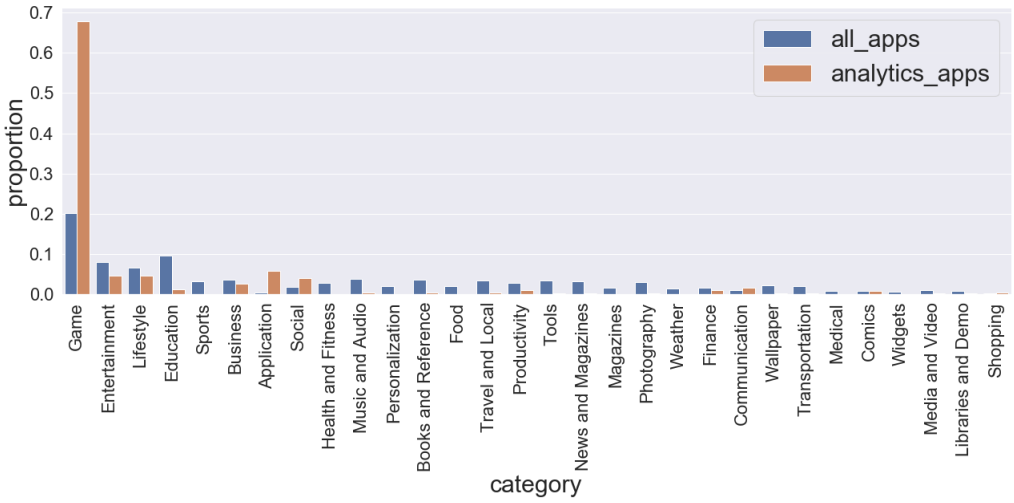
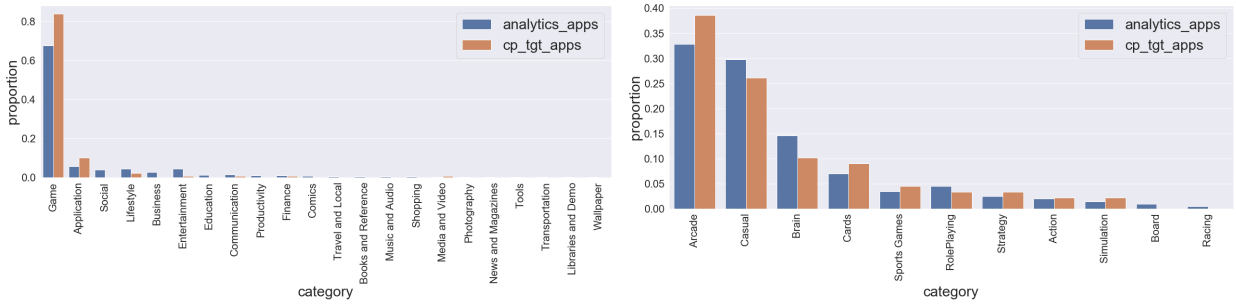


Figure 1 **App genres of `all_apps` and `analytics_apps` samples**

Second, we examine how representative CP target apps (`cp_tgt_apps`) are within `analytics_apps` sample in terms of the app genres. As game is the dominating genre, we generated two plots: one for overall genres and the other for game subgenres. Figure 2(a) shows that in both samples, game genre is the largest one (67% and 83%), followed by the application genre. Within the game genre, Figure 2(b) shows that arcade, casual, and brain games take the largest portion in the two samples. These results show that CP target apps are similar to analytics-enabled apps.

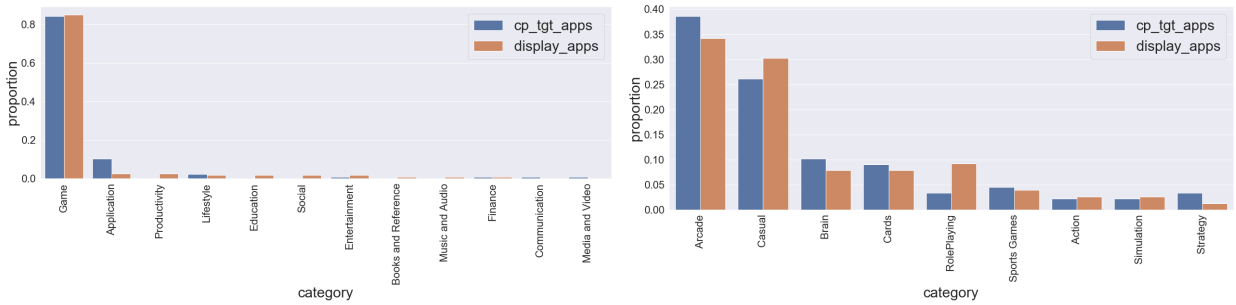


(a) App genre (overall)

(b) Game subgenre

Figure 2 **App genres of `analytics_apps` and `cp_tgt_apps` samples**

Third, we compare the app genres of the apps that conducted CP campaigns (`cp_tgt_apps`) and those with MDA campaigns (`display_apps`) to examine if there is any selection bias by the app developers. We again generated two plots: one for overall genres and the other for game subgenres. Figure 3(a) shows that the proportions of game apps in the two samples are almost identical (83% and 84%). Within the game genre, Figure 3(b) shows that arcade, casual, and brain games take the largest portion in the two samples. One notable difference is that role-playing game apps actively adopted MDA campaigns than CP campaigns, but this subgenre takes a small proportion of the whole app samples. In sum, these results show that in terms of app genres, the apps conducting CP and MDA campaigns are not significantly different from each other.



(a) App genre (overall)

(b) Game subgenre

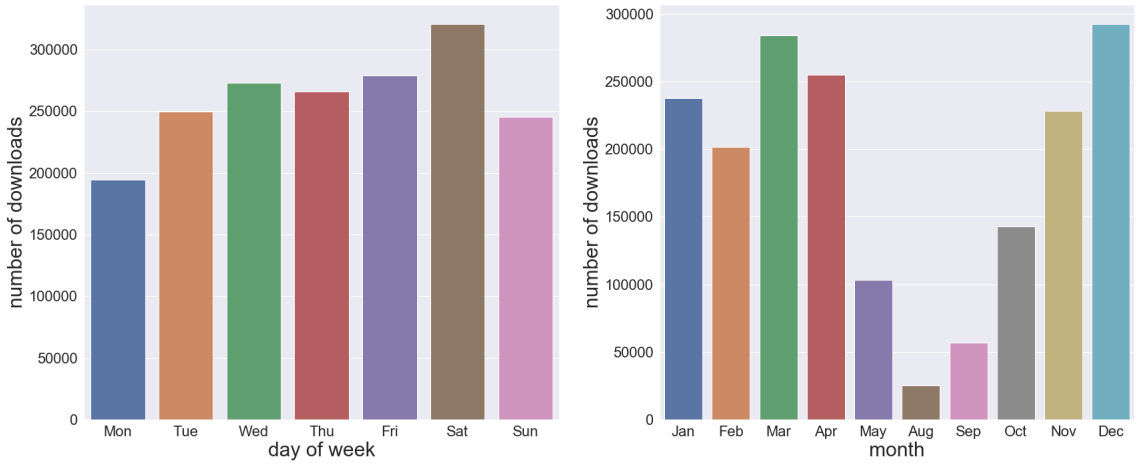
Figure 3 **App genres of `cp_tgt_apps` and `display_apps` samples**

Lastly, we plot the number of downloads from CP campaigns for each day of week and month. Figure 4(a) shows that there are more app downloads on Saturday, which intuitively makes sense as many game players will be active during the weekend. Figure 4(b) shows that most downloads occurred between November 2013 and May 2014. We also observed relatively small number of downloads in August and September 2013 because the CP platform was testing the analytics system during the time. In our empirical analysis, we only used the data from November 2013 to May 2014.

A.3. More Details on Ad Effectiveness in CP and MDA Campaigns

In Section 3.2, we found that CP campaigns are effective in acquiring new app downloads, which can boost up the app rankings and visibility. However, at the same time, the results also show that the users acquired from CP campaigns spent significantly less time on the downloaded apps than those from MDA campaigns or organic downloads. We argue that this can be an evidence of a free rider issue, in which a user downloaded the target app just for in-app credits without genuine interests in using the app.

However, as app developers can select the kind of promotion campaigns they will conduct, there could be selection bias. Thus we examine if there is any systematic difference between the apps that conducted CP and MDA campaigns. First, as shown in Figure 3, we did not find a significant difference in genres between `cp_tgt_apps` and `display_apps` samples. Next, we compare the post-download app usage of



(a) day of week (`weekday_t`) (b) month (`month_t`)

Figure 4 CP-induced app downloads in each day of week and month

the users from CP and MDA campaigns in different app genres. Table 2 shows that except for one genre (`Application`), the users from MDA campaigns spend significantly more time on the downloaded apps than those from CP campaigns. Even for the application genre, the difference of the averages between the two user groups is within the standard deviation. Furthermore, we examined if there is a specific app that CP campaigns can acquire more active users than MDA campaigns. In our sample, there are 35 apps that conducted both types of campaigns, and we found that the users acquired from MDA campaigns are always more active. Hence, we believe the post-download usage performance difference between CP and MDA campaigns is due to the campaign mechanisms, rather than the intrinsic characteristics of the involved apps. But note that CP campaigns dominate MDA campaigns in terms of the number of downloads.

Table 2 Ad effectiveness of CP and MDA campaigns across app genre

Sample	`cp_tgt_apps`	`display_apps`
Post-download app usage	Avg session hour mean (stdev)	Avg session hour mean (stdev)
`Application`	0.013 (0.321)	0.005 (0.341)
`Entertainment`	0.049 (0.727)	0.068 (0.804)
`Lifestyle`	0.000 (0.000)	0.337 (2.710)
`Game`	0.080 (6.359)	0.744 (8.158)
`Game-Arcade`	0.073 (2.415)	0.807 (8.373)
`Game-Cards`	0.400 (6.162)	4.758 (21.91)
`Game-Brain`	0.124 (3.722)	3.800 (40.04)
`Game-Casual`	0.131 (3.149)	2.470 (16.44)
`Game-Sports`	0.020 (1.051)	0.344 (3.889)
`Game-Simulation`	0.085 (0.974)	0.997 (3.838)
`Game-Action`	0.211 (2.314)	2.894 (9.337)
`Game-Strategy`	0.015 (0.585)	0.058 (0.442)

B. LDA Topic Modeling on App Descriptions

Our paper analyzes how the relationship between source and target apps affects a user's download and post-download usage behavior. Thus, a critical task is to quantify the feature relatedness of the two groups of apps. One possible method is to use pre-defined app genre. Specifically, if a pair of two apps belongs to the same app genre, we treat the pair as "similar apps". In the other case, the pair is considered as "unrelated apps". However, this has limitations due to the following reasons. First, this app genre-based approach only provides binary values, where two apps are either similar or not. Moreover, many of the apps in our study belong to the same genre, namely games, hence this approach may not provide appropriate granularity.

To address this challenge, we propose a text-mining approach where we leverage the textual descriptions of the apps in the app market. Latent Dirichlet Allocation (LDA) has been widely adopted in various research contexts in the IS literature (e.g., Shi et al. (2016), Lee et al. (2016), Shin et al. (2019)). The merit of this approach is that, based on the data, the algorithm identifies the groups of related keywords (topics), instead of treating each keyword as an independent dimension. By estimating the topic models from the collection of app descriptions, we can identify the common themes/features in the focal app market. In addition, topic model results provide app-level multinomial distribution on the identified topics. Thus, we can represent an app's features as a vector, where each element represents the weight on a topic. By calculating the cosine similarity between two app's topic vectors, we can quantify the app similarity.

One important hyperparameter that researchers should decide in LDA topic model is the number of topics. Multiple methods have been proposed to guide the optimal number of topics, including perplexity, log likelihood estimates, and inter-topic similarity. In this paper's context of app categorization, we want the constructed app topics to be as distinct as possible. Thus our first decision criterion is to find topics with minimum overlaps. The overlap can be measured by the cosine similarity between two topic's distributions on the vocabulary (vocabulary is the set of unique keywords shown in the text corpus). Since the estimation of LDA model is an iterative process to maximize the log likelihood function, we also use the converged log likelihood estimates as the second criterion.

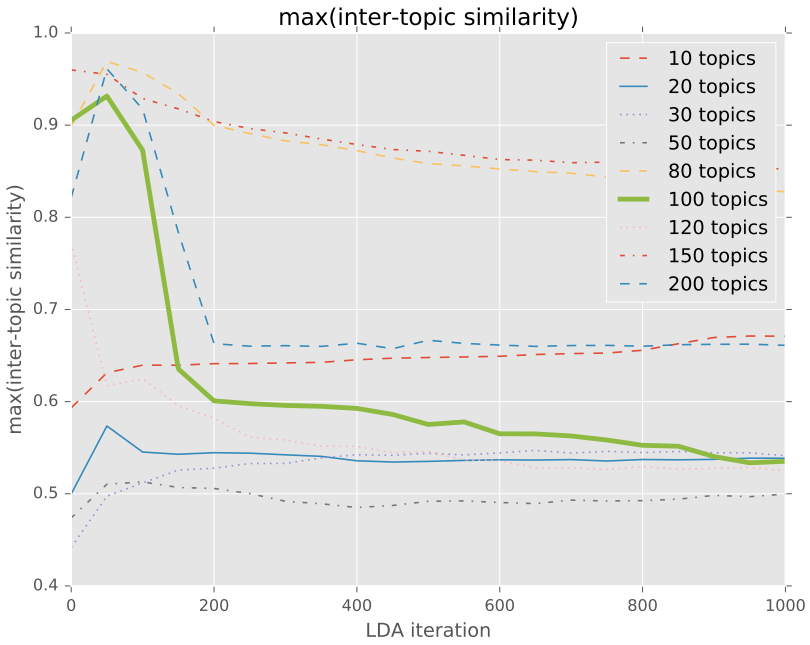
To find the best number of topics, we varied the number of topics to be 10, 20, 30, 50, 80, 100, 120, 150, and 200. As shown in Figure 5, the optimal choice turns out to be 100 (represented as a solid green line), which minimizes the inter-topic similarity between the closest pairs and gives sufficiently high log likelihood estimates. For qualitative evaluation, we show the top keywords for each of the 100 topics in Tables 3 and 4. We note that our empirical results are robust to other choices of number of topics.

Table 3 The 100-topic model (topics 0–49) (Korean keywords are translated into English)

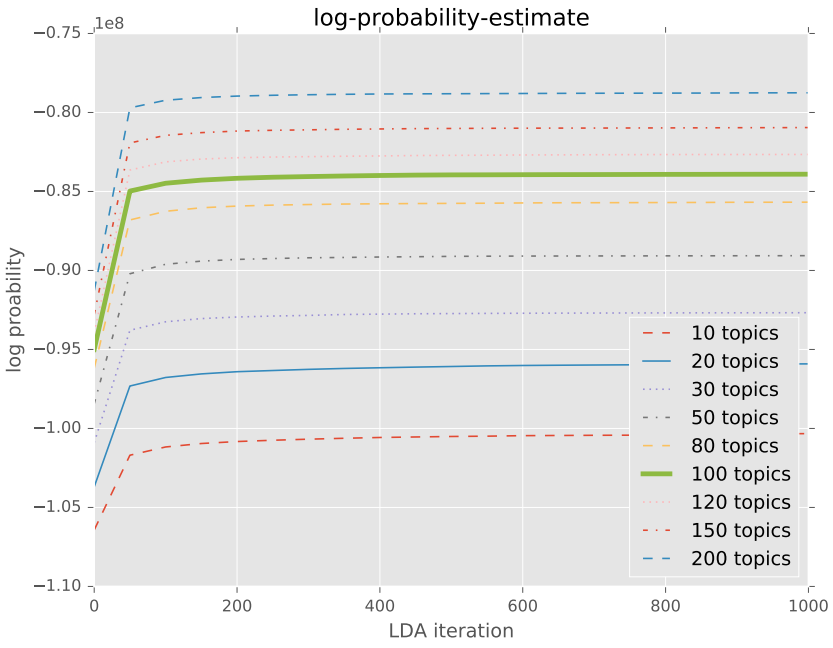
Topic ID	Keywords
0	piano, sskin, classic, flipfont, sound, symphony, Beethoven, Mozart
1	Naver, Kakaotalk, subway, radio, radion, radic, developer, DMB
2	Cat, customer services, login, blick, graphics, free cartoon, video, eula
3	color ring, background, service provider, copyright, go launcher
4	live, color tone, movie, English conversation, speak, writing
5	magazine, ars (automated response service), subscription, vingle, voice menu
6	kids, Cocomong, animation, hearts, master, fun, Cocomong2
7	step, red sun, Young-gook Kim, mbc, stress, setting, amazing content
8	icon, Hello Kitty, atom, screen, game, cute, diary
9	episodes, podcasts, foreign languages, tv drama, weather information
10	LTE, contract, content, SK Telecom, SKT, promotion, free call
11	hymn, copyright, bible, the Lord's prayer, the Apostle's Creed, Ten testaments
12	LTE, voice plans, cake, content, sk telecom, skt
13	series, galaxy, final, system, fantasy, wifi, network, player
14	friends, facebook, play, graphics, developers, upgrade, twitter
15	slug, stage, TOEIC, play, pickat, series,
16	car, racing, simulation, parking, bicycle, place, driver, graphic, simulator
17	radic, game loft, system, graphics, hearts, Naver, talk
18	point, gift card, reference, cookie run, content, kakaotalk, convenience store
19	Chinese, maker, content, foreign language, kids, Korean, HSK, mp3
20	Christmas, Pink Pong, kids, friends, strikers, story, hearts
21	calculator, radic, personal finance, crazyremote, converter, voice
22	personal finance, taleshop, Q&A, scenario, home page, story
23	uparu, trainer, karaoke, sentence, tomato, listening, speaking
24	twitter, love, games, version, g5e, g5games, paid version, free version
25	camera, image, frame, emoticon, sticker, gallery, twitter, facebook, email, friends
26	dragon, Tayo, season 1, animation, friends, tomato, story
27	music island, epilus, mr karaoke, karaoke, hellip, pop, musicsum, romance, sound
28	lotto, tethering, seller, lottery, lottery number, round
29	social commerce, shopping mall, gifts, brand, style, emart store, category
30	style, style tag, smart, i-coach, designer, email
31	phone trend, Three county game, kakao, eternity, users
32	Three country game, system, pvp, naver, RPG, the apps, quest
33	English listening, smart teps, ted, smart, player, vocabulary, movie
34	Pororo, friends, animation, sing, kids, adventure, content
35	give, fun, application, Cocomong, Boddle book, puppy, kid songs
36	what's the number, poweramp, go locker, bull, phone number, dotemu, voice phishing
37	Japanese, level, power ranger, Hiragana, toddler, series
38	image, face, beautiful, fun, usage, animation
39	video, iPhone, email, iPad, image, player, message, audio, text
40	online, daum, push, 100, systems, favorites, wiki, click, live
41	naver, dodol launcer, dodol home, blog, icon, dodol, installation
42	kakao talk, alarm, cell phone, kakao story, password, message alarm sound
43	learn, icon, navigation, hearts, concentration, pop-up, banner, screen
44	planet, launcher, home page, applicable, launch planet, icon
45	recruiting, mobile t-money, resume, check card, Job Korea, OK Cashback
46	bull, version, friends, memory, process, paypin, music, music videos
47	bill, mobile phone, skt, login, customer services, point, password
48	calendar, anniversary, diary, OK Cashback, making, point
49	subway, station, guide, public transportation

Table 4 The 100-topic model (topics 50-99) (Korean keywords are translated into English)

Topic ID	Keywords
50	Com2us, partial payment, homepage, bull, phone decoration, Q&A
51	Korean, Korea, tourism, 2nd stage, smart wallet, bull, travel information
52	EBS (Education Broadcasting System), application, radio, upload, live show, 1644
53	fortune telling, 2014, love, fortune, health, spouse
54	system, monsters, RPG, kakao, sotory, naver, play, customer services
55	prime, speedup, Doosan Donga, system, interested items, contents
56	TV drama, video-on-demand, contents, Inkaboda, price, test, YouTube
57	Waterbear software, app, twitter, facebook, up down, app store
58	app developer, iPhone, smart, friends, messages, emails, gmail, facebook
59	Baby bus, bull, spell correction, dice, Blue Marble, history, play
60	offline, GPS, navigation, Internet, map, POI (Point of interest)
61	English vocabulary, test, native speaker, TOEIC, words, textbook
62	fish, native speakers, award, Chinese, foreign, translation, Japanese
63	story, kids, animation, series, books, Cinderella, fun, Snow White
64	speaking, English conversation, speaking, proud, TOEIC, expressions
65	Hackers, listening, real estate, best sellers, membership, contents
66	live, Galaxy, beautiful, icon, images, launcher
67	NFC (near field communication), touch, USIM, smart, nemo, cell phone
68	diary, calorie, recipe, TOEIC, stretching, application, fitness
69	battery, hungry app, memory, booster, system, Naver, manager
70	mobile, plus, application, Gameville, network, terminal
71	Hootos, animation, friends, song, Cocomong, kids
72	kids, fun, alphabet, animation, tomokids, smart,
73	chapter, certificate, OPIC, bike, study, speaking, cell phone, lesson
74	radic, home, love, cross, love, to you
75	map, keyboard, nbsp, Japanese, store, voice information, sign, setup
76	sports, baseball, gina, nba, worldcup, score, proto, 2014, kbo
77	exercise questions, written exam, government official, test, pre-test
78	security card, sau, delivery service, delivery food, cryptography, password
79	personal information, game master, 2014, system, people, animation bible
80	book 21, Korea, story, series, Galaxy, home page, Facebook
81	title, YBM, CNN, TOEIC, YFS, word book, android, test
82	smart, wifi, flash light, battery, ring tone, navigation
83	lock screen, kakaotalk, phone theme, findnlock, copyright
84	kakaotalk, icon, go launcher, phone number, phone background image
85	MP3, battery, kids, 50 songs, Internet, series, player, recorder
86	unit, learning module, clubs, impossible, chaos, rings, beatbox
87	MP3, removed, highlight, preview, artist, OST
88	image, friends, full version, yellow, draw, punch shot, exam prep
89	Naver, blog, post, mail, Korean, 010, English diary
90	English, cloud, restart, schedule, story, English book, talking, waterbear
91	customer service, bluepin, content, series, animation
92	library, N drive, naver, lol, time table, announcements
93	Korean, Spanish, Chinese, French, German, Japanese, Italian
94	MC square, membership, bar code, T-store, re-download, tennis
95	EXO, English composition book, ovjet, Cloud bread, content, kids, animation
96	blind dating, profile, social dating, partner, random chatting
97	background image, live, image, subway, live background image
98	app, game, free, time, new iPhone, music, fun
99	voca, MegaBox, highway, vocabulary bible, traffic information, contents



(a) Inter-topic similarity



(b) Log likelihood

Figure 5 LDA results with different numbers of topics

C. Heckman Selection Model

The total usage that target app j acquires by conducting a CP campaign on source app i can be represented by the following formula:

$$usage_{ij} = \sum_{t=1}^{T} 1(M_{ijt} = 1) * us\tilde{a}ge_{ijt} * ND_{ijt} \tag{C.1}$$

where t represents the t^{th} day of the entire CP campaign period T, M_{ijt} is a binary variable indicating if source app i and target app j are matched on day t, $us\tilde{a}ge_{ijt}$ represents the average usage on target app j for users who downloaded j via source app i's CP campaign on day t, and ND_{ijt} is the number of users from source app i who downloaded target app j on day t. This formula assumes two interdependent stages of user decisions: download and post-download usage. We model user behaviors using a two-stage model, characterizing each stage.

C.1. Stage 1: Download

When a mobile user is using source app i, the user is exposed to a list of target apps matched by the CP platform. The user will be offered an award in source app i if the user downloads one of the target apps. Apparently, the user's decision on whether to download target app j from source app i on day t is dependent on the utility that the user can obtain from the download. Since downloads are rewarded, it is possible that users download the target app without the intention of using it. The latent utility that an average user of source app i earns from downloading target app j ($\tilde{ud}_{ijt}$) on day t can be expressed by using source app i's and target app j's characteristics and time effect t, as well as average user characteristics of source app i's users. The formula is as follows:

$$\begin{gathered}\tilde{ud}_{ijt} = \alpha_0 + \alpha_1 X_{it} + \alpha_2 X_{jt} + \alpha_3 X_{ijt} + \alpha_4 topic_ij + \alpha_5 topic_sq_ij + \alpha_4 X_t + \delta_1 \varepsilon_{1ijt}, \\ \varepsilon_{1ijt} \sim N(0,1),\end{gathered} \tag{C.2}$$

where X_{it}, X_{jt}, X_t, X_{ijt}, $topic_ij$, and $topic_sq_ij$ are the vectors of characteristics as in Section 5.1. δ_1 is the standard deviation of unobserved characteristics in users' download decisions. For simplicity, formula (C.2) can also be written as:

$$\tilde{ud}_{ijt} = \mu_{ijt} + \delta_1 \varepsilon_{1ijt}, \tag{C.3}$$

where μ_{ijt} is the observed part of the download latent utility.

Given the latent utility formula above, we can define the number of users from source app i who chose to download target app j after seeing the CP campaign on day t with the following formula:

$$ND_{ijt} = NA_{ijt} \times P[\tilde{ud}_{ijt} > 0 | X], \tag{C.4}$$

where NA_{ijt} represents the number of active users on source app i on day t who are matched with target app j.[1] Without loss of generality, we set $\delta_1 = 1$. Then the conditional probability for an average user of source app i downloading target app j on day t given other observable characteristics is:

$$P[\tilde{ud}_{ijt} > 0|X] = P[\varepsilon_{1ijt} > -\mu_{ijt}] = \Phi(\mu_{ijt}), \tag{C.5}$$

where Φ is the cumulative distribution function of the standard normal distribution.

C.2. Stage 2: Post-Download Usage

In the second stage, a user of source app i who has downloaded target app j will decide how much time to spend on target app j based on the usage utility. We model the usage utility of a user who downloads target app j from source app i as follows:

$$\begin{gathered} \tilde{usage}_{ijt} = \beta_0 + \beta_1 X_{it} + \beta_2 X_{jt} + \beta_3 X_{ijt} + \beta_4 topic_ij + \beta_5 topic_sq_ij + \beta_6 X_t + \delta_2 \varepsilon_{1ijt} + \delta_3 \varepsilon_{2ijt} \\ \varepsilon_{1ijt} \sim N(0,1),\ \varepsilon_{2ijt} \sim N(0,1) \end{gathered} \tag{C.6}$$

where X_{it}, X_{jt}, X_{ijt}, X_t, $topic_ij$, and $topic_sq_ij$ are the vectors of characteristics as in Section 5.1. From formula (C.6), we can easily calculate that the standard deviation of unobserved characteristics is $\sigma = \sqrt{\delta_2^2 + \delta_3^2}$ and the correlation between the unobserved characteristics of the two stages is $\rho = \delta_2 / \sqrt{\delta_2^2 + \delta_3^2}$. Again, for display simplicity, formula (C.6) can also be represented in the following way:

$$\tilde{usage}_{ijt} = \gamma_{ijt} + \delta_2 \varepsilon_{1ijt} + \delta_3 \varepsilon_{2ijt}. \tag{C.7}$$

C.3. Log-Likelihood Function

Following the maximum likelihood estimation of Heckman's sample selection model (Heckman (1979)), for all users, the log-likelihood formula considering both their download and usage behavior in our data set is the following:

$$\begin{aligned} ln\mathscr{L}(\theta) = \sum_{t=1}^{T} \sum_{j} \sum_{i} 1(M_{ijt} = 1) \times [&(NA_{ijt} - ND_{ijt}) ln(1 - \Phi(\mu_{ijt})) \\ &+ ND_{ijt}(ln(\Phi(\mu_{ijt})) + ln(h(\tilde{usage}_{ijt}|X, \alpha, \beta, \rho, \sigma)))], \end{aligned}$$

where

$$h(\tilde{usage}_{ijt}|X, \alpha, \beta, \rho, \sigma) = \frac{f((\tilde{usage}_{ijt} - \gamma_{ijt})/\sigma)}{\sigma \Phi(\mu_{ijt})} \Phi\left(\frac{\rho(\tilde{usage}_{ijt} - \gamma_{ijt})/\sigma + \mu_{ijt}}{\sqrt{1 - \rho^2}}\right). \tag{C.8}$$

To estimate the coefficients in the model, we use the classic Nelder-Mead method to find the maximum of the log-likelihood function (Nelder and Mead (1965)).[2] To reduce computational time and exclude local optimum problems, we set the initial parameters to be the parameter vectors α and β estimated from the reduced-form regressions in Section 5.

[1] According to the CP platform, a user of a source app will be shown seven random target apps in each promotion opportunity. Thus we calculate the number of active users with access to target app j on source app i by the following formula:

$$NA_{ijt} = \frac{\text{Number of total active users of source app } i \text{ on day } t * 7}{\text{Number of total number of target apps on day } t}$$

[2] The detailed derivation of the likelihood function is described in Appendix D.

D. Maximum Likelihood Estimation of Heckman's Sample Selection Model

The probability for an average user of source app i of downloading the target app j is as follows:

$$P[\tilde{ud}_{ijt} > 0|X] = P[\varepsilon_{1ijt} > -\mu_{ijt}] = \Phi(\mu_{ijtk}). \tag{D.1}$$

The cumulative distribution function of post-download usage conditional on downloading app j and other characteristics can be represented by

$$\begin{aligned} H(us\tilde{a}ge_{ijt}|X) = & \quad P(us\tilde{a}ge_{ijt} < \hat{u}|D_{ijt} = 1, X) \\ = & \quad \frac{P(us\tilde{a}ge_{ijt} < \hat{u} \cap D_{ijt} = 1|X)}{P(D_{ijt} = 1|X)} \\ = & \quad \frac{P(us\tilde{a}ge_{ijt} < \hat{u} \cap D_{ijt} = 1|X)}{\Phi(\mu_{ijt})} \end{aligned} \tag{D.2}$$

Without loss of generality, let us assume that $\delta_2 > 0$. We can then rephrase formula D.2 as follows:

$$\begin{aligned} H * \Phi(\mu_{ijt}) = & \quad P[-\mu_{ijt} < \varepsilon_{1ijt} < us\tilde{a}ge_{ijt} - \gamma_{ijt} - \delta_3\varepsilon_{2ijt})/\delta_2|X, \alpha, \beta, \rho, \sigma] \\ = & \quad \int_{-\infty}^{\infty} P[-\mu_{ijt} < \varepsilon_{1ijt} < (\tilde{usage}_{ijt} - \gamma_{ijt} - \delta_3\varepsilon_{2ijt})/\delta_2|X, \alpha, \beta, \rho, \sigma] \times f(u)du \\ = & \quad \int_{-\infty}^{(\tilde{usage}_{ijt} - \gamma_{ijt} + \delta_2\mu_{ijt})/\delta_1} [F((\tilde{usage}_{ijt} - \gamma_{ijt} - \delta_3\varepsilon_{2ijt})/\delta_2) - F(-\mu_{ijt})] \times f(u)du \\ = & \quad \int_{-\infty}^{(\tilde{usage}_{ijt} - \gamma_{ijt} + \delta_2\mu_{ijt})/\delta_1} F((\tilde{usage}_{ijt} - \gamma_{ijt} - \delta_3\varepsilon_{2ijt})/\delta_2) f(u)du \\ & \quad - F(-\mu_{ijt}) F((\tilde{usage}_{ijt} - \gamma_{ijt} + \delta_2\mu_{ijt})/\delta_1). \end{aligned} \tag{D.3}$$

If we replace $v = us\tilde{a}ge_{ijt} - \gamma_{ijt} - \delta_3\varepsilon_{2ijt}$ in D.3, we obtain the following formula:

$$\begin{aligned} H * \Phi(\mu_{ijt}) = & \quad \frac{\delta_2}{\delta_3} \int_{-\mu_{ijt}}^{\infty} F(v) f((us\tilde{a}ge_{ijt} - \gamma_{ijt} - \delta_2 v)/\delta_3)dv \\ & \quad - F(-\mu_{ijt}) F((\tilde{usage}_{ijt} - \gamma_{ijt} + \delta_2\mu_{ijt})/\delta_1) \\ = & \quad \int_{-\mu_{ijt}}^{\infty} F((us\tilde{a}ge_{ijt} - \gamma_{ijt} - \delta_2 v)/\delta_3) f(v)dv. \end{aligned} \tag{D.4}$$

The conditional density function is then:

$$\begin{aligned} h(us\tilde{a}ge_{ijt}|X) = & \quad \frac{\partial H(us\tilde{a}ge_{ijt}|X, \alpha, \beta, \rho, \sigma)}{\partial us\tilde{a}ge_{ijt}} \\ = & \quad \frac{1}{\delta_3 \Phi(\mu_{ijt})} \int_{-\mu_{ijt}}^{\infty} f((us\tilde{a}ge_{ijt} - \gamma_{ijt} - \delta_2 v)/\delta_3) f(v)dv. \end{aligned} \tag{D.5}$$

From the standard normal density f, we can show that

$$\int_{-\infty}^{c} f(a+bx)f(x)dx = \frac{f(a/\sqrt{b^2+1}}{\sqrt{b^2+1}} \times F(c\sqrt{b^2+1}+ab/\sqrt{b^2+1}). \tag{D.6}$$

We can thus easily obtain

$$\begin{aligned} h(us\tilde{a}ge_{ijt}|X) = & \frac{f((us\tilde{a}ge_{ijt}-\gamma_{ijt})/\sqrt{\delta_3^2+\delta_2^2}}{\sqrt{\delta_3^2+\delta_2^2}F(\mu_{ijt})} \\ & \times F(\frac{\delta_2(us\tilde{a}ge_{ijt}-\gamma_{ijt})+(\delta_3^2+\delta_2^2)\mu_{ijt}}{\delta_1\sqrt{\delta_3^2+\delta_2^2}}). \end{aligned} \tag{D.7}$$

If we substitute

$$\delta_3 = \sigma\sqrt{1-\rho^2},\ \delta_2 = \rho\sigma,$$

then formula D.7 can be simplified to:

$$h(us\tilde{a}ge_{ijt}|X,\gamma,\mu,\rho,\sigma) = \frac{f((us\tilde{a}ge_{ijt}-\gamma_{ijt})/\sigma)}{\sigma\Phi(\mu_{ijt})}\Phi(\frac{\rho(us\tilde{a}ge-\gamma_{ijt})/\sigma+\mu_{ijt}}{\sqrt{1-\rho^2}}). \tag{D.8}$$

E. App Similarity Measure Using Alternative Models

Besides LDA topic modeling, there are alternative machine learning models to measure the similarity of textual content. We investigate two representative approaches: TF-IDF and doc2vec.

When measuring the similarity between a pair of documents, the most naïve approach is to count the common words in the two documents, which is called term frequency (TF) approach. However, the issue of this approach is that all words, including stop words and frequently appearing words, have equal weights in the document distinguishing process. To address this issue, term frequency-inverse document frequency (TF-IDF) approach considers the inverse document frequency to put smaller weights on the keywords that commonly appear in many documents (Salton and Buckley (1988)). We use TF-IDF as an alternative approach to quantify the similarity of a pair of app descriptions. TF-IDF still assumes that each term represents a unique dimension in the semantic space, where there could be related keywords (e.g, mobile and device) and synonyms (e.g., mobile phone and cellphone). Topic modeling approach such as LDA is to group related keywords into topics.

However, TF, TF-IDF, and LDA all have two common limitations: First, the models do not take the order of words into account as they use the *bag-of-words* approach. Second, these models represent each word or topic in discrete atomic symbols, which leads data sparsity issues and requires more data in order to successfully train the underlying statistical models.

Vector space model (VSM) is an approach to address these issues. VSMs represent (or embed) each word into a continuous vector space where semantically related words are mapped to nearby positions (word embeddings). VSM assumes distributional hypothesis, which means that co-occurring words share the contexts and semantics. Word2vec (Mikolov et al. (2013)) is a widely used neural network-based VSM that predicts a word using the neighboring words. Doc2vec is an extension to word2vec that are used to learn document-level embeddings (Le and Mikolov (2014)).

We use Python's gensim implementation[3] of the doc2vec model to create a distance measure `doc2vec_ij`, and Python's scikit-learn implementation[4] of the TF-IDF model to create a similarity measure `tfidf_ij` between a pair of apps using their textual descriptions. Please find the summary statistics of these two variables in Table 5. Note the sample size has decreased from 41,294 to 41,254 because of the missing values of `doc2vec_ij` and `tfidf_ij`. Since the TF-IDF model is to count common words in each document, most of the similarity values are quite close to zero. As mentioned, multiple words may describe the same concept, but this could not be captured by the TF-IDF model. Hence, we believe that TF-IDF model based similarity measure may not be a good candidate. Doc2vec model generates numeric vectors where individual elements can be negative values, while LDA model always generate nonnegative vectors.

[3] `https://radimrehurek.com/gensim/index.html`

[4] `https://scikit-learn.org/stable/index.html`

Table 5 Summary statistics of the two alternative measures

Variables	Obs.	Mean	Stdev	Min	25%	50%	75%	Max
doc2vec_ij	41,254	12.965	2.135	4.941	11.522	12.968	14.438	19.717
tfidf_ij	41,254	0.0242	0.0339	0	0.00852	0.0165	0.0301	0.542

Thus we use the doc2vec-based distance measures in this robustness check, meaning that a smaller value of doc2vec_ij indicates that the two apps have more overlapping contexts.

We repeated all the empirical analysis in Section 5 and the results are reported in Tables 6, 7, and 8. For the post-download session time and connection counts, the results show that the increase of doc2vec_ij will lead to a decrease of post-download usage,[5] which further validates our findings in Section 5. For the download ratio measure, except for the linear model, we observe an inverse U-shaped relationship between the download ratio and the doc2vec_ij in all other model specifications. Because of the reversed relationship between the doc2vec_ij (distance) and topic_ij (similarity), we cannot directly compare the estimated coefficients to see if these two measures are comparable for the download ratio. But, still, we can generate the impact of two measures at different levels for download ratio based on the session two-stage model in Figure 6. The results show that the CP campaign will have a highest download ratio when doc2vec_ij is around 13.25 and topic_ij is about 0.5.

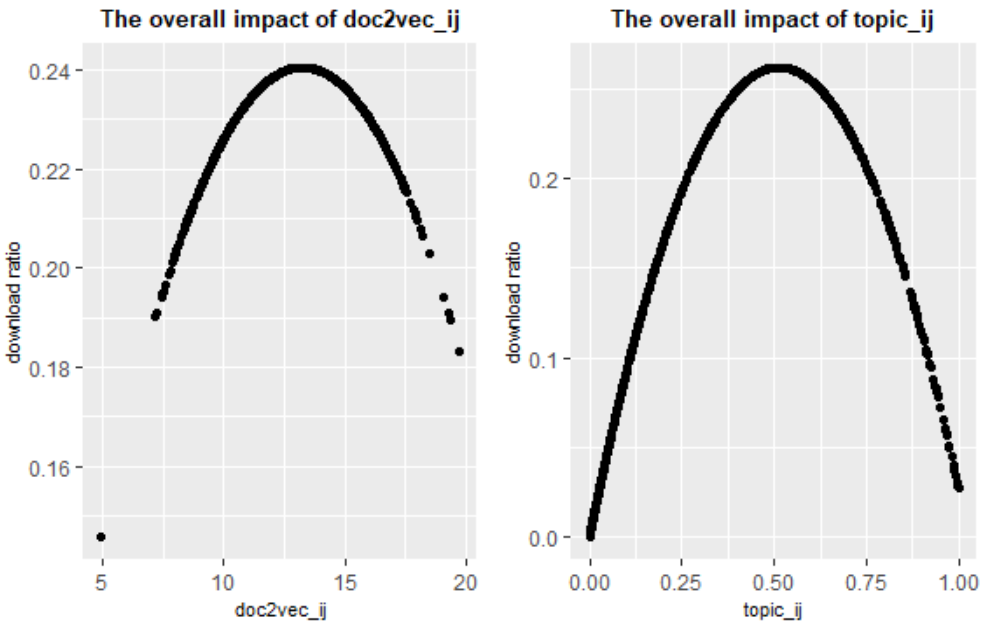


Figure 6 Comparison of app distance/similarity's impact on download ratio (doc2vec and LDA models)

[5] For the session time based two-stage model, the coefficient of the doc2vec_sq_ij is positive and significant, the conditional marginal effect of doc2vec_ij is still positive even with the largest value of doc2vec_ij.

Table 6 Empirical results on download ratio (doc2vec model)

	linear		Tobit	hurdle		two-stage	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Variables	dn_ratio_ijt	dn_ratio_ijt	dn_ratio_ijt	dn_ratio_ijt	selection	sess_ijt	conn_ijt
doc2vec_ij	0.0971	-0.0671***	0.647***	0.633***	0.0265	0.0363***	0.0785***
	(0.0595)	(0.0126)	(0.186)	(0.0508)	(0.0301)	(0.00382)	(0.00495)
doc2vec_sq_ij	-0.00659***		-0.0231***	-0.0232***	-0.00102***	-0.00137***	-0.00305***
	(0.00233)		(0.00731)	(0.00202)	(0.00117)	(1.44e-4)	(1.78e-4)
Constant	-7.310***	-8.065***	-8.831***	-4.161***	-1.356***	-3.038***	-2.968***
	(0.425)	(0.261)	(1.816)	(0.352)	(0.217)	(0.0533)	(0.0557)
Control variables	Yes	Yes	Yes	Yes	Yes	Yes	Yes
App FE	Yes	Yes					
App genre FE			Yes	Yes	Yes	Yes	Yes
Ranking	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Time FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Observations	41,254	41,254	41,254	41,254	41,254	41,254	41,254

Note: This table represents the empirical results showing how source app users' target app download behavior vary with different source–target app pairs. doc2vec_ij and doc2vec_sq_ij are the doc2vec based distance between the source app and the target app and its squared term, respectively. Other control variables include source app *i*'s and target app *j*'s time-varying app characteristics, the day's characteristics, and time-varying dyadic characteristics defined by source app *i* and target app *j*. In columns (1) and (2), the data is analyzed by the panel data fixed effects linear model. In column (3), the data is analyzed by the Tobit model. In columns (4) and (5), the data is analyzed by the hurdle model. In columns (6) and (7), the data is analyzed by the two-stage model. * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 7 Empirical results on post-download session time (doc2vec model)

	linear		Tobit	hurdle		two-stage
	(1)	(2)	(3)	(4)	(5)	(6)
Variables	sess_ijt	sess_ijt	sess_ijt	sess_ijt	selection	sess_ijt
doc2vec_ij	-0.324**	-0.172***	-0.202***	-0.139***	-0.0119***	-2.988***
	(0.137)	(0.0234)	(0.0904)	(0.0219)	(0.00503)	(0.103)
doc2vec_sq_ij	0.00609					0.0737***
	(0.0110)					(0.00412)
Constant	18.56***	17.63***	-131.5	12.45***	-2.709***	-3.260***
	(0.953)	(0.378)	(471.1)	(0.982)	(0.192)	(1.24)
Control variables	Yes	Yes	Yes	Yes	Yes	Yes
App FE	Yes	Yes				
App genre FE			Yes	Yes	Yes	Yes
Ranking	Yes	Yes	Yes	Yes	Yes	Yes
Time FE	Yes	Yes	Yes	Yes	Yes	Yes
Observations	41,254	41,254	41,254	41,254	41,254	41,254

Note: This table represents the empirical results showing how source app users' target app post-download average session time vary with different source–target app pairs. doc2vec_ij and doc2vec_sq_ij are the doc2vec based distance between the source app and the target app and its squared term, respectively. Other control variables include source app *i*'s and target app *j*'s time-varying app characteristics, the day's characteristics, and time-varying dyadic characteristics defined by source app *i* and target app *j*. In columns (1) and (2), the data is analyzed by the panel data fixed effects linear model. In column (3), the data is analyzed by the Tobit model. In columns (4) and (5), the data is analyzed by the hurdle model. In column (6), the data is analyzed by the two-stage model. * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

F. Parallel Test for the DID Analysis

To test the parallel trend assumption of the DID specification in Section 6 (Angrist and Pischke (2008)), we add a set of interaction terms of the bi-weekly dummies and the dummy whether the app is source or target app (Autor (2003)). This way, we explore how apps' ranking data evolve between the treated and

Table 8 Empirical results on post-download connection count (doc2vec model)

	linear		Tobit	hurdle		two-stage
	(1)	(2)	(3)	(4)	(5)	(6)
Variables	conn_ijt	conn_ijt	conn_ijt	conn_ijt	selection	conn_ijt
doc2vec_ij	-0.0853*	-0.0561***	-0.0108	-0.00945	-1.12e-5	-0.178
	(0.0443)	(0.00753)	(0.0120)	(0.00781)	(0.00415)	(0.150)
doc2vec_sq_ij	0.00117					-0.0964***
	(0.00314)					(0.00598)
Constant	6.637***	6.457***	-1.055	1.873***	-1.454***	2.063
	(0.303)	(0.110)	(0.689)	(0.182)	(0.112)	(1.92)
Control variables	Yes	Yes	Yes	Yes	Yes	Yes
App FE	Yes	Yes				
App genre FE			Yes	Yes	Yes	Yes
Ranking	Yes	Yes	Yes	Yes	Yes	Yes
Time FE	Yes	Yes	Yes	Yes	Yes	Yes
Observations	41,254	41,254	41,254	41,254	41,254	41,254

Note: This table represents the empirical results showing how source app users' target app post-download average connection counts vary with different source–target app pairs. doc2vec_ij and doc2vec_sq_ij are the doc2vec based distance between the source app and the target app and its squared term, respectively. Other control variables include source app i's and target app j's time-varying app characteristics, the day's characteristics, and time-varying dyadic characteristics defined by source app i and target app j. In columns (1) and (2), the data is analyzed by the panel data fixed effects linear model. In the column (3), the data is analyzed by the Tobit model. In columns (4) and (5), the data is analyzed by the hurdle model. In the column (6), the data is analyzed by the two-stage model. * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

control groups before and after participating in the CP campaigns. Specifically, we aggregate apps' daily activity information into a bi-weekly level and drop the dummy for the two weeks right before the initial CP participation date as the reference groups. If the parallel trend assumption holds, we would expect the interaction dummies before the treatment dates are not statistically different from zero while the post-treatment interaction dummies are consistent with the main DID analysis results. The graphs representing all the estimated coefficients of those interactions are displayed in Figures 7 and 8. Most of the pre-treatment interaction dummies are not significantly different from zero. For the target apps, the treatment effects of participating the CP campaigns are long lasting and the magnitudes are largest right after their initial CP campaign participation. On the other hand, the treatment effects are much weaker for source apps, as we have observed from the main DID analysis.

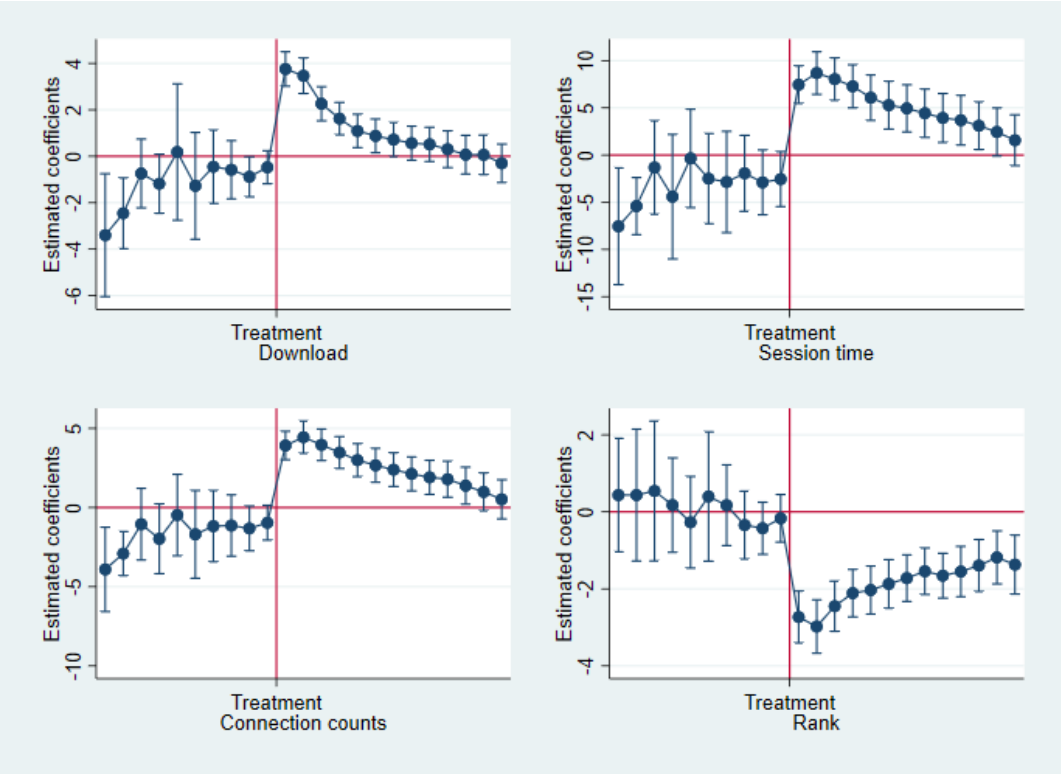


Figure 7 **Parallel trend tests for the target apps**

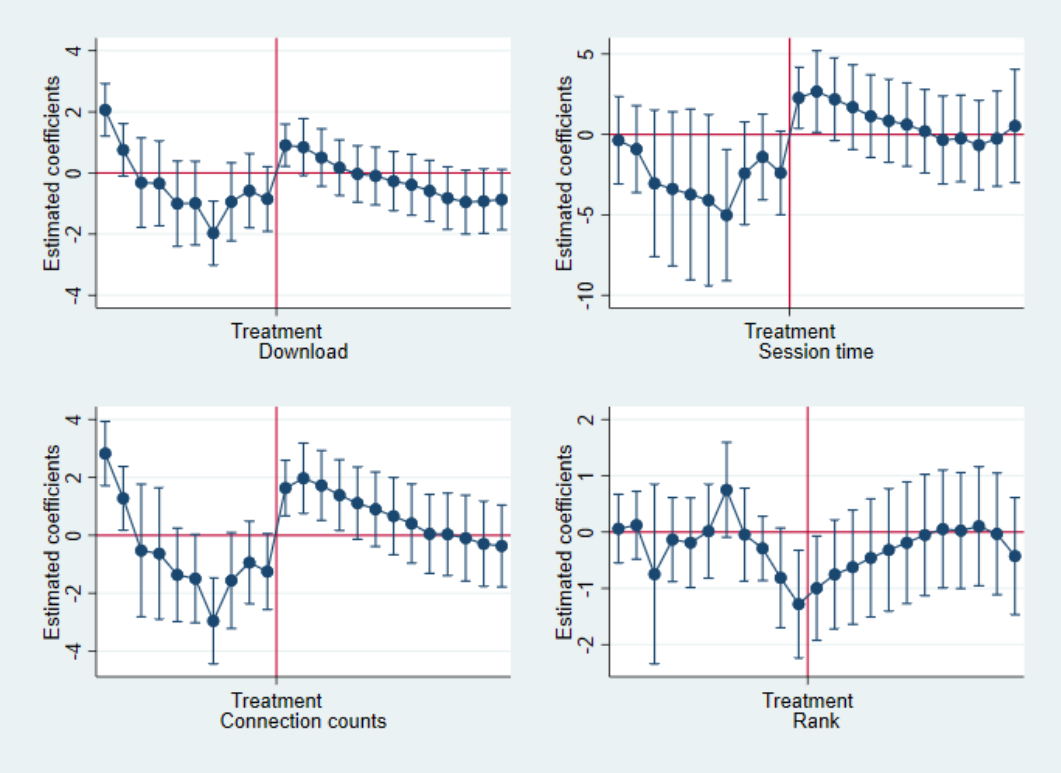


Figure 8 **Parallel trend tests for the source apps**

G. Generalized Deferred Acceptance Algorithm

We describe the generalized deferred acceptance algorithm for app matching for CP campaigns. As a high-level description of the algorithm for two-sided markets, one side proposes to the other side and the other side accepts one of the proposed counterparts. This propose-and-accept procedure continues until no individual players have incentives to make further proposals. Either side of a two-sided market can be the proposing side. Taking a source app-proposing algorithm as an example, our generalized deferred acceptance algorithm can be described as follows:

- **Initial Step**: All source and target apps are unmatched.
- **Proposing Step**: Each unmatched source app selects seven target apps based on its preference list. The preference list, or choice function, includes all possible target apps with the corresponding utility values for each source app. We derive the preference list for each app based on the expected download ratio and post-download usages from the machine learning-based prediction models from Section 7.2.[6]
- **Accepting Step**: Each target app will accept no more than a fixed number of source (seven in our matching experiment) apps from the proposer pool, based on its own preference list. The target app may replace previously matched source apps with other source apps in case of higher expected utility values. The rejected source app removes the rejecting target app from its preference list.
- **Termination Step**: If any unmatched source app has more proposals to make, then go to the Proposal Step. Otherwise, the algorithm is terminated.

Since the source apps are the proposers in this case, we call this algorithm a *source-proposing deferred acceptance algorithm* (`src_da_match`). The pseudocode of the algorithm is given in Algorithm 1.

G.1. Variations of Deferred Acceptance Algorithm

The algorithm can vary by changing the proposing side and switching the utility function. First, the proposing step can be initiated by the target app side instead of the source app side (which we term *target-proposing deferred acceptance algorithm* (`tgt_da_match`)). Second, the utility function can be defined in the following two ways: (1) by maximizing expected downloads and (2) by maximizing expected post-download usage. For contracts with a fixed unit price per download, one can intuitively assume that source app developers would optimize for the total number of downloads, which would maximize their profits. From the target app's perspective, one can assume that target apps will optimize for either download or usage. By optimizing for downloads, target apps can acquire more users and improve their ranks in the app market charts. By optimizing for usage, target apps acquire more loyal users. In the case of a fixed fraction profit-sharing contract, both the source and target app developers would want to optimize for post-download usage. We conduct experiments to compare these two scenarios: (i) both source and target optimize downloads (to

[6] In theory, app developers may report their own preference lists. However, in the case of crowded app markets with massive number of apps, individual developers may not have a full understanding of the predicted numbers of downloads or post-download usage for a potential match.

inputs : S: set of source apps, T: set of target apps, U: preference matrix where $U_{s,t}$ is the utility of s if matched with t, MAX: per-app match upper bound

output: matching matrix M, where $M_{s,t}$=1 if s and t are matched and $M_{s,t}$=0 otherwise

// initialize the matching matrix

for *each* $s \in S$ **do**
  for *each* $t \in T$ **do**
    $M_{s,t} = 0$
  end
end

for *each* $s \in S$ **do**
  $tried_s \leftarrow \emptyset$ // already proposed targets for each source
  $failed_s \leftarrow \emptyset$ // already proposed, but failed targets for each source
end

$new_proposers \leftarrow S$ // in the first step, all sources will propose

// keep the loop until there are no more proposers

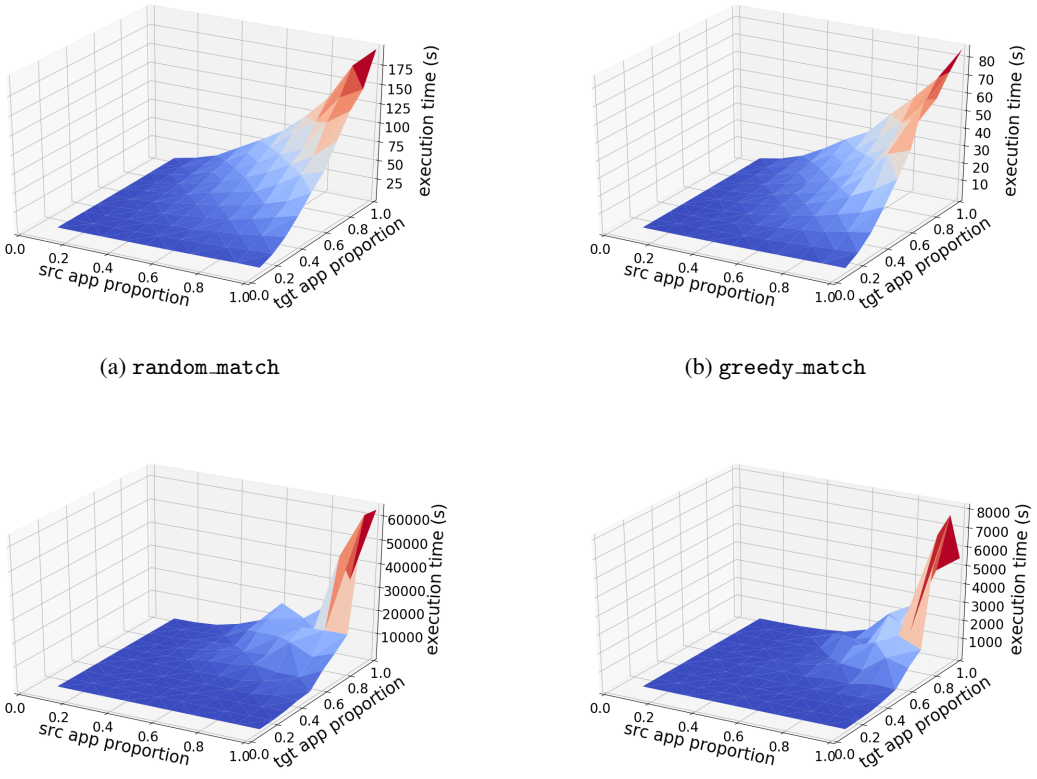
while $new_proposers != \emptyset$ **do**
  // Proposing step: each source will decide new targets to propose
  $new_proposers \leftarrow \emptyset$
  for *each* $s \in S$ **do**
    $pool \leftarrow T - (tried_s \cup failed_s)$ // available target pool
    $proposing_s \leftarrow$ up to MAX targets from $pool$ with the highest utilities
    for *each* $t \in proposing_s$ **do**
      $M_{s,t} = 1$
    end
    $tried_s \leftarrow tried_s \cup proposing_s$
    if $proposing_s != \emptyset$ **then**
      $new_proposers \leftarrow new_proposers \cup s$
    end
  end
  // Accepting step: each target will decide which proposers to keep
  for *each* $t \in T$ **do**
    $pool \leftarrow$ set of source apps with $M_{s,t} = 1$ // available source pool
    $accepted \leftarrow MAX$ sources from $pool$ with highest utilities
    $rejected \leftarrow pool - accepted$
    for *each* $s \in rejected$ **do**
      $M_{s,t} = 0$
      $failed_s \leftarrow failed_s \cup t$
    end
  end
end

Algorithm 1: Source-proposing deferred acceptance algorithm

simulate the case of fixed price model); (ii) both source and target optimize usages (to simulate the case of benefit-contingent scheme). The performance results are reported in Section 7.3.

G.2. Scalability of Matching Algorithms

In our data sample, the number of apps is in the order of hundreds and the matching processes take less than a second in our compute server (2.86 GHz CPU). Specifically, in our simulation, `random_match`, `greedy_match`, `src_da_match`, and `tgt_da_match` took only 20, 2.5, 1.1, and 0.9 milliseconds, respectively. However, considering the massive scale of the app market, it is prudent to test the scalability of the proposed matching algorithms. To do that, we artificially generated 10,000 apps (preserving the source and target app proportion from the original data) by replicating the apps from our data sample with some random noises added to the app characteristics. Then we randomly sampled from 10% to 100% from the generated data and ran matching simulations. Figure 9 shows the results on computation times, where X- and Y-axes are the source and target app proportions (0.1 to 1.0) and Z-axis is the execution times in seconds. In all four algorithms, we find that the execution time gradually increases as the numbers of source and target apps increase. In the full data of 10,000 apps, `random_match` and `greedy_match` algorithms took 180 and 80 seconds to produce app matches, respectively. The proposed deferred accept algorithms (`src_da_match`, and `tgt_da_match`) took 16.7 and 2.08 hours for the matching processes, respectively. The different performance between the two algorithms is due to the fact that only up to seven target apps can be matched with one source app simultaneously, while there is no limitation on the number of source apps that one target app can be matched with. As source and target apps enter and leave the CP market in a daily basis, we think that it will be practical to use `tgt_da_match` matching algorithm in a crowded CP market, even with the number of apps is in the order of thousands. With more powerful compute servers, the matching process can be further expedited.



(a) `random_match`

(b) `greedy_match`

(c) `src_da_match`

(d) `tgt_da_match`

Figure 9 **Computation time for matching algorithms with simulated app data**

References

Angrist, Joshua D, Jörn-Steffen Pischke. 2008. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press.

Autor, David H. 2003. Outsourcing at will: The contribution of unjust dismissal doctrine to the growth of employment outsourcing. *Journal of Labor Economics* **21**(1) 1–42.

Heckman, James J. 1979. Sample selection bias as a specification error. *Econometrica* **47**(1) 153–161.

Le, Quoc, Tomas Mikolov. 2014. Distributed representations of sentences and documents. *International Conference on Machine Learning*. 1188–1196.

Lee, Gene Moo, Liangfei Qiu, Andrew B. Whinston. 2016. A friend like me: Modeling network formation in a location-based social network. *Journal of Management Information Systems* **33**(4) 1008–1033.

Mikolov, Tomas, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Nelder, J. A., R. Mead. 1965. A simplex method for function minimization. *Computer Journal* **7** 308–313.

Salton, Gerard, Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* **24**(5) 513–523.

Shi, Zhan, Gene Moo Lee, Andrew B. Whinston. 2016. Toward a better measure of business proximity: Topic modeling for industry intelligence. *MIS Quarterly* **40**(4) 1035–1056.

Shin, Donghyuk, Shu He, Gene Moo Lee, Andrew B. Whinston, Suleyman Cetintas, Kuang-Chih Lee. 2019. Enhancing social media analysis with visual data analytics: A deep learning approach. *MIS Quarterly, Forthcoming* .