AN INFORMATION SYSTEMS JOURNAL OF THE INSTITUTE FOR OPERATIONS RESEARCH AND THE MANAGEMENT SCIENCES

ISR Information Systems Research

Volume 28 • Number 1 • March 2017

Copycats vs. Original Mobile Apps: A Machine Learning Copycat-Detection Method and Empirical Analysis

Quan Wang, Beibei Li, Param Vir Singh

http://pubsonline.informs.org/journal/isre/

INFORMATION SYSTEMS RESEARCH
Articles in Advance, pp. 1–19
ISSN 1047-7047 (print), ISSN 1526-5536 (online)

Copycats vs. Original Mobile Apps: A Machine Learning Copycat-Detection Method and Empirical Analysis

Quan Wang,[a] **Beibei Li,**[a] **Param Vir Singh**[b]

[a] Heinz College, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213; [b] Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Contact: quanw@andrew.cmu.edu (QW); beibeili@andrew.cmu.edu, http://orcid.org/0000-0001-5466-7925 (BL); psidhu@cmu.edu, http://orcid.org/0000-0002-0211-7849 (PVS)

Received: January 30, 2016
Revised: November 7, 2016; March 11, 2017
Accepted: May 5, 2017
Published Online in Articles in Advance: April 2, 2018

https://doi.org/10.1287/isre.2017.0735

Copyright: © 2018 INFORMS

Abstract. While the growth of the mobile apps market has created significant market opportunities and economic incentives for mobile app developers to innovate, it has also inevitably invited other developers to create rip-offs. Practitioners and developers of original apps claim that copycats steal the original app's idea and potential demand, and have called for app platforms to take action against such copycats. Surprisingly, however, there has been little rigorous research analyzing whether and how copycats affect an original app's demand. The primary deterrent to such research is the lack of an objective way to identify whether an app is a copycat or an original. Using a combination of machine learning techniques such as natural language processing, latent semantic analysis, network-based clustering, and image analysis, we propose a method to identify apps as original or copycat and detect two types of copycats: deceptive and nondeceptive. Based on the detection results, we conduct an econometric analysis to determine the impact of copycat apps on the demand for the original apps on a sample of 10,100 action game apps by 5,141 developers that were released in the iOS App Store over five years. Our results indicate that the effect of a specific copycat on an original app's demand is determined by the quality and level of deceptiveness of the copycat. High-quality nondeceptive copycats negatively affect demand for the originals. By contrast, low-quality, deceptive copycats positively affect demand for the originals. Results indicate that in aggregate the impact of copycats on the demand of original mobile apps is statistically insignificant. Our study contributes to the growing literature on mobile app consumption by presenting a method to identify copycats and providing evidence of the impact of copycats on an original app's demand.

History: Vijay Mookerjee, Senior Editor; Amit Mehra, Associate Editor.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/isre.2017.0735.

Keywords: mobile apps • copycats • unstructured data • text analysis • image analysis

1. Introduction

Mobile commerce has grown immensely around the world in the past decade. Much of the industry's revolution is driven by mobile software applications, hereafter referred to as "apps." According to app analytics provider Flurry, the average U.S. consumer spends 2 hours and 42 minutes per day on mobile devices, 86% of which is spent on mobile apps (Khalaf 2014). In terms of monetary expenditures, Apple announced that its users spent $10 billion in the App Store in 2013, with $1 billion coming in December alone (Brustein 2014). The tremendous demand for mobile apps has created significant financial incentives and business opportunities for app developers. As a result, over 1,000,000 mobile apps have been created and offered on each of the two leading platforms, Apple iOS and Google Android. A top-10-grossing iOS game can make approximately $47,000 in daily revenue and over $17 million per year (Koekkoek 2013).

While this enormous economic opportunity has attracted an increasing number of mobile developers to invest in app innovation, it inevitably invites copycat developers to imitate the design and appearance of the original apps. An industry report by *The Guardian* shows that Apple's App Store was flooded with copycat gaming applications (Dredge 2012). Within only a few days after the original game "Flappy Bird" hit the market, the popularity of "Flappy Bird" copycats rocketed to become four out of the top five free apps in the iOS App Store. Currently, one can easily find dozens of "Flappy Bird" knock-offs on any major mobile platform (Bilton 2014). "Flappy Bird" is not the only app with a clone. Many successful apps are closely followed by copycats (Davidson 2014). The copycat phenomenon has become so ubiquitous that "derivative" works not only come from smaller developers—they also come from large firms imitating smaller firms. Large gaming studios such as Zynga have used their ample engineering resources to imitate smaller games such as "Tiny Tower" (Sareen 2013). Large companies also love to parrot other large companies to maintain feature parity (for example, see "Instagram" versus "Vine," Sareen 2013).

Several unique features of the mobile app market have contributed to the prevalence of copycat apps. First, the barrier to entry is low. While developing an app may cost several hundred thousand dollars, it is still significantly lower than the cost of product development in other industries. Second, the app market is still nascent, and there are very few companies that have branding power. Most apps are developed by stand-alone developers or small companies. Because of limited budget constraints, it is not possible for developers to spend money identifying copycats and alerting consumers on their own. Third, while the source codes of an app is protected under intellectual property laws, the idea is not. Finally, a copycat does not imply poor quality. As a result, a consumer may not care whether an app is an original or a copycat.

The copycat app problem has caused significant distress for the developers of original apps. Original app developers have argued that it is difficult enough to create a hit app, but then they have to worry about copycats stealing their sales, damaging their brand, and turning off their customers. Several practitioners and developers have called for the major app platforms to do more to address the copycat problem. Surprisingly, however, there is little rigorous empirical research that shows the impact of copycats on an original app's demand. This lack of research on the impact of copycats is due to two notable challenges. *First*, given the millions of apps, identifying which app is a copycat and which is an original is not straightforward. There is no existing tool available that can identify similarities between apps. To further complicate the problem, there is no clear definition of a copycat in the context of mobile apps, and the extent of imitation could vary greatly. *Second*, it is not obvious whether prohibiting copycats is in the best interests of an app platform. Empirical evidence that supports the original app developers' claims that copycats hinder their sales is absent.

Thus, to address these challenges, this study has two major goals. First, we aim to introduce a novel copycat-detection method that is able to identify different types of copycat apps in dimensions of both function and appearance by using diverse sources of publicly available data. This method must be fast and scalable to accommodate the large scale and variety of mobile apps. Second, we aim to empirically analyze how copycat apps affect the demand for original apps.

To achieve these goals, we formally define copycats based on both app *functionality* and *appearance*. In particular, we refer to the well-established literature on innovation and imitation and define copycat apps as those that provide similar functions to the original app but that are released at a later date. Then, we further classify two types of copycat apps—deceptive and nondeceptive—based on the level of imitation in the app appearance. We validate our method by collecting fine-grained panel data for gaming apps from the iOS App Store and AppAnnie.com over five years. The data contain both structured data, such as download ranks and the characteristics of the apps, and unstructured data, such as textual and graphical descriptions of the apps. We combine various machine learning techniques to analyze the unstructured text and image data to detect the copycats. We evaluate the accuracy of the proposed detection approach by conducting user studies on Amazon Mechanical Turk. The results show that our method can accurately detect the pairwise similarity between apps with over 91.9% probability.

Then, we examine the impact of copycat apps on the demand for original apps using archival data. Our major findings are the following. (1) Copycat apps can be either friends or foes of original apps, depending on the quality and imitation type of the copycats. (2) Nondeceptive copycats, particularly those with high quality, are likely to be competitors of the original app and will cannibalize the sales of the original app. (3) Deceptive copycats, especially those with low quality, positively affect demand for the original app.

Our major contributions are the following. *First*, to our knowledge, our paper is the first to define the copycat phenomenon in the mobile app market. Our proposed novel copycat detection approach, which combines a variety of statistical and machine learning methods, allows both practitioners and researchers to empirically identify copycats on a large scale using multiple and diverse sources of structured and unstructured publicly available data. *Second*, this is the first study to rigorously demonstrate the effect of a copycat app on an original's demand after considering the quality and imitation type of the copycat.

2. Theory Development and Related Literature

Nelson and Winter (2009) define innovation, particularly technical innovation, as the implementation of a design for a new product or the implementation of a new way to produce a product. Innovators, also known as early entrants and first movers, are the firms that initiate the design and implementation of a new product (Massini et al. 2005). Imitators, by contrast, follow the innovator to produce the same product or adapt the innovator's design to provide similar products (Caulkins et al. 2007). Aligned with the literature, we define an original app as a new app produced by app developers that work creatively with an extremely sparse set of clues. A copycat app, by definition, is an imitator that borrows heavily from what has already been produced, although the copycat app can offer incremental improvements on other aspects of a product, such as the user interface or specific competences.

There is considerable empirical research that has investigated factors that may affect the demand of first mover or late entrant and how first mover could deter the entry of the late entrant. Studies have found that the first mover could gain a competitive advantage over the late entrant through technology leadership, preemption of assets/raw materials, and buyer switching costs. For example, if the production process for the product requires learning such that the cost of production goes down with cumulative output, a first mover could get cost advantages over the late entrant (Datta et al. 2013). First movers could also gain a significant advantage when the technology involved is innovative and can be protected through patents (Lee and Mendelson 2007). If the product requires scarcely available raw material, a first mover can gain an advantage by getting better deals from the suppliers (Spence 1979). Products that have high switching costs would provide a greater advantage to a first mover compared to a late entrant (Varadarajan et al. 2008, Lee and Mendelson 2007).

An underlying assumption in all these studies is that the late entrant would always negatively affect the demand of the first mover. In this study, in contrast to past studies, we provide reasoning why copycats may have a positive effect on the demand of the original. Second, we highlight two new factors, quality of the copycat and the imitation type, that may determine the extent and the direction of impact of copycats on the demand of original apps.

2.1. Copycat vs. Original Apps

While the original apps are innovators, they do not always enjoy technological advantages over the copycats. The innovation aspect of an app is primarily the concept of the app that is not protected by patents (Kraaijenzank 2013). As a result, a copycat can imitate the original and beat it by exploiting an overlooked product positioning (added features), undercutting pricing or out-advertising it (Robinson and Min 2002). In this scenario, the original app's developers have asked the platforms for protection of their innovation. Lacking such protection, they argue that it is not worthwhile to innovate. While the developers of original apps have argued that copycats hurt their demand, there is no empirical evidence to support their contention. Theoretically, copycats could potentially affect demand for the original in either direction. Copycat apps compete with the original apps by providing a similar product with almost the same functionality. As a result, copycats can attract potential consumers who would have otherwise chosen the original. However, in theory, copycats can also have a positive effect on demand for the original. The mobile app industry is one where sellers have very limited branding power. Most apps are developed by small companies or lone developers who have limited resources to advertise and raise awareness about their apps. As a result, consumers face considerable ignorance of and uncertainty about the quality and reliability of mobile apps. The fact that an original app is being copied by others may act as a positive signal about the quality of the original app, which may lead to greater sales for the original. Copycats could also potentially help an original's sales because of pooling of advertisement efforts. When a copycat advertises, it not only generates awareness for itself but also for the type of apps that the original and the copycat represent. Hence, demand for the original could benefit from the indirect awareness generated by a copycat's advertising efforts.

For better illustration, consider two types of consumers in the market: (i) type I being aware of the original app but having not yet downloaded it, and (ii) type II being unaware of the original app. Importantly, we would like to note that even for type I customers who are aware of the original app, they may not be able to distinguish between the original and the copycats.

(1) *Substitution effect*:

For type I customers, when they get exposed to promotions by copycats, they can be deceived into thinking that the promotion that they received from the copycat is actually from the original and thus get deceived into installing that app by directly clicking on the promotion. For type II customers, when they get exposed to promotions by copycats, they may not be deceived into thinking that the promotion they received from the copycat is from the original because they are not aware of the original in the first place. However, the fact that they receive promotions from the copycats leads to natural market competition for the original app demand. As a result, this will also lead to a substitution effect from the copycats (in a similar manner as compared to the normal product competition in the market). Therefore, we expect the substitution effect exists for both type I and type II customers.

(2) *Promotional/Advertising effect*:

For type I customers, when they get exposed to promotions by copycats, they may be reminded of the original app and then download it from the app platform. This sort of promotional effect is similar to the "persuasion role" of advertisements (e.g., Ferguson 1974, Tuerck 1978, Leffler 1981), where the exposure to a promotion or advertisement can remind/confirm consumers' interests of the focal product of which they are already aware. For type II customers, when they get exposed to promotions by copycats, although they may not be aware of the original, the exposure to copycats may lead them to explore other similar apps (both in terms of functionality and appearance). Therefore, this may lead these customers to discover the original app during the search exploration process. This sort of promotional effect is similar to the "information role" of advertisements (e.g., Nelson 1974, Leffler 1981), where the exposure to a promotion or advertisement can help customers discover a new product or product features that they were not aware of in the past. (Once they discover the original app during the information search process, type II customers will become type I customers.) Therefore, we expect the promotional/advertising effect exists for both type I and type II customers.

2.2. Imitation Type and Quality of Copycat

We discuss two aspects of copycats that have not been investigated by prior research in determining the impact of imitator on innovator—imitation type and quality of copycat. We classify copycats into two types: deceptive and nondeceptive. A deceptive copycat tries to deceive the customer into thinking it is the original app by choosing an app name and app icon very similar to that of the original. By contrast, a nondeceptive copycat tries to differentiate itself from the original by choosing a name and icon very different from that of the original.

We argue that imitation type and quality of the copycat would determine how the copycat may affect the demand of the original. Being a copycat is not an indicator of low quality. As argued in Section 2.1, copycats can provide better features than the original as they can learn from the original's experience. Mobile app consumers are unlikely to care whether an app is an innovator or an imitator. They are more likely to care about the quality of the app. Consumers look for cues when they face uncertainty about products (Schmalensee 1982), which is prevalent in the mobile app market because of limited branding power. Faced with similar apps the users may take cues from the perceived quality of the apps represented by consumer reviews for the app. As a result, among a set of comparable apps consumers are more likely to pick the one with higher quality. As a result, compared to the low-quality copycat, a high-quality copycat would attract more consumers away from the original. As a result, we hypothesize that, all else being equal, high-quality copycat apps (regardless of deception level) are more likely to negatively affect the demand of the original app because of a greater substitution (competition) effect.

Hypothesis 1. *The effect of the copycat on the demand of the original would be negatively moderated by the quality of the copycat.*

Individuals primarily get mobile apps two ways: (1) through mobile app install campaigns and (2) through searching and installing of an app through the app store (Tiongson 2015). A recent industry report revealed that approximately 40% of users get an app through app stores and the remaining users get an app through mobile app install campaigns (Tiongson 2015). As a result, any app that wants to acquire customers has to advertise. The main way a copycat can attract customers is through extensive mobile app install campaigns. In such campaigns it is a single-click install for the app and a user does not go to the app store where she could get more information about the app. The app store provides detailed information about the apps and typically lists the apps based on consumer ratings and installed base. If a user goes through the app store, she is likely to choose the one with high quality and the one that has been installed by more users. It would be very hard for a copycat to deceive the consumers in this case as the original is likely to have a greater installed base as it came out earlier.

The deceptiveness of the copycat would determine the extent of substitution and promotion effects. The deceptiveness of the copycat would increase the substitution effect because a greater number of people would be deceived into thinking that it is the original and adopting it. This increases the substitution effect over and above what the similarity of features may lead to. However, the deceptiveness of the copycat would also increase the promotion effects. Consumers who get exposed to the copycat's ad may construe it to belong to the original as they may not be able to differentiate between the two. Overall, the deceptiveness of the copycat would hurt the demand of the original because of a higher substitution effect and help the demand of the original because of a higher promotion effect. If the increased substitution effect dominates the increased promotion effect, then deceptiveness will negatively moderate the effect of the copycat on the demand of the original. By contrast, if the increased promotion effect dominates the increased substitution effect, then deceptiveness will positively moderate the effect of the copycat on the demand of the original. This leads to a two-way hypothesis.

Hypothesis 2A. *The effect of the copycat on the demand of the original will be positively moderated by the level of deception of the copycat.*

Hypothesis 2B. *The effect of the copycat on the demand of the original will be negatively moderated by the level of deception of the copycat.*

By jointly examining Hypotheses 1, 2A, and 2B, we can disentangle the impact from four different types of copycats: high-quality deceptive, high-quality nondeceptive, low-quality deceptive, and low-quality nondeceptive.

2.3. Related Literature

Our study is related to the stream of literature on information goods. Specifically, the literature that studies competition between original and pirated goods (e.g., Oberholzer-Gee and Strumpf 2007, Smith and Telang 2009, Ma et al. 2016) and original versions of a good available in a different format (e.g., Chen et al. 2017, Forman et al. 2009, Ghose et al. 2006).

Similar to our study, substitution and promotion effects are the key trade-offs considered across these studies. The findings from these studies have been mixed and depend on which of the effects dominate the other, which depends on the context. In the context of pirated information goods, Oberholzer-Gee and Strumpf (2007) and Smith and Telang (2009) find a positive effect of pirated goods on that of the original suggesting that the substitution effect dominates the promotion effect. By contrast, Ma et al. (2016) find a negative effect of pirated goods on that of the original indicating that the substitution effect dominates the promotion effect.

The literature that studies the interplay between different versions of an information good available through different channels has studied how e-books affect print books (Chen et al. 2017, Forman et al. 2009) or how used books affect new book sales (Ghose et al. 2006). Studies such as Chen et al. (2017) and Forman et al. (2009) empirically investigate the substitution effect and promotion effect of e-books on print books. The promotion effect exists because the availability of e-books affects the overall brand awareness of the book title through affecting consumers' channel preference and online word-of-mouth (Li 2015). The empirical evidence (Chen et al. 2017, Li 2015) shows that the delay of e-book release decreases industry-level book revenue and individual-level book consumption, whereas it boosts print book sales.

Compared with prior literature, in our setting, the copycat phenomenon has its own uniqueness. There are at least four aspects that distinguish our research from the previous research setting. First, the content and appearance of the pirated goods, used goods, and e-versions are often the same as that of the original goods, which does not hold in the mobile copycat app setting. Second, in the previous piracy setting, the authentic goods usually are well-established products or brands. However, in our setting many authentic mobile apps may not be well recognized by customers. Therefore, in the mobile app setting, the authentic goods have much less brand power in the market compared to the traditional piracy setting for information goods; hence, their response to copycats can be different. Third, e-versions and print versions are offered by the same firm and hence even though they compete with each other, a firm cares about the total profit and not the profit for each version. Fourth, the versions of the products are easily identifiable in the case of original versus pirated information goods, e-books versus print, or used versus new goods. This is not true in the case of copycats. Furthermore, in the traditional piracy setting, authentic goods and piracy goods are usually released in different channels (e.g., legal platform versus illegal sites), whereas in our mobile app setting, both the original and copycat apps are released through the same legal channel (i.e., the App Store). Therefore, it can make it even harder for customers to distinguish an authentic from a copycat in such a market, which calls for a more effective strategy for piracy detection and business response. In sum, because of these unique features in the mobile app market, the insight from prior literature might not be directly applied to our case.

3. Data

The research context of this study is the iOS App Store, which is a leading mobile app distribution platform from Apple, Inc. Initialized as the first mobile app store ever, the iOS App Store has grown exponentially since it opened in July 2008. Starting with 500 apps in 2008, the store saw an increase to over 1 million app offerings in 2013. Every day, the store sorts the apps by daily download and daily revenue and organizes the apps on three ranking charts: paid rank, free rank, and grossing rank. The search box in the store allows shoppers to find apps by their app name. Below the search box, the trending search terms are prominently displayed in real time, allowing shoppers to find popular searched phrases and discover more apps. Once the search is conducted, a list of related apps is displayed where each app is presented with the app name, icon, and review ratings. The shopper can download any app on this search result page or click the app icon to go to the app's homepage to find more details.

The data set used in this study is publicly available app information from the U.S. iOS store for the iPhone. It consists of a sample of 10,100 action games by 5,141 developers released between July 2008 and December 2013. We focus on the action games subcategory because games account for the largest proportion of mobile app revenue and action games is among the largest games subcategories. Many novel and famous apps such as "Angry Birds," "Fruit Ninja," and "Clash of Clans" belong to action games. However, we also have collected data on two other subcategories: arcade games and puzzle games. The action games data are used in all of the empirical analyses, while the data on the other two subcategories are used in robustness checks.

Our data contain cross-sectional information on app homepages in December 2013. The collected data include the app name, app icon image, developer's

Table 1. Summary Statistics

Variable	No. of Obs.	Mean	Std. dev.	Min.	Max.
Download price	10,100	0.7815	3.8655	0	349.99
Paid dummy	10,100	0.4810	0.4997	0	1
App age	10,100	26.5970	13.5489	1	65
Rating	10,100	3.4268	1.0041	1	5
Game center dummy	10,100	0.3823	0.4860	0	1
No. of apps by the developer	10,100	7.7479	16.4381	1	113
No. of characters in the description	10,100	872.2314	676.4216	0	3,994
No. of screenshots	10,100	3.6557	1.7782	0	6

Apple ID, paid or free indicator, current download price, file size, average consumer rating, subcategory type, release date, whether the app is connected to Game Center, number of screenshots, app description, and user reviews. Our data also include an app-level daily panel about daily download rank, daily grossing rank, daily download price, and version update since release. Using the daily download rank and daily grossing rank, we approximate the daily download quantity by following the calibration method proposed by Garg and Telang (2013).[1] Although it is ideal to have access to the actual download quantity, it is challenging to do so for the extensive app population. Calibrating sales rank to approximate sales quantity has been widely used in the literature (e.g., Brynjolfsson et al. 2003, Chevalier and Goolsbee 2003). After the download quantity is calculated, we aggregate the daily values and convert the data to a monthly panel. Organizing the data by month is convenient because our panel spans a period of five years. It also helps reduce idiosyncratic errors associated within a short period of time, i.e., the lagged effect of copycat treatment that may last for a few days.

Table 1 presents the summary statistics of apps in December 2013. We see that 48% of the apps in our sample are paid apps and that the remaining 52% are free apps. The average download price is $0.78. The average consumer rating, which has an ordinal scale of 1–5, is 3.43. The average age of the apps, measured by the month count since the release, is 26.60. The Game Center dummy refers to whether an app is connected to the iOS Game Center, which lets users play and share games with their friends. Of the total number of apps, 38% are connected to Game Center. And 6,916 apps have sibling apps—i.e., their developers have published more than one app in the store, while the remaining 3,184 apps are the only app published by their developers.

4. Copycat Detection

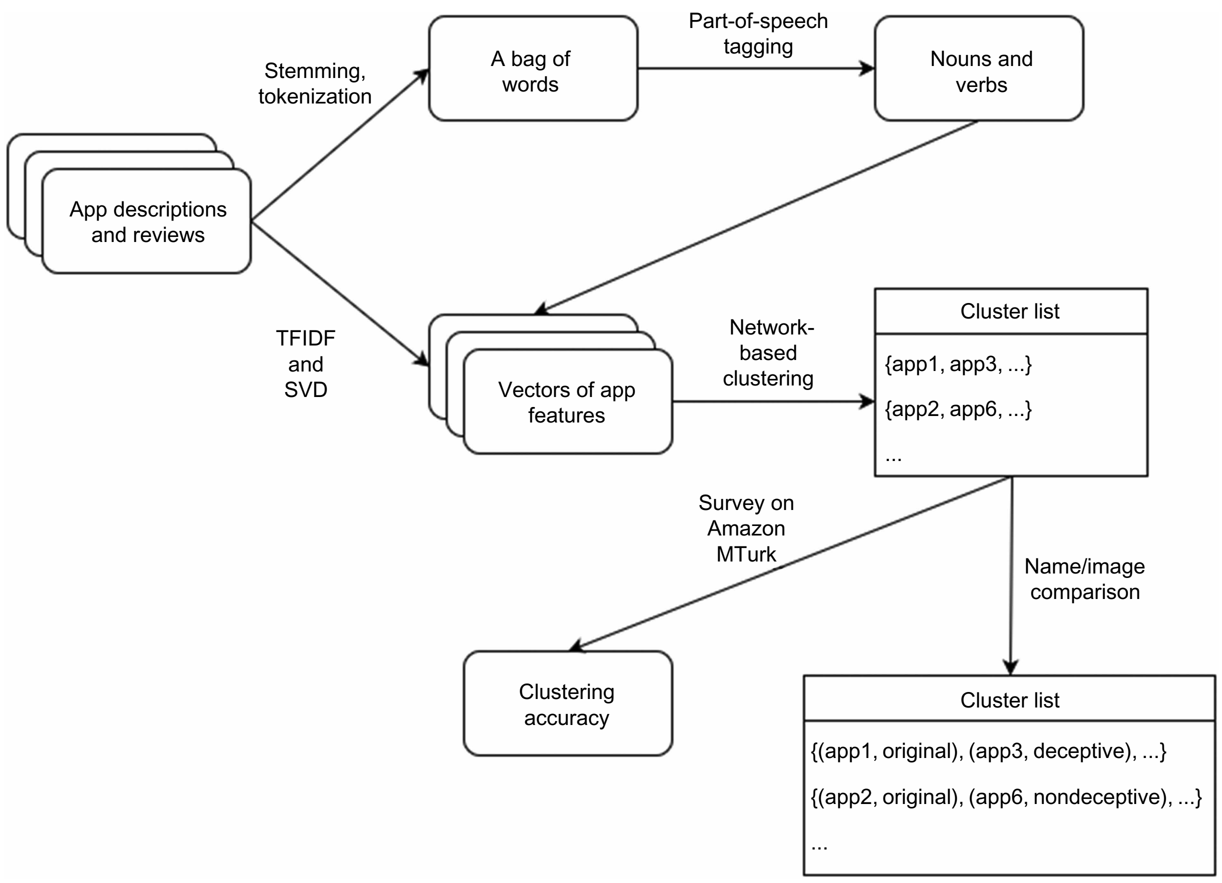
To distinguish copycat apps from original apps, we propose and verify a detection framework that is able to identify copycats based on both functionality and appearance. It is achieved by using a combination of several state-of-the-art machine learning methods. Using this framework, we are able to classify an app into the category of original app, deceptive copycat app, or nondeceptive copycat app. Figure 1 provides an overview of detection flow. In Sections 4.1–4.4, we discuss the intuition of the process. Online Appendix A provides a more detailed technical report of the method.

4.1. Step 1: Detecting Functional Similarity Based on Textual Descriptions and Customer Reviews Using Natural Language Procession

The main purpose of this step is to first map each individual app to a collection of functional features and then measure the app similarities at the feature level. We conduct textual mining on both the app descriptions and consumer reviews to extract the app functionality for each app. First, we convert the textual descriptions and consumer reviews of all of the apps to a bag of words. We then perform text preprocessing, including tokenization, removing stop words, and part-of-speech tagging. Second, we keep the unique nouns and verbs in the bag of words to create a dictionary of app features. We keep only the nouns and verbs because we believe they are more relevant to the app features than other word categories. Third, given an app, we compute the term weights of the app features that are included in its preprocessed textual descriptions and user reviews. The term weights are calculated using the standard term frequency-inverse document frequency (TF-IDF) scheme, which is a measure of how important a word is to a document in a collection (Salton and McGill 1983). By doing so, we map each app to a vector of features. In this vector, each value represents the weighted frequencies of an app feature that appears in an app's description or reviews.

However, the pure TF-IDF scheme is imperfect (Aggarwal and Zhai 2012). The dimensionality of the text representation is very large (in our case, there are 26,642 unique stem words for 10,100 documents), but the underlying data are sparse. Furthermore, the TF-IDF algorithm assumes that the words are independent of each other; it ignores the synonymy, polysemy, and underlying correlations between words. To reduce the dimensionality and the independence between words, we conduct singular value decomposition (SVD). SVD is widely used in large-scale data mining contexts to reduce the dimensionality of feature vectors but preserve the similar structures among vectors (Landauer et al. 1998). Hence, we apply SVD to the TF-IDF vectors. Finally, we calculate the feature similarity between apps by taking a cosine of their feature vectors. The cosine similarity is a value between zero and one that captures the probability of being identical. A larger value indicates that the pair of apps share a stronger functional similarity based on their textual descriptions and reviews.

Figure 1. Flow Chart of Copycat Detection Framework



4.2. Step 2: Network-Based Clustering Using the Markov Clustering Algorithm

The main goal of this step is to identify clusters of apps where the apps within a cluster are similar to each other in functionality but dissimilar to apps not in the cluster. The output of the last step in Section 4.1 can be viewed as an undirected probabilistic graph. In this graph, a node represents an app. The pairwise similarity of apps represents an arc, and the arc is undirected. The value of similarity represents how likely it is that the two connected nodes are the same. Therefore, our goal in this section is to cluster this graph network based on the structure of the network. The expected outputs of step two are clusters of apps where the apps in the same cluster are similar in terms of functionality and gameplay, and apps in different clusters are divergent.

To achieve this goal, we apply a network-based clustering method, which is an unsupervised learning method that allows us to leverage the network structure to extract groups of similar items. In particular, we use the Markov clustering algorithm to cluster our app network (Enright et al. 2002). Once we extract the clusters of similar apps, the next step is to distinguish the original apps from the copycats. We consider the app release date as our standard. If an app is the first app released in a cluster, it is labeled as an original. Otherwise, it is labeled as a copycat app.

4.3. Step 3a: Detecting Appearance Similarity in App Titles Using String Soft Matching

We identify an app as deceptive if *either* its title is similar to the original app's title *or* its icon looks similar to the original app's icon. To classify copycats as deceptive or nondeceptive, we conduct two separate analyses using string soft matching (step 3a) and image-matching analysis (step 3b). To extract similarities in app titles, we conduct string soft matching using the edit distance metrics to compare app names. A smaller distance indicates higher similarity. For each copycat app in a cluster, we compute a pairwise distance between the copycat app name and the original app name. Using a rule-of-thumb cutoff value of 0.7 (Kim and Lee 2012), we are able to find the app pairs with similar titles.

4.4. Step 3b: Detecting Appearance Similarity in App Icons Using Image-Matching Analysis

To detect imitations of app icons, we need an image-matching algorithm that is invariant to the image scale, rotation, change in illumination, etc., because copycat developers may not take the exact same image as the original. However, it is very likely that copycat developers will rescale, rotate, or add noises to the original icon. To address this challenge, we employ the scale-invariant feature transform (SIFT) algorithm proposed by Lowe (1999). The algorithm is one of the most robust and widely used image-matching algorithms based on local features in the field of computer vision (Mikolajczyk and Schmid 2005). It extracts a core set of features from an image that reflect the most important and distinctive information from local regions of the image. After we represent the image by this core set of features, we can match the image with another image, a part of another image, or a subset of the core features extracted from another image. Therefore, SIFT is able to detect similar graphical patterns between images, even when the images have undergone structural transformations.

Using the SIFT algorithm, we match all copycats' icons against the original apps' icons for each cluster of apps. To present some examples of image matching results, we show two pairs of original apps and deceptive copycat apps in Figure 2. The first image is the icon for Angry Birds, which is a famous original game. The second image is the icon of Cut the Birds, which has a similar appearance to Angry Birds but is actually produced by an unrelated developer. The SIFT algorithm recognizes them as a matched pair. The third image is from Plants vs. Zombies, which is also a featured original game. The last image is from Cut the Zombies, which looks like the original game but is offered by a different developer. The algorithm also recognizes these two as a matched pair. Overall, the image-matching process reports 473 copycats whose icons are similar to the original app's icon.

Table 2 provides an overview of our goals, data in use, and methods employed in each step.

Table 2. Summary of Different Methods for Copycat Detection

Goals	Data	Methods
Extract app functional similarity based on textual descriptions and reviews	App textual description, user reviews	• Part-of-speech • TF-IDF • Latent semantic analysis (singular vector decomposition) • Cosine similarity
Cluster apps based on functional similarity	Textual similarity scores derived	• Markov cluster algorithm
Identify original apps versus copycats	Release date, developer ID	—
Identify deceptive versus nondeceptive copycats	App title, app icon image	• String soft matching (edit distance) • Image-matching analysis (scale-invariant feature transform)

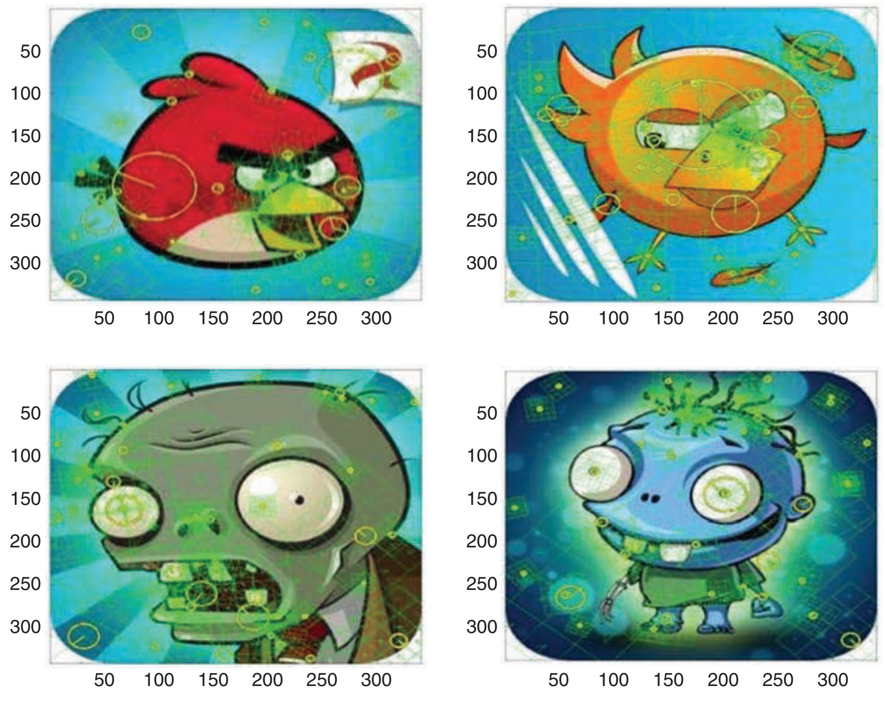


Figure 2. (Color online) Examples of Original Icons vs. Deceptive Icons

Table 3. Example of Extracted App Features

App name	Features
Contract killer	Weapon, contract, mission, killer, gun, crash, enforcer, assassin
Clash of clans	Build, clan, army, clash, defend, cannon, raiders, barbarians
Zombie road trip	Zombie, survival, escape, shoot, gore, death, horde, eaten
Fruit ninja	Slicing, fruit, sword, ninja, juice, blade, hit, dojo, multislice

4.5. Summary Statistics of Copycat Detection Results

This section provides a few summary statistics of the detection results. We start with showing examples of extracted app features. In Table 3, four popular apps—Contract Killer, Clash of Clans, Zombie Road Trip, and Fruit Ninja—are listed together with the extracted features. Comparing the app content with the extracted features, we can see that the features capture the app content accurately.

(1) *Evolution of Mobile App Copycats*
To explore the evolution of the market structure, we take snapshots of the 50 largest clusters over time, as shown in Figure 3. Panels (a)–(c) represent the app market in December 2009, December 2011, and December 2013. In these panels, a node represents an app, and an arc represents a nonzero similarity score between the two connected apps. Figure 3 reveals that the density of the clusters has grown rapidly from 2009 to 2013, indicating a huge growth in the number of copycats over that time.

(2) *Release of New (Original) Apps*
To understand the dynamics of original apps and copycat apps over time, we plot the total amount of apps released over time and the percentage of original apps released over time. Figure 4 is the time trend of the released app number per month. It reveals that less than 50 apps were released per month in 2008 but that this number reached 400 per month in 2013. However, if we look at the percentage of apps that are original among the monthly released apps (in Figure 5), it has decreased from over 90% in 2008 to approximately 45% in 2013. In other words, for every two apps released in late 2013, one is a clone of an existing app.

(3) *Comparison Between Different Types of Apps*
By December 2013, original apps still dominate the existing mobile app market. In our sample of 10,100 action game apps, there are 5,257 (52.05%) original apps, while there are 4,843 copycat apps. Among the

Figure 4. Number of New Apps Released per Month

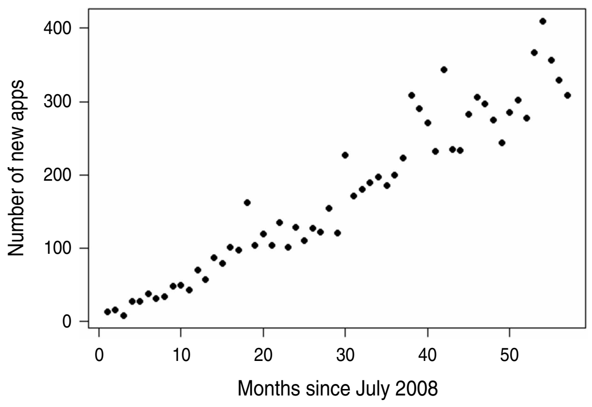


Figure 5. Percent of Original Apps Released per Month

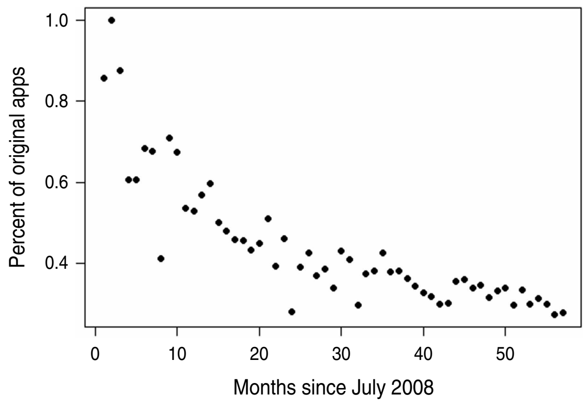


Figure 3. Clusters of Mobile App Copycats Over Time

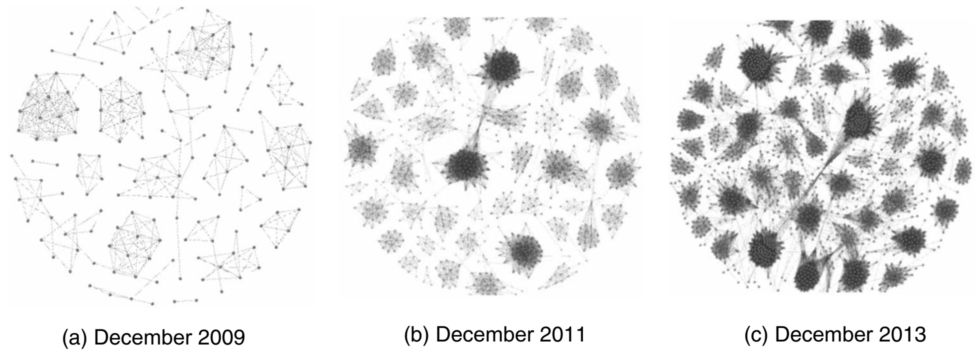


Table 4. Summary Statistics by App Type

Variable	Original with copycat	Original without copycat	Nondeceptive copycat	Deceptive copycat
Paid dummy	0.6567	0.4819	0.4613	0.4497
No. of apps by developer	10.1811	7.8737	7.1241	6.5738
Price	0.9171	0.8427	0.7098	0.6079
Rating	3.4438	3.3541	3.4821	3.5278
App age	37.3502	26.0843	24.5514	26.6745
No. of ratings per month	8.8095	4.1111	7.6127	7.7983
No. of screenshots	3.6335	3.5657	3.7285	3.7987
No. of characters in description	959.0575	822.2600	895.7106	878.2953
No. of apps in this group	1,165	4,092	4,247	596

copycat apps, there are 4,247 (42.05%) nondeceptive copycats and 596 (5.90%) deceptive copycats. The clustering result shows that 3,667 clusters are identified in which 1,720 clusters contain more than one app.

Table 4 reports the mean values of several app features by app type. To compare the original apps with copycat apps more easily, we split the original apps into two groups: originals with at least one copycat app and originals with no copycat apps. According to Table 4, original apps and copycat apps are different in several respects. On average, the download price and the proportion of paid apps are highest for the original apps that have copycat apps, followed by the original apps that do not have copycat apps, and then the nondeceptive copycats and deceptive copycats.

4.6. External Evaluation of Our Copycat Detection Method

We evaluate the accuracy of the copycat-detection method by comparing its results with human-coded data that we collect through an experiment on Amazon Mechanical Turk (MTurk). MTurk is a crowdsourcing web service that assists task requesters in hiring and paying workers to complete tasks that require human intelligence. Typical tasks are well-defined work such as image tagging, sentiment judgment, and survey completion. In machine learning and social science, the MTurk workforce is considered as an important participant pool, constantly available to complete research studies at a high quality and low cost (Heer and Bostock 2010, Paolacci and Chandler 2014).

Specifically, the creation of the MTurk questionnaire is as follows. We start by sampling 1,250 pairs of apps from all possible combinations of apps.[2] Then, we randomly divide the pairs into 250 groups and generate questionnaires using Amazon's self-contained HTML-rendered page called Human Intelligence Task (HIT). Our HIT (Online Appendix B) describes the task and associated compensation. It is only available to qualified workers who are located in the United States and who have a prior acceptance rate higher than 95%. Qualified workers can preview the HIT and decide whether to complete the task. On the HIT, we provide a brief tutorial about the definition of similar apps. Then, we ask the workers to visit the homepages of five pairs of apps and decide whether the pairs are similar. If the pair of apps is considered similar, we also ask if the name, icon, and gameplay are similar. The worker can choose one or multiple options. To avoid any anchoring effect, the sequence of the five app pairs is randomly decided. To control quality, we ask three independent workers to answer the same questionnaire, and the final result is based on a majority vote. To make the worker pool as close to the U.S. iPhone users as possible, we ask the workers to report their most frequently used iOS app and we reject the abnormal answers. If the answer passes the quality control, the worker gets paid \$0.05. A screen shot of our MTurk job description is provided in Online Appendix B.

5. Empirical Model

In this section, we analyze whether copycat apps affect the demand of original apps. If so, we examine under which conditions the displacement is statistically and economically significant. To test the effect of copycats on an original app's download, we conduct analyses on the monthly panel in the following way. We denote the natural log-transformed download of original app i during month t with y_{it}, where $t = 1, \ldots, T$.[3] This is the dependent variable. We denote the natural log-transformed download of copycat apps for the original app i during month t with x_{it}. This is the treatment variable. We model y_{it} as a function of x_{it}, time-varying attributes of the original app D_{it}, time-invariant app-specific heterogeneity captured by the app fixed effects λ_i, time fixed effects φ_t, and other unobserved time-varying variables ε_{it}. Thus, our base model specification is

$$y_{it} = \alpha x_{it} + D_{it}\beta_1 + \lambda_i + \varphi_t + \varepsilon_{it}, \tag{1}$$

where $t = 2, \ldots T$.

The time-varying characteristics of the original app D_{it} include: *Log Original Price*, which is the natural log-transformed monthly average download price of the original app; *Original Version*, which is the count of new version releases during the month; *App Age*, which is the age of the app at month t; *App Age*^{2}, which is the square of the app age; *Log Developer Download*, which is the natural log-transformed monthly downloads of other apps by the same developer; and *Dev Version*, which is the count of the version updates of other apps by the same developer. The scalar α and vector β_1 are the main parameters to be estimated. Product fixed effects (λ_i) control for app-specific time-invariant heterogeneity, including any observable (such as paid/free pricing type and has embedded mobile ads or not) and unobservable time-invariant app characteristics (inherent quality of the app). We use monthly controls (φ_t) to account for time trends. The unobserved error term (ε_{it}) is assumed to be orthogonal to other independent variables.

There are some possibilities of other online and offline promotional activities that affect original app downloads and coincide with copycat downloads. The activities that universally affect all apps in the same way are captured by the time fixed effect. If the activities are app specific and constant over time, they are captured by the app-specific fixed effect. For the time-varying and app-specific activities, we combine two strategies to handle them. Our first strategy is to use the search volume for the app on Google (captured through Google Search Trends) as a proxy of the unobserved marketing mix trend (Ghose et al. 2012). In particular, Google Search Trends works as an index to measure the search volume of the title of an original app on the Internet. A significant proportion of users search through a search engine such as Google when they are looking for an app. As a result, the search volume for the app on Google should be a good indicator of the interest in the app at any time. Although it may not be a perfect measure of time-varying unobserved marketing mix activities, it should be highly correlated with them. To operationalize this idea, we augment the model by explicitly including the Google Search Trends data as a control variable in our model.

Our second strategy is to use appropriate instrumental variables (IVs) for copycat sales. We introduce two different types of IVs. First, we use the lagged copycat downloads as an instrument for the current-period downloads. In particular, we use the lagged terms in three successive months as the set of IVs. The underlying assumption is that the lagged period copycat downloads are not systematically correlated with unexpected changes in current-period unobserved demand shock of the original apps. Longer lags will potentially provide better instruments because more-distant lags are less correlated with the disturbances in the current-period error term, although more-distant lags are more likely to be weakly correlated with the troublesome explanatory variable. In the main analysis, we also use a second type of instrumental variable: the average file size of the copycat apps in the cluster during the current month. A valid instrumental variable should be correlated with downloads of copycat apps but uncorrelated with the time-varying unobserved error term (which might be correlated with the original app's downloads). We argue that the file size of copycat apps can be a valid instrument because it is a cost indicator and reflects the quality of the content. Hence, it is significantly correlated with copycat downloads. Meanwhile, as it is a cost side factor, it is likely to be uncorrelated with the time-varying unobservable demand shocks of the original app (such as marketing mix and word of mouth). To test the validity of the proposed instruments, we use the Cragg–Donald F statistic to test the joint null hypothesis that the proposed instruments are weak and the eigenvalues of the concentration matrix are sufficiently small. If the F statistic is bigger than the rule of thumb of 10, the hypothesis is rejected. Also, as the number of IVs is larger than one, the model is overidentified and we can use Sargan's J test to test whether the instruments are uncorrelated with the error term. If the J statistic is too large and significant at the 5% test level, either one or more of our instruments are invalid. We always observe that the model passes both tests. The F statistic and the J statistic are reported in the estimation results (Tables 5 and 6).

6. Estimation Results

We first analyze the aggregate effect of copycats on the demand of the original. We present the results in Table 5. Model 1 is our baseline model, which includes all of the independent variables and fixed effects; Google Search Trends are added in Model 2; lagged copycat downloads are added as IVs in Model 3; and file size of the copycats are added as IVs in Model 4. The results in Models 1–4 consistently show that the overall effect of the copycat app download on the original app's download is statistically insignificant. A potential explanation for this result is that the negative substitution effect is balanced by the positive advertising effect. Version updates of the original app are positively associated with higher original downloads, while version updates of sibling apps by the same developer is negatively associated with original downloads. Downloads of sibling apps is positively associated with original downloads, perhaps because cross-promotion is likely to occur between sibling apps. Download quantity decreases as app age increases, but the speed of decrease decreases as an app gets older. Finally, the model shows that Google Search Trends has a positive and significant association with download performance.

Next, we investigate how the quality and the imitation type of the copycats affect original apps' demand.

Table 5. Overall Effect of Copycats Apps

	(1) Fixed effect	(2) With Google search trends	(3) With lagged terms as IV	(4) With lagged term and file size as IV
Log copycat apps	0.0029	0.0027	−0.0165	−0.0173
	(0.0135)	(0.0135)	(0.0166)	(0.0166)
Log original price	0.0139	0.0135	0.0141	0.0141
	(0.0110)	(0.0110)	(0.0106)	(0.0106)
Original version	0.6624***	0.6609***	0.5705***	0.5705***
	(0.0408)	(0.0406)	(0.0456)	(0.0456)
Log developer download	0.2251***	0.2248***	0.2091***	0.2092***
	(0.0193)	(0.0193)	(0.0201)	(0.0201)
Dev version	−0.0221***	−0.0223***	−0.0196***	−0.0196***
	(0.0064)	(0.0064)	(0.0061)	(0.0061)
App age	−0.0277***	−0.02791***	−0.0244***	−0.0244***
	(0.0034)	(0.0034)	(0.0034)	(0.0034)
App age^2	0.0002***	0.0002***	0.0002**	0.0002**
	(0.0000)	(0.0000)	(0.0000)	(0.0000)
Log search		0.1103***	0.1036**	0.1037**
		(0.0492)	(0.0409)	(0.0409)
Individual fixed effect	Yes	Yes	Yes	Yes
Time fixed effect	Yes	Yes	Yes	Yes
Adjusted/pseudo-R^2	0.2209	0.2306	0.1302	0.1301
Weak instrument F test			6,239.4	4,106.8
Overidentification J test			3.159	7.622
Number of apps	1,165	1,165	1,165	1,165
Number of observations	109,166	109,166	109,166	109,166

Note. Standard errors are in parentheses.
$^{**}p < 0.05$; $^{***}p < 0.01$.

We split *Log Copycat Apps* into *Log High-Quality Copycats* and *Log Low-Quality Copycats* as two independent variables in Equation (1). High-quality copycat apps refer to copycats that have higher aggregate consumer ratings than the original app, whereas low-quality copycat apps refer to copycats that have lower aggregate consumer ratings than the original app. To examine the effect of heterogeneity in the appearance of copycats, we break down the copycat app download into deceptive copycat download and nondeceptive copycat download. Finally, we want to examine the interactions between deceptiveness and quality. To operationalize this idea, we split the copycat apps into four subgroups: high-quality deceptive copycats, low-quality deceptive copycats, high-quality nondeceptive copycats, and low-quality nondeceptive copycats. We report the results of these analyses in Table 6.

Column (1) of Table 6 shows that high-quality copycats significantly negatively affect the demand of the original. More interestingly, low-quality copycats significantly positively affect the demand of the original. This finding supports Hypothesis 1, which states that the effect of the copycat on the demand of the original will be negatively moderated by the quality of the copycat. The model coefficient should be interpreted as "if the copycat download is changed by 1%, we would expect the original download to change by β%." For example, in column (1) of Table 6, the coefficient of *Log High-Quality Copycat* is −0.0486. It means that a 10% increase in the high-quality copycat downloads results in an average of 0.486% decrease in the original app's downloads. The coefficient of *Log Low-Quality Copycat* is 0.0953. It also means that a 10% increase in the low-quality copycat downloads results in an average 0.953% increase in the original app's downloads. The coefficients for the control variables remain similar in this model compared with Table 5.

Column (2) of Table 6 presents the estimated effects of copycat apps with different levels of deceptiveness. Nondeceptive copycats significantly and negatively affect demand for the original apps. However, the effect of deceptive copycat apps is negative but statistically insignificant. These results indicate that for nondeceptive copycats, the competition effect dominates the marketing effects. By contrast, for deceptive copycats, the competition effect is neutralized by marketing and signaling effects. This finding supports Hypothesis 2A, which states that the effect of the copycat on the demand of the original will be positively moderated by the level of deception.

Column (3) of Table 6 shows the estimated effects where the copycats have been further categorized based on the interaction between quality and copycat type. High-quality, nondeceptive copycats have a statistically significant negative effect on the original

Table 6. Heterogeneous Effect of Copycats Apps

	(1) Quality differences	(2) Deceptiveness	(3) Quality and deceptiveness
Log high-quality copycat	−0.0486***		
	(0.0156)		
Log low-quality copycat	0.0953***		
	(0.0337)		
Log deceptive copycat		−0.0564	
		(0.0480)	
Log nondeceptive copycat		−0.0561***	
		(0.0167)	
Log high-quality deceptive copycat			−0.0641
			(0.0815)
Log low-quality deceptive copycat			0.0948***
			(0.0345)
Log high-quality nondeceptive copycat			−0.0892***
			(0.0156)
Log low-quality nondeceptive copycat			0.0352***
			(0.0121)
Log original price	0.0160	0.0160	0.0161
	(0.0111)	(0.0113)	(0.0111)
Original version	0.6518***	0.6696***	0.6516***
	(0.0407)	(0.0435)	(0.0407)
Log developer download	0.2140***	0.2353***	0.2140***
	(0.0192)	(0.0207)	(0.0191)
Dev version	−0.0233***	−0.02110***	−0.0233***
	(0.0062)	(0.0065)	(0.0062)
App age	−0.0333***	−0.0301***	−0.0334***
	(0.0031)	(0.0034)	(0.0031)
App age^2	0.0003***	0.0002***	0.0003**
	(0.0000)	(0.0000)	(0.0000)
Log search	0.1060**	0.0982**	0.1060**
	(0.0436)	(0.0471)	(0.0436)
Individual fixed effect	Yes	Yes	Yes
Time fixed effect	Yes	Yes	Yes
Adjusted/pseudo-R^2	0.1479	0.1217	0.1480
Weak instrument F test	5,632.4	6,532.1	2,651.0
Overidentification J test	7.023	4.586	11.159
Number of apps	1,165	1,165	1,165
N	109,166	109,166	109,166

Note. Standard errors are in parentheses.
$^{**}p < 0.05$; $^{***}p < 0.01$.

app's sales. However, the negative effect from the high-quality, deceptive copycats is not significantly different from zero. By contrast, for low-quality copycat apps, deceptive copycat apps result in an increase in the original app's sales. However, for nondeceptive copycat apps, the spillover effect is statistically insignificant.

Overall, the results show that when the quality of a copycat is high, the competition effect dominates the awareness effects. By contrast, as the level of deception increases, the awareness effects dominate the competition effect. When both the quality and the degree of deception of the copycat are either high or low, the competition effect is neutralized by the awareness effect. However, when the quality is high, a lower level of deception helps the competition effect dominate the awareness effect. By contrast, when the quality is low, a higher level of deception helps the awareness effect dominate the competition effect.

6.1. Robustness Checks

This section examines the robustness of our findings in two aspects: the accuracy of the proposed machine learning–based copycat-detection method and the robustness of empirical findings. We start by examining the detection accuracy.

6.1.1. Test Copycat Detection Accuracy. *Alternative Text Mining Approach*: *Topic Modeling.* In text mining, there are multiple ways to discover hidden semantic structures in a collection of documents. Topic modeling is one such method. To compare our proposed text mining method with topic modeling, we use

Table 7. Text Mining Evaluation Result

Number of topics	20	100	400	TF-IDF
Number of app clusters	31	881	1,769	3,667
True positive	437	428	437	501
True negative	425	661	682	648
False positive	290	54	33	67
False negative	98	107	98	34
Precision	0.6011	0.8880	0.9298	0.8820
Recall	0.8168	0.8000	0.8168	0.9360
F-measure	0.6926	0.8417	0.8697	0.9080

Table 9. Icon Deceptiveness Evaluation Result

Method	SIFT	SURF	BRISK	KAZE
Number of recognized deceptive app icons	264	257	213	202
True positive	263	256	212	201
True negative	930	930	930	930
False positive	1	1	1	1
False negative	56	63	107	118
Precision	0.9962	0.9961	0.9953	0.9951
Recall	0.8245	0.8025	0.6646	0.6301
F-measure	0.9022	0.8889	0.7970	0.7716

a mainstream topic-modeling algorithm called latent Dirichlet allocation (LDA) to learn the app features. The LDA method posits that each document is a mixture of a small number of topics. The number of topics considered in the model is a hyperparameter determined by the modeler. We experiment with a few numbers. Then, we compare the performance with our proposed TF-IDF approach. Four metrics—true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN)—are used to summarize the differences between the machine output and human-coded results from MTurk. The *F*-measure (e.g., Larsen and Aone 1999) is used to evaluate the performance. The results are reported in Table 7. The results show that the *F*-measure increases from 0.6926 to 0.8697 as the number of preselected topics increases from 20 to 400. By contrast, with our TF-IDF approach, the *F*-measure is 0.908. Previous studies find that the appropriate number of topics is typically between 10 and 100 (Greene et al. 2014). However, even with as many as 400 topics, the result is not as good as our proposed approach. Apparently, topic modeling does not outperform the simple TF-IDF method in terms of leaning the app text.

Alternative Name Similarity Threshold: The 0.7 threshold for edit distance is a rule of thumb value widely used in the text mining literature. To understand whether it is the best-practice threshold to determine name deceptiveness, we try a few candidate thresholds between 0.5 and 0.9. Then, we benchmark the result using the MTurk answers. The name detection results are reported in Table 8. We observed that when the threshold is set to 0.7, the detection method achieves the highest *F*-measure of 0.8661. When the threshold varies between 0.6 and 0.75, the performance is still good. However, when the threshold is set too small or too large, the trade-off between precision and recall results in a decreased *F*-measure. This test supports that 0.7 is an appropriate threshold for our data set.

Alternative Image-Matching Methods: Although SIFT is among the most popular and influential image-matching algorithms, it still needs to be compared with alternative approaches to verify whether it is appropriate for the app image data. To do so, we choose SURF (Speeded up robust features), BRISK (Binary robust invariant scalable keypoints), and KAZE (a Japanese word that means wind) to be the alternative options as they are representative of divergent image-detection frameworks. All four methods are used to find similarity correspondences between the 1,250 pairs of benchmark images. The results, as reported in Table 9, show a few interesting patterns. All four algorithms have small false positive values, meaning that they are unlikely to take unmatched images as matched. Yet they are not as good as human beings at recognizing matched app images, as indicated by the relatively weighty false negative values. SIFT and SURF perform better than BRISK and KAZE, primarily because they can find more deceptive pairs than the other two. SIFT, in particular, has the best precision, recall, and *F*-measure among the four algorithms.

Overall, the external human evaluation dovetails with our machine learning output, thereby indicating that our state-of-the-art methods are able to capture important data properties in a more effective manner than the alternative ones. We briefly discuss why our methods are empirically superior from here on. In the text mining part, the sophisticated topic modeling does not perform better than the classic and straightforward term-based methods such as TF-IDF

Table 8. Name Deceptiveness Evaluation Result

Threshold	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9
True positive	343	377	389	414	430	447	470	495	519
True negative	710	708	705	701	687	646	550	368	200
False positive	192	158	146	121	105	88	65	40	16
False negative	5	7	10	14	28	69	165	347	515
Precision	0.6411	0.7047	0.7271	0.7738	0.8037	0.8355	0.8785	0.9252	0.9701
Recall	0.9856	0.9818	0.9749	0.9673	0.9389	0.8663	0.7402	0.5879	0.5019
F-measure	0.7769	0.8204	0.8330	0.8598	0.8661	0.8506	0.8034	0.7190	0.6616

and Latent Semantic Analysis (LSA). This is perhaps because app descriptions and reviews are short and informal. A document can contain many topics. As proved by Tang et al. (2014), topic modeling is not always good at processing informal and unconventional data sets. In the app name comparison, 0.7 as the string similarity threshold achieves the highest accuracy. This observation is highly consistent with prior literature and indicates that 0.7 might be associated with some latent statistical or linguistic features of the English language and hence captures its patterns better. In the image mining part, SIFT has high quality and stable performance compared with three alternative approaches. This is perhaps because SURF and BRISK are not good at handling blurring and rotation (Juan and Gwun 2009, Leutenegger et al. 2011), and KAZE is less accurate for images with high rotational variances (Andersson and Marquez 2016). Yet these image transformation tricks are very likely to occur in copycat icons.

6.1.2. Test Empirical Model Estimation. *Propensity Score Matching (PSM) and Coarsened Exact Matching (CEM)*: One potential concern with our analysis is that we only consider original apps that have copycats in our regressions. It is possible that the apps that attract copycats are systematically different from the apps that do not. For example, apps with a higher volume of historical downloads might be more likely to attract copycats. To check the robustness of our results to this potential selection bias, we use PSM and CEM. The goal of the matching process is to generate a matched sample to mimic a randomized experiment, where there is no significant difference between the group of original apps that have copycat apps (Treatment) and the group of original apps that do not (Control). We then estimate the weighted difference between the original apps in the treatment group and those in the control group.

To conduct matching, we first identify a set of observable covariates that would potentially influence the treatment decision (original app being imitated by any copycat app). The goal is to match the distribution of covariates so that both the control and the treatment groups will have the same probability of getting treated. In the matched sample, the difference in the outcome between these two groups will be attributed to the treatment only. The influential observable covariates include pretreatment downloads, and various publicly available app characteristics including download rank, download price, description length, number of supported devices, number of screenshots, file size, whether the app is connected to Game Center, the level of the content advisory, and the number of clusters to which the developer belongs. Prior to matching, we also divide the apps into treatment and control groups. The members in the control group are original apps who have no copycat followers during the entire sample period. The members in the treatment group are original apps being followed by at least one copycat during the first three months since release.[4]

In PSM, we use the logit regression and K-nearest-neighbor matching methods to match each treated original app with an untreated original app in the control group that has a closest propensity score. Then, we incorporate the weights into our model. The results are reported in Table 10. However, the PSM method relies on a nontrivial assumption, which is that the true functional form of the assignment follows the logit or probit specification. To release this assumption, we use CEM (Lacus et al. 2011) as an independent and alternative matching method. It matches group members using a nonparametric functional form. The results are reported in Table 10. Comparing both matching results with the main estimate results in Table 6, we see that they are highly consistent. Both the sign and magnitude of the estimated coefficients are similar.

Other Game Subcategories: So far, all of the examinations are conducted on the action games subcategory. To enhance the external validity of the results, analyses are repeated on two other subcategories: arcade games and puzzle games. We do not go beyond game apps to test utility apps primarily because they are less comparable in terms of content, market structure, and user behavior. The copycat phenomenon is also less compelling for the utility apps. In this robustness test, we use a panel of 13,259 arcade games and a panel of 2,514 puzzle games to repeat the copycat-detection and econometric exercise. The results are reported in Table 11. Comparing the results with the findings for action games, we find that the copycat effect persists in the arcade genre and puzzle genre. Arcade games, in particular, show highly comparable patterns to the action games. The coefficient corresponding to *Log High-Quality Deceptive* games stays negative but becomes significant for puzzle games. This coefficient is negative but insignificant for action and arcade games. The coefficients of high-quality deceptive copycats and high-quality nondeceptive copycats for puzzle games are statistically insignificant. This indicates that for puzzle games, when the quality of the copycat is high, the level of deception does not vary its impact on the demand of the original.

Instrumental Variable for Download Price: One potential concern of the main analysis is that the download price is endogenously decided by the developer, meaning that the developer anticipates download changes and adjusts prices accordingly. Although the reason of price change is invisible to researchers, we are able to test for the presence and magnitude of price endogeneity. We introduce an IV for the focal download price—the average download price of other apps that are developed by the same developer. This IV is valid because it is unlikely to be affected by the unobserved

Table 10. Propensity Score Matching and Coarsened Exact Matching

	Propensity score matching			Coarsened exact matching		
	(1) Quality differences	(2) Deceptiveness	(3) Quality and deceptiveness	(4) Quality differences	(5) Deceptiveness	(6) Quality and deceptiveness
Log high-quality copycat	−0.0262**			−0.0046		
	(0.0103)			(0.0199)		
Log low-quality copycat	0.0308**			0.1046**		
	(0.0136)			(0.0284)		
Log deceptive copycat		−0.0176			0.0607	
		(0.0207)			(0.0388)	
Log nondeceptive copycat		−0.0028			−0.0619***	
		(0.0075)			(0.0257)	
Log high-quality deceptive copycat			−0.0121			−0.0514
			(0.0219)			(0.0397)
Log low-quality deceptive copycat			0.0317***			0.1204***
			(0.0105)			(0.0446)
Log high-quality nondeceptive copycat			−0.0209***			−0.0783***
			(0.0082)			(0.0147)
Log low-quality nondeceptive copycat			0.0199*			0.0754***
			(0.0100)			(0.0228)
Individual fixed effect	Yes	Yes	Yes	Yes	Yes	Yes
Time fixed effect	Yes	Yes	Yes	Yes	Yes	Yes
Adjusted/pseudo-R^2	0.1778	0.1756	0.1776	0.1200	0.1169	0.1189
Number of apps	2,328	2,328	2,328	2,439	2,439	2,439
N	77,733	77,733	77,733	77,520	77,520	77,520

Notes. Standard errors are in parentheses. Variables controlled for but not reported above include *Log original price*, *Original version*, *Log developer download*, *Dev version*, *App age*, *App age*^{2}, and *Log search*.
$^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$.

shocks in the download of the focal app, while it is correlated with the focal price because of the shared cost structure. We report the analysis results using the IV of price in Table 11. In this table, the coefficients of download price remain negative and statistically insignificant. The coefficients of the treatment variables are qualitatively the same as not having the price correction. The first-stage *F* statistic is 15.52, which exceeds 10 and indicates that the proposed IV is not a weak instrument. We conclude that the level of bias introduced by the endogenous price is small.

Table 11. Robustness Checks

Variables	Arcade games	Puzzle games	Price corrected by IV
Log high quality deceptive copycat	−0.0151	−0.0610***	−0.0462
	(0.0221)	(0.0059)	(0.0347)
Log low quality deceptive copycat	0.0306***	0.0983*	0.0330***
	(0.0043)	(0.0409)	(0.0026)
Log high quality nondeceptive copycat	−0.0177***	−0.0514*	−0.0193**
	(0.0050)	(0.0290)	(0.0107)
Log low quality nondeceptive copycat	0.0169**	0.0386***	0.0189***
	(0.0075)	(0.0054)	(0.0027)
Individual fixed effect	Yes	Yes	Yes
Time fixed effect	Yes	Yes	Yes
Adjusted/pseudo-R^2	0.2	0.1985	0.1395
Weak instrument *F* test			15.52
Number of apps	4,921	931	450
N	258,524	66,003	20,749

Notes. Standard errors are in parentheses. Variables controlled for but not reported above include *Log original price*, *Original version*, *Log developer download*, *Dev version*, *App age*, *App age*^{2}, and *Log search*.
$^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$.

Potential of Discontinuous Effect: This part aims to test whether there might be a discontinuity in original app downloads as a function of copycat app downloads. The related concern is that the function can have a sharp drop or rise near the point of zero copycat demand. To empirically check this point, we compare the original apps that do not have copycat apps with the original ones that have at least one copycat app. In particular, we match the two using a PSM approach and generate a binary dummy indicating whether the original app has nonzero copycat apps in the current period. The coefficient of this dummy variable captures whether there is discontinuity in the function of original downloads in response to copycat downloads. The results are reported in Online Appendix C. We find that the coefficients of the nonzero copycat download dummy are negative (−0.0204, −0.1105, −0.1455, and −0.1322) in all of the models, but none of them are statistically significant. Therefore, we conclude that the original app download function is smooth near the zero copycat download.

Alternative Definition of Copycat Apps: In the main analysis, we consider copycat apps as developer specific instead of app specific. Using the analogy of the software industry, we are inclined to not take Windows 10 as a copycat product of Windows 8. However, it is still valuable to present how robust our findings would be if we allow similar apps created by the same developer to be copycat apps. Therefore, we redo all of the econometric analysis while accounting for the similar apps by the same developer as copycat apps. The results are reported in Online Appendix C. They show that the estimated coefficients remain highly consistent with the previous models.

7. Conclusion

Although mobile apps are now penetrating many aspects of our lives, there are few research studies in this field. Despite the fact that the copycat issue has been emerging for quite some time, little is known about its causes and effects. How are copycat apps defined and identified? What economic impact do copycats have on the original apps? From the original app's point of view, understanding the effects of copycats can help developers find better strategies to combat or accommodate copycats. From a platform's point of view, the sustainable and healthy development of an app store depends on an understanding of the copycat phenomenon. These factors motivate the need to obtain a deeper understanding of the demand side of copycats and original products in the setting of mobile apps.

In this paper, we propose an automatic machine learning approach to identify copycat apps using large-scale and diverse sources of unstructured textual and image data. We verify the accuracy of our proposed method using human coding though an Amazon Mechanical Turk experiment. We find strong evidence that the proportion of copycats among new app releases has increased dramatically in the action game genre over the past five years. We examine two countervailing effects that copycat apps might have. On one hand, copycat apps compete with original apps by providing similar functionality. On the other hand, copycat apps may have an advertising effect that may help increase awareness of the group of similar apps so that the original app becomes more visible. Interestingly, we find that both effects exist in reality. However, which effect becomes dominant depends on the relative quality and the degree of deceptiveness in the appearance of the copycat.

The managerial implications of our study can be summarized from several different perspectives.

From the app developer's perspective, our study can assist developers in potential legal situations in several ways. *First*, our method provides a way for the original developers (innovators) to identify the potential copycats and competitors in the market, and how different types of copycats affect their original apps' business revenues. Moreover, it provides a way to determine what features of the original apps are likely to be copied and how to leverage such knowledge to better design apps to minimize potential loss in the future. Our scalable detection approach can help in discovering a list of copycat apps that covers more apps than using human eyeballing or heuristic approaches. *Second*, our method has the potential to allow app developers to adopt and seek systematic evidence in app design in any potential legal situation. For example, the original developer can monitor the potential copycat apps over time to determine which ones are more threatening and worth suing. It might not be worthwhile to sue every entity, but our approach allows original app developers to create a more complete list of threatening copycat apps to proactively protect themselves from potential risk in both the short run and the long run. Moreover, we believe that the machine learning–based copycat-detection results can help offer more *objective* evidence to facilitate jury decisions in the lawsuit. Besides, our human-based Amazon MTurk evaluation results show a high consistency between our proposed machine learning–based method and the human eyeballing–based method (with over 90% accuracy). Our methodology evaluation indicates strong confidence of our method in facilitating effective and scalable human decision making with support from large-scale, publically available data. Furthermore, when the copycat app is popular, the textual information on the app description and user-generated reviews will be even richer and help enhance the detection performance. *Third*, our study shows that not all of the copycats are threatening. If certain types of copycat apps have a strong advertising (promotional) effect, then it is possible that the original app developers could craft strategies to profitably leverage its benefits but minimize its damage (e.g., keep a close eye on the sales performance of these copycats in the market without interfering with the promotional efforts from word-of-mouth for a certain period). Potentially, there are specific apps or specific periods during their life cycle when the promotional effects of copycats might outweigh the cannibalization effects. For example, low-quality deceptive copycats in general might not be worth suing; instead, they have promotional effects for the originals. In such cases, while it is important to monitor the trend of copycats in the market, it might be more strategic for the original app developers to focus on active innovation and quality improvement of their own app products.

From an app user's perspective, our method enables the users to not only more quickly find a set of similar apps with similar desired functions or appearance but also identify the original app among a number of similar apps. It has the potential to reduce user search cost in the mobile app market and to provide better guidance for user app recommendation.

From a platform's perspective, our proposed machine learning approach can provide the platform an efficient and scalable way to automatically detect suspected copycat products in the market. This approach will help the platform to better identify and eliminate potential counterfeit products on the platform to improve user trust and the reputation of the platform. Moreover, the methodology we propose in this study has the potential to be generalized beyond the mobile app platform as well. It can benefit other types of platforms that contain large-scale unstructured information in their product descriptions.

Finally, from a policy-maker's perspective, our empirical findings can provide important insights toward how to better regulate markets with potential copycat products to encourage and facilitate innovation in the future. This is an important policy question especially when one tries to understand long-term platform innovation. Our findings suggest a strong heterogeneous effect among different types of copycats. Therefore, policy makers should carefully take into consideration such heterogeneity when designing policies.

Our paper has some limitations that could act as fruitful areas for future research. First, in the copycat-detection section (Section 4), we assumed that a copycat app has one major original app to imitate. Therefore, we did not assign the copycat app to two or more original apps. In reality, a copycat can borrow ideas from multiple original apps, which fosters future research work down the road. Second, users may not fully digest the app content until they actually run it. Hence, the human judgment of app similarity based on the app name, icon, and textual description can still be inaccurate and have underestimated the copycat and original app pairs. Failing to detect unobvious copycat apps can add noise to the external evaluation results. Third, because of data limitation, we do not observe the in-app ads or purchase information. Although we have tried to control for the pricing type of apps by using the app-level fixed effect, we would have deeper insights on the app heterogeneity if we had known the in-app ads and purchase data. Fourth, we would have ideally used actual download quantities to conduct our analysis. However, because of the unavailability of the actual download numbers, we calibrated download quantities from the download ranks. Therefore, we must make several assumptions, including that the overall size of the action games market remains constant over time. Despite these limitations, our paper represents the first study that uses machine learning combined with an economic analysis approach to understand the behavior of copycats and originals in the context of mobile apps, which will help both practitioners and researchers better understand this rapidly growing industry. We hope that our work can pave the way for future research in this important area.

Endnotes

[1] Our estimated shape parameter is −0.9996, which is similar to their reported parameter of −0.944.

[2] Note that because of the scarcity of similar app pairs, a random sampling method does not work in this specific context because very few pairs will appear to be similar under random sampling. As a result, the accuracy measures will not be sensitive to false negatives or false positives. To solve this issue, we oversampled the similar pairs in the external evaluation process. For a robustness test, we tried different sampling methods, and the results stay highly consistent.

[3] Throughout the paper, for all variables, we add 1 to the variable before log-transforming it.

[4] We have also tested other possible values of n (e.g., 6, 12, 24). The results stay very consistent.

References

Aggarwal CC, Zhai C (2012) A survey of text clustering algorithms. Aggarwal CC, Zhai C, eds. *Mining Text Data* (Springer, Boston), 77–128.

Andersson O, Marquez SR (2016) A comparison of object detection algorithms using unmanipulated testing images: Comparing SIFT, KAZE, AKAZE and ORB. Working paper, KTH Royal Institute of Technology, Stockholm.

Bawa K, Shoemaker R (2004) The effects of free sample promotions on incremental brand sales. *Marketing Sci.* 23(3):345–363.

Bilton N (2014) Flappy Bird copycats keep on flapping. *New York Times* (February 24). https://bits.blogs.nytimes.com/2014/02/24/flappy-bird-copycats-keep-on-flapping/?_php=true&_type=blogs&_r=0.

Brustein J (2014) Apple users spent $10 billion on apps in 2013. *Bloomberg Business* (January 7). https://www.businessweek.com/articles/2014-01-07/apple-users-spent-10-billion-on-apps-in-2013.

Brynjolfsson E, Hu J(Y), Smith MD (2003) Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Sci.* 49(11):1580–1596.

Caulkins J, Hartle R, Kort P, Feichtinger G (2007) Explaining fashion cycles: Imitators chasing innovators in product space. *J. Econom. Dynam. Control* 31(5):1535–1556.

Chen H, Hu JY, Smith MD (2017) The impact of ebook distribution on print sales: Analysis of a natural experiment. Working paper, Georgia Institute of Technology, Atlanta.

Chevalier J, Goolsbee A (2003) Measuring prices and price competition online: Amazon.com and BarnesandNoble.com. *Quant. Marketing Econom.* 1(2):203–222.

Datta A, Richard R, Jessup L (2013) Commercialization of innovations: An overarching framework and research agenda. *Amer. J. Bus.* 28(2):147–191.

Davidson J (2014) This is the biggest problem with mobile gaming today. *TechnoBuffalo* (April 23). https://www.technobuffalo.com/2014/04/23/biggest-problem-with-mobile-gaming/.

Dredge S (2012) Should Apple take more action against march of the iOS clones? *The Guardian* (February 3). https://www.theguardian.com/technology/appsblog/2012/feb/03/apps-apple?newsfeed=true.

Enright AJ, Van Dongen S, Ouzounis CA (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.* 30(7):1575–1584.

Ferguson J (1974) *Advertising and Competition: Theory, Measurement and Fact* (Ballinger Publishing, Cambridge, MA).

Forman C, Ghose A, Goldfarb A (2009) Competition between local and electronic markets: How the benefit of buying online depends on where you live. *Management Sci.* 55(1):47–57.

Garg R, Telang R (2013) Inferring app demand from publicly available data. *MIS Quart.* 37(4):1253–1264.

Ghose A, Ipeirotis P, Li B (2012) Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content. *Marketing Sci.* 31(3):493–520.

Ghose A, Smith MD, Telang R (2006) Internet exchange for used books: An empirical analysis of product cannibalization and welfare impact. *Inform. Systems Res.* 17(1):3–19.

Greene D, Callaghan D, Cunningham P (2014) How many topics? Stability analysis for topic models. Calders T, Esposito F, Hüllermeier E, Meo R, eds. *Machine Learn. Knowledge Discovery Databases*, Lecture Notes Comput. Sci., Vol. 8724 (Springer, Berlin Heidelberg), 498–513.

Heer J, Bostock M (2010) Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. *Proc. SIGCHI Conf. Human Factors Comput. Systems* (ACM, New York), 203–212.

Juan L, Gwun O (2009) A comparison of SIFT, PCA-SIFT and SURF. *Internat. J. Image Processing* 3(4):143–152.

Khalaf S (2014) Apps solidify leadership six years into the mobile revolution. *Flurry* (April 1). http://www.flurry.com/bid/109749/Apps-Solidify-Leadership-Six-Years-into-the-Mobile-Revolution#.VCuYffkapjc.

Kim J, Lee H (2012) Efficient exact similarity searches using multiple token orderings. *IEEE 28th Internat. Conf. Data Engrg.*, 822–833.

Koekkoek H (2013) Publication: What is needed for top positions in the app stores? *Distimo* (June 27). http://www.distimo.com/blog/2013_06_publication-what-is-needed-for-top-positions-in-the-app-stores.

Kraaijenzank S (2013) Innovation without patents? Evidence from the smartphone app markets. Working paper, Universidad Carlos III de Madrid, Madrid.

Lacus S, King G, Porro G (2011) Causal inference without balance checking: Coarsened exact matching. *Political Anal.* 20(1):1–24.

Landauer TK, Laham D, Foltz P (1998) Learning human-like knowledge by singular value decomposition: A progress report. Jordan MI, Kearns MJ, Solla SA, eds. *Adv. Neural Inform. Processing Systems*, Vol. 10 (MIT Press, Cambridge, MA), 45–51.

Larsen B, Aone C (1999) Fast and effective text mining using linear-time document clustering. *Proc. Fifth ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (ACM, New York), 16–22.

Lee D, Mendelson H (2007) Adoption of information technology under network effects. *Inform. Systems Res.* 18(4):395–413.

Leffler K (1981) Persuasion or information? The economics of prescription drug advertising. *J. Law Econom.* 24(1):45–74.

Leutenegger S, Chli M, Siegwart R (2011) BRISK: Binary robust invariant scalable keypoints. *IEEE Internat. Conf. Comput. Vision*, 2548–2555.

Li H (2015) Multichannel management with digital products: How e-books affect online and offline print book sales. Working paper, University of Pennsylvania, Philadelphia.

Lowe DG (1999) Object recognition from local scale-invariant features. *Proc. Seventh IEEE Internat. Conf. Comput. Vision* 2: 1150–1157.

Ma L, Montgomery A, Smith M (2016) The dual impact of movie piracy on box-office revenue: Cannibalization and promotion. Working paper, Carnegie Mellon University, Pittsburgh.

Massini S, Lewin A, Greve H (2005) Innovators and imitators: Organizational reference groups and adoption of organizational routines. *Res. Policy* 34(10):1550–1569.

Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Machine Intelligence* 27(10):1615–1630.

Nelson P (1974) Advertising as information. *J. Political Econom.* 82(4):729–754.

Nelson RR, Winter SG (2009) *An Evolutionary Theory of Economic Change* (Harvard University Press, Cambridge, MA).

Oberholzer-Gee F, Strumpf K (2007) The effect of file sharing on record sales: An empirical analysis. *J. Political Econom.* 115(1): 1–42.

Paolacci G, Chandler J (2014) Inside the turk: Understanding Mechanical Turk as a participant pool. *Current Directions Psych. Sci.* 23(3):184–188.

Robinson W, Min S (2002) Is the first to market the first to fail? Empirical evidence for industrial goods businesses. *J. Marketing Res.* 39(1):120–128.

Salton G, McGill MJ (1983) *Introduction to Modern Information Retrieval* (McGraw-Hill, New York).

Sareen H (2013) The evolution of an underground copycat app environment. *Huffington Post* (July 24). http://www.huffingtonpost.com/himanshu-sareen/post_5236_b_3647228.html.

Schmalensee R (1982) Product differentiation advantages of pioneering brands. *Amer. Econom. Rev.* 72(3):349–365.

Smith M, Telang R (2009) Competing with free: The impact of movie broadcasts on DVD sales and Internet piracy. *MIS Quart.* 33(2):321–338.

Spence M (1979) Investment strategy and growth in a new market. *Bell J. Econom.* 10(1):1–19.

Tang J, Meng Z, Nguyen X (2014) Understanding the limiting factors of topic modeling via posterior contraction analysis. *Proc. Machine Learn. Res.* 32(1):190–198.

Tiongson J (2015) Mobile app marketing insights: How consumers really find and use your apps. Think with Google (May). https://www.thinkwithgoogle.com/articles/mobile-app-marketing-insights.html.

Tuerck D (1978) *Issues in Advertising: The Economics of Persuasion* (American Enterprise Institute for Public Policy Research, Washington, DC).

Varadarajan R, Yadav M, Shankar V (2008) First-mover advantage in an Internet-enabled market environment: Conceptual framework and propositions. *J. Acad. Marketing Sci.* 36(3):293–308.