Title
Essays on the Mobile App Platform Choice and Firm Innovation Disclosure

Author
Liu, Yongdong

Publication Date
2015

Peer reviewed|Thesis/dissertation

Essays on the Mobile App Platform Choice and Firm Innovation Disclosure

by

Yongdong Liu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Agricultural and Resource Economics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Brian Wright, Chair
Professor Denis Nekipelov
Professor Minjung Park
Professor Jeffery M Perloff

Spring 2015

Essays on the Mobile App Platform Choice and Firm Innovation Disclosure

Abstract

Essays on the Mobile App Platform Choice and Firm Innovation Disclosure

by

Yongdong Liu

Doctor of Philosophy in Agricultural and Resource Economics

University of California, Berkeley

Professor Brian Wright, Chair

This dissertation combines two papers on industrial organizations and innovations. The first paper focuses on the market evolution in the mobile app platform and the second paper is an empirical study on firm's innovation disclosure and its impact on firm's intellectual property management.

Chapter 1 studies mobile app's platform choice decisions. Ever since Apple and Google launched their mobile application (app) stores in 2008, the market for mobile apps has experienced rapid growth and represents an enormous business opportunity. The success of an app platform largely relies on a great variety of apps, especially innovative and high-quality apps. Given the existence of multiple app platforms, fundamental questions in the app industry are how app developers choose which app platform to enter and which market designs benefit the platform expansion. This chapter studies the platform choice decisions of app developers and the implications for the app market evolution through using a unique and big daily-level panel data set that contains information on every app in the two leading app stores, Apple's App Store and Google Play, over a 2-year period. Combining machine learning techniques for big data problems and computationally efficient econometric approaches, I construct and estimate a structural model for heterogeneous app developers' platform choice decisions within an incomplete-information game framework. I find that in general low-quality apps make the platform less favorable for high-quality entrants. In Google app store, the presence of low-quality apps tends to induce more low-quality apps to enter, while Apple app store exhibits strong competitive effects among high-quality apps. Increasing smartphone user base and improving user engagement are very useful measures to accelerate the platform expansion, but these policies simultaneously encourage many low-quality apps to enter. Regulations on low-quality apps and attenuating competition are more effective on attracting high-quality apps. Platforms can bundle these policies to achieve the optimal market design.

Chapter 2 focuses on an interesting phenomenon in firm's intellectual property management. Owners of knowledge sometimes choose to disclose their private innovation to the public domain, instead of filing patents or keeping them in secret. Such behaviors are called

knowledge disclosure. Once disclosed, the private innovation becomes public knowledge free to use and is no longer patentable. Since it is long believed that private companies take various measures to securely protect their proprietary innovations, the question of how firms benefit from disclosing innovations is worth exploring. Employing a very unique data set of IBM innovation disclosures, I empirically investigate firm's strategic disclosures of private innovation. I further study how such disclosures affect other firm's patented innovation and the focal firm's selective exploitation of follow-up innovation. First, empirical results show that disclosures are not very defensive. They do not undermine citing patents, but lead to stronger citing patents. I also find that IBM discloses relatively low quality innovation on the periphery of its expertise without patenting these. Meanwhile, IBM often discriminatorily cites other firm's patents that are built on its disclosures and are distant from IBM patents assents. This selective utilization results in broader IBM patents and therefore extends its innovation domain.

Contents

List of Figures

List of Tables

Acknowledgments

I am deeply indebted to Brian Wright, Denis Nekipelov and Minjung Park for their guidance and encouragement. Without their tremendous and constant support, I would not be where I am today. I am also very grateful to Jeffery Perloff, J. Miguel Villas-Boas and Przemyslaw Jeziorski for their suggestions.

I am also thankful to my wife, Yiting. She encourages me to pursue my interests and inspires me to be a better person. I am very lucky to have such an amazing wife, best friend and very smart colleague, with who I can discuss research at any time. I am truly grateful to my parents and their unconditional love always powers me. In the end, I would like to thank my friends in Berkeley and I will miss our friendship.

Chapter 1

Mobile App Platform Choice

1.1 Introduction

The market for mobile apps is booming. Since Apple and Google launched their online application stores in 2008, the number of smartphone applications has been growing exponentially. By the end of 2013, more than 1 million apps were available on Google app store (now known as the "Google Play App Store") and Apple app store. In 2013, the total revenue of the Apple and Google app stores was about $15 billion, twice Facebook's annual revenue. Apple's app revenue exceeds its total sales revenue from music, movie and book sales on iTunes. More importantly, the growth in the app market continues. Gartner 2014 estimates that annual app downloads on mobile devices will reach 268 billion in 2017, with a forecast revenue of $77 billion. The very fast growing app market therefore represents an enormous business opportunity.

An app platform constitutes a multi-sided market, connecting app developers, mobile device manufacturers, and consumers. As the expansion of an app platform depends on the presence of a great variety of apps, especially very innovative and high-quality apps, app supply by app developers is crucial for platforms. Given the existence of multiple app platforms, how an app developer chooses an app platform is a fundamental question, because answering this question enables platforms to know which market characteristics and mechanisms benefit certain types of apps. Further, the platform can improve its market designs to optimize the platform evolution. The data used by this research cover the most dynamic phrase of app market from 2011-2013, during which the market size tripled and Google app store developed into another giant comparable with Apple app store. Therefore, the findings and implications in this chapter are particularly useful for new platforms or similar industries experiencing rapid growth.

Intuitively, the platform choice decisions of app developers are based on the trade off between market size and competition, as a platform with more downloadable apps is associated with a stronger network effect, but on the other hand, it can also be more competitive. The countervailing effects of the market size make the platform choice decision less straightfor-

ward, especially for heterogeneous apps. For instance, the same set of market characteristics would have distinctive impacts on very innovative apps and junk apps. Moreover, in a given platform the distribution of app quality is constantly changing, which complicates the platform choice decision.[1]

The evolution of the app market is of substantial economic value, but is relatively under-researched. Hence, this study aims to understand the platform choice decisions of app developers and, in turn, how such decisions influence the evolution of the overall app market. We investigate this using a unique and big dataset that contains information on every app in the two leading app stores, Apple and Google, over a 2-year period. The daily panel data incorporate very rich app characteristics, such as price, version, rating information, developer, ranking, etc. We focus on game apps as these are usually stand-alone, with profits collected mainly by companies in the app industry. Many other apps (e.g. the United Airlines app, the Bank of America app) are developed by companies as tools to facilitate their business in other industries. The platform choice decision for game apps is thus more central.

Based on these data, we construct a structural model for heterogeneous app developers' platform choice decisions within an incomplete information static game framework, adopting the Bayesian Nash equilibrium concept. In each period, a group of potential entrants with pre-determined quality levels arrives, and entrants simultaneously decide which platform (Apple or Google) to enter. By observing the platform characteristics and the incumbent composition, each entrant forms an expectation of other potential entrants' entry probabilities, and further updates its beliefs about the post-entry distribution of app quality for each platform. Based on these expectations and beliefs, each potential entrant makes a platform choice decision by comparing the expected profits associated with each platform.

This structural model on platform choice decisions not only characterizes the evolution of the app market, but also enables us to conduct policy experiments to explore other essential topics in this area, such as how an app platform can better design the marketplace to expand its market size and improve its app composition, which is especially crucial for new platforms such as Microsoft's Windows Phone Store.

There are several challenges to be addressed. The first stems from the enormous quantity of data: about 15 TB, with more than 1 billion observations. During the data period, more than 0.2 million game apps entered either Apple app store or Google app store. The Bayesian Nash equilibrium of the entry game requires that expected actions be consistent with the optimal strategy. Estimation is usually achieved through nesting a fixed-point algorithm in a maximum-likelihood approach. However, in our context, the extremely large entry scale with heterogeneous entrants renders this approach virtually impossible. The computation burden caused by the big data problem is addressed through a computationally efficient econometric method for game estimation(e.g. Aguirregabiria and Mira 2007; Bajari et al.

[1]For instance, in the early stages of the app market, Apple and Google app stores were very unbalanced. Apple's App Store had more than 100,000 apps by the end of 2009, while Google had only 16,000. It is also believed that the former had more high-quality apps than the latter in the first few years. By 2013, however, both app stores had reached the 1-million-app milestone, and now they are competing head to head.

2010). This two-step estimation approach uses a very flexible non-parametric estimation to approximate the equilibrium, instead of solving it by the expensive fixed-point algorithm. The rich information of the big data guarantees a very accurate approximation of the equilibrium. Therefore, the application of the two-step approach on big data problem is not only very computationally efficient, but also becomes a great advantage.

A second challenge arises from the high degree of heterogeneity in app quality. In this chapter, game apps differ from each other, and such heterogeneity is crucial. "Off-the-shelf" sub-genres (e.g. action, puzzle, sports) of game apps and market refinement techniques borrowed from the machine learning literature could be used to group apps with similar functions and reduce heterogeneity. However, game apps in each sub-genre still differ drastically in terms of interface, game plot, and graphic and/or animation design. To handle such heterogeneity, we project apps' high-dimensional heterogeneity onto their quality level (i.e. high and low), and assume that high-quality apps and low-quality apps receive and exert heterogeneous competitive effects. However, app quality is not directly observed in the data. In principle, high-quality apps will tend to be downloaded more, but download numbers can be affected by factors other than quality. For instance, top chart rankings greatly influence app downloads, and established and newly released apps also have distinctive downloading patterns. Therefore, app quality cannot simply be approximated by the number of downloads, leading to the development of an app adoption model[2] to recover app quality (§1.4), which will be an input for the entry estimation.

The third challenge relates to organizing and preparing the big data for analyses. This challenge is addressed through state-of-the-art machine learning methodologies developed by computer scientists. For instance, for estimation identification purposes (detailed in §1.6), we need to identify the cross-platform apps[3] from a large pool of game apps on the two platforms, which involves 60 billion pairs of apps to compare. A highly efficient hierarchical sequential linking schema saves the computation by 70%. Another important data preparation is to define the game app market. In order to precisely characterize platform choice decisions, it is crucial to have a clearly defined market (i.e. game genre) on both platforms for each potential entrant. There are two issues with the data in this respect. First, the "off-the-shelf" game genres in Google are too coarse and are not aligned with the well-defined game genres in Apple app store. Second, many game apps in Apple app store are assigned to multiple genres,[4] which inflates the size of the game market. To address these issues, the primary genre for each app in Apple's App Store that has multiple genres is first identified. Google game apps

[2]Because of the enormous number of apps on the market, search cost is very high for consumers, and not every app will be discovered by potential users. The app adoption process is therefore assumed to be driven by two forces: exposure probability and adoption probability conditional on exposure. The exposure probability is obtained via a sampling process. The conditional adoption problem is a BLP-style (Berry, Levinsohn, and Pakes 1995) demand system where the high-dimensional app heterogeneity is approximated by the mean consumer utility, denoted as app quality.

[3]In this chapter, cross-platform apps are defined as apps that operate on both Apple and Google.

[4]For instance, "Words With Friends" is one of Zynga's most popular crossword game apps. This game is labeled as both a "Board" and a "Word" game.

are then re-categorized in accordance with these better defined game genres in Apple app store. Both steps are achieved by leveraging machine learning techniques of natural language processing and text classification developed in the Computer Science literature: we extract app features from app descriptions, vectorize the extracted features and use the quantified feature vectors as classification variables in the support vector machine (SVM), a newly developed classifier in computer science that is particularly suitable for text classification problems.

This chapter makes several contributions. Theoretically, it provides insights into both app demand and app supply. On the demand side, the app adoption model in this chapter extends the existing research by considering search cost. On the supply side, this chapter adds to the few studies on the mobile app entries. Substantially, this research offers policy recommendations for app platforms regarding various market mechanisms. Methodologically, the integration of state-of-the-art machine learning techniques with structural estimation techniques reduces the computational burden associated with big data, and affords more opportunities in leveraging big data in economic research.

The rest of the chapter is organized as follows. §1.2 briefly reviews related work. §1.3 documents essential data preparation work, including matching cross-platform apps and re-defining markets. §1.4 develops and estimates the app adoption model to infer app's quality. The empirical model of apps' platform choice decisions is developed in §1.5, and §1.6 discusses identification and estimation strategies. §1.7 conducts several policy experiments regarding market designs. Finally, we conclude with a schedule of further steps in §1.8.

1.2 Related Work

This research relates to three streams of literature. First, it builds on the rich literature on market entry (for a complete survey please see Berry and Reiss 2007). The choice of an app platform is most closely related to the location choice studied by Seim 2006, who, to the best of our knowledge, was the first to endogenize product type (i.e. location in her paper) in the entry problem. In Seim 2006, firms make two separate decisions: an entry and a location decision. Entrants are assumed to be homogeneous themselves but are differentiated by location choice. Competitive effects vary across locations, and competitors have incomplete information on the profit function. Entry decision and location choice are characterized as a Bayesian Nash equilibrium, estimated using a fixed-point algorithm nested in maximum-likelihood estimation (MLE). Our research deviates from Seim 2006 in several ways. First, product type has different inherent meanings. Product type in Seim 2006 refers to the store location, while in this research it refers to app quality rather than platform (which is similar to location). Further, Seim 2006 assumes that product type (location choice) is endogenous, while we assume that product type (app quality) is predetermined and exogenous. Given this exogenous quality, apps choose which platform to enter. Second, we allow competitive effects to vary with app type. Third, because of the large market size of app stores, the fixed-point approach is not feasible; instead, we apply a two-step estimation approach.

Thus, this research also relates to the rich body of literature on the estimation of discrete games, originating from Bjorn and Vuong 1984's pioneer work (e.g. Bresnahan and Reiss 1991a; Bresnahan and Reiss 1991b; Tamer 2003; Seim 2006; Sweeting 2009; Pakes et al. 2011). Specifically, this research builds on the two-step estimation approach developed by Aguirregabiria and Mira 2007; Bajari, Benkard, and Levin 2007; Bajari et al. 2010; Pakes, Ostrovsky, and Berry 2007). This approach can be more easily implemented than the fixed-point algorithm, although it tends to converge at a slower rate. The computational burden associated with big data is reduced by applying the two-step approach. Meanwhile, the rich information contained in big data allows very flexible non-parametric estimation for the first step. Therefore, big data are not a burden but a great advantage for the two-step approach.

Finally, this research is also closely related to the nascent but growing literature on the app market. Most papers explore issues on the demand side. To the best of our knowledge, Ghose and Han 2014 were the first to estimate app demand using a structural model. They estimate a BLP-style app demand system using a panel dataset containing the top 400 apps over about a six-month period and find that app demand increases with the length of the app description, the number of screen shots, in-app purchase options, app/version age, and the number of previous versions; conversely, app demand decreases with file size and in-app advertisements. Earlier work examines factors affecting app demand, using descriptive analyses or reduced-form approaches. For instance, applying a reduced-form model, Carare 2012 uses daily data on the rankings of the top 100 Apple apps to study the causal impact of today's bestseller rank on tomorrow's demand. He finds that consumers' willingness to pay is about $4.50 greater for a top-ranked app than for the same unranked app. In addition, the effect of bestseller status on willingness to pay declines steeply with rank on the list, but remains economically significant for the apps in the first half of the top 100 list. Engstrom and Forsell 2013 apply a regression discontinuity design to evaluate how rating stars and numbers of previous downloads displayed to users affect app downloads. As app download data are very hard to obtain, Garg and Telang 2013 propose a method for inferring an app's actual downloads from ranking information publicly available from app stores. Kim 2013 examines how the contributing effects of mobile applications on smartphone adoption differ across smartphone operating systems. The empirical results suggest that Apple's apps provide more benefits to users, but there is no evidence that Google's apps are of lower quality.

This research contributes to the literature on the app market in that it considers both app supply and app demand. To infer app quality, which is central to the entry problem, we build an app demand model that takes into account most of the factors identified by previous researchers. In addition, the app adoption model takes a different perspective on the adoption process: first, built upon existing app demand models, search cost is explicitly incorporated into the app download decision; and second, app demand is modeled as an iterative adoption process in which sales and app performance in previous periods influence app adoption in the current period.

The supply side considers app developers' entry decisions. A small number of papers explore supply-side issues in the app market. For instance, Yin, Davis, and Muzyrya 2014

study the innovation process of "killer" apps and find that cumulative improvement benefits non-game killer apps while developer experience is more important for game killer apps. The study that is the most related to ours is perhaps Bresnahan, Orsini, and Yin 2014 (denoted by BOY hereafter), who also study developers' platform choice decisions. This research differs from BOY in several major respects. First, in BOY, each developer makes a decision independently, while we consider the strategic interactions between developers. Second, we explicitly model competitive effects; in contrast, BOY model market share as a random distribution with parameters to be estimated, but these parameters do not directly reveal the effect of market structures on platform entry decisions. Third, while BOY analyze selected apps and use parsimonious indicators to represent app characteristics, we analyze all game apps and carefully handle app heterogeneity.

1.3 Data

This section first explains the major variables and the basic structures of the data used for this research. Some variables and market information essential for analyses are not included in the original data. The state-of-the-art machine learning methodologies developed in computer science are adopted to prepare the big data, which is also briefly documented in this section.

Data Overview

Two sources of data are combined in this research: app data and global smartphone shipment data. The app data used are provided by a company specializing in app data analysis and consulting, and comprise all Apple and Google game apps from November 2011 to October 2013. They were collected daily from Apple and Google app stores, resulting in a daily panel database. Table 1.1 lists the relevant variables and brief descriptions. Some variables, such as app name and developer name, are nearly invariable over time. Other variables, such as app price and the number of ratings, change on a daily basis. The global quarterly smartphone shipment data for the same period are provided by Gartner, an IT consulting company.

Identify Cross-platform Apps

Developers' experience on the two platforms consists of the important exclusion restriction that is crucial to the identification strategy, which will be discussed in detail in §1.5. To infer developer experience on the two platforms, it is necessary to identify all apps developed by one developer on both platforms.

App name and developer name can be used together to match cross-platform apps. This matching needs to be examined for all possible pairs of Apple and Google game apps, and there are more than 60 billion such pairs. To reduce the exceedingly high computational

Table 1.1: Variables and Descriptions

Category	Item	Description	Unavailability	
			Apple	Google
App Identifier	App Name	App Names shown on phones		
	App ID	String of number or characters uniquely identifying apps		
Developer Info	Developer Name	Developers names shown on phones		
	Developer ID	String of number or characters uniquely identifying developers		X
	Developer URL	Developer webpage (if applicable)	X	
	Developer Email	Developer contact email (if applicable)		
App Detail	App Price	App price and currency unit		
	App Size	App size in kilobyte		
	Release Date	App launch date		X
	Genre	App category		
	App Description	One or several paragraphs of app function introduction		
	Maturity Recommendation	App maturity rating		
	App Screen Snapshot	Images of app screenshots (if applicable)		
	App Video Demo	A video clip introducing the subject app (if applicable)		
	Whether has game center	Indicator whether the subject app is multiple-player game		X
	Related Apps	Apps featured on webpage of the subject app		
Review, Rating and Download	The Number of Ratings	Both daily incremental and accumulative number of ratings received		
	Ratings	The daily and overall ratings		
	Review Contents	The texts of each review made by users		
	Download Intervals	An interval indicating accumulative downloads	X	
Version and Update	App Version/Update	Version number and update date		
	Version Description	What is updated in the new version		
Availability and Compatibility	Require OS	OS required to run the subject app		
	Compatible Devices	Devices required for the subject app		X
	Available Language	The available languages or countries		
Popularity	Ranking on Top Charts	Ranking on various top charts		

This table summarizes the major variables covered in our database, even though this research employs part of these variables. Variables are classified into 7 categories. Several variables are exclusively available in one platform, and the unavailability is marked in the last column.

burden involved, we propose a highly efficient hierarchical sequential linking schema to identify cross-platform apps. This matching algorithm relies on three edit distances that are specialized to detect different aspects of text similarity. The hierarchical schema gives a higher priority to developer name matches, because developer names are more distinguishable than app names. In the algorithm, we first calculate a general and computationally efficient distance for each pair. If the developer names are poorly matched when measured by this general edit distance, the remaining steps are skipped and the pair is labeled "unmatched". If instead the developer names are well matched, we calculate a stricter distance based on both developer names and app names, to further confirm the match. This procedure is repeated until the pair passes all three similarity metrics. This sequential design reduces computation by around 70%. We train the matching algorithm on a small sample to set the matching thresholds, and manually examine matching accuracy for several random samples. The type I error (failure to match) is below 20%, and the type II error (mismatch) is below 5%. The technical details of the algorithm are provided in Liu, Nekipelov, and Park 2014.

Redefine Game App Market

To characterize the platform choice decisions, it is crucial to have a clearly defined market (i.e. game genre) on each platform for each potential entrant. Well-defined markets are crucial to many empirical studies. A market should include mutually competitive agents, and this competition relation may rise from functional similarity or closeness in geographical location. In our case, if irrelevant apps are included in a market, the market competition level would be misleading and could cause serious bias in the analysis of app adoption and platform choices.

In the existing literature, market definition is not usually a problem that requires particular attention. First, most tangible goods already have a clear market boundary (such as cereals, cars and CPUs). Second, in the entry literature, geographical location, such as airline routes, convenience stores within the same zip code area, and plumbers or dentists in relatively isolated cites or towns, naturally define the market.

In our case, however, this issue needs to be addressed. Game apps have very different features and very specific targeted users. For instance, a violent fighting game is very unlikely to compete with a crossword puzzle game. In addition, game apps in the two app stores in the study account for around 0.4 million apps. Given this large market size, a carefully defined market would reduce the number of market players and further reduce the computational burden. Unfortunately, there are problems with the available market definitions. Although Apple and Google each categorize game apps into several subcategories, the two systems are not aligned with each other.[5] There are 18 relatively precise subcategories for Apple games apps, but only 6 very coarse subcategories for Google game apps. The Google game app

[5]In making a platform choice decision, a potential entrant needs to compare the expected profits from each platform. Hence, it is necessary to have perfectly aligned app classifications.

market can therefore be refined by introducing correspondence between its game categories and those of Apple app store. In addition to the issue of alignment, about 80% of Apple's game apps are categorized under two or more genres,[6] which inflates Apple's market size in each genre and creates the misleading impression that Apple app store is more appealing to new entrants.

To address these issues, we first identify the primary genre for each app in Apple app store that has multiple genres, and then recategorize the Google game apps in accordance with Apple's subgenres. Both steps are accomplished using techniques of natural language processing and classification. App descriptions contain detailed information about app characteristics, which helps to identify the primary genre for each Apple game app and allows similar Google game apps to be recategorized. However, textual information needs to be quantified before it can be used for analysis. First, we extract text features from the app descriptions and vectorize these features by counting the frequency. As the occurrence of high-frequency words introduces considerable bias, we apply term frequency-inverse document frequency (TF-IDF) to normalize the feature vectors. Because feature vectors do not have semantic meanings, synonymous terms cannot be clustered and polysemous words cannot be distinguished. Consequently, we apply latent semantic analysis (LSA) to further process the feature vectors. These pre-processed feature vectors are used as classification variables in the support vector machine (SVM), which is a newly developed non-linear classifier that is highly suitable for high-dimensional text classification problems. The parameters in the SVM are determined through grid cross-validation of the training dataset. [7].

Multiple genres of Apple apps and the alignment of Apple and Google genres could both be treated as a classification problem, which can be solved using the support vector machine (SVM). To detect the primary genre of apps with multiple genres, SVM is trained on a sample comprising Apple's App Store games apps that have unique labels. To address the alignment problem, the SVM is first fitted on the Apple apps and is then used to classify the Google apps. Before fitting the SVM, features vectors are extracted from the app descriptions. This section first introduces the steps of feature extraction and then discusses the SVM and the evaluation of the classification results.

Feature Extraction

The simplest way to quantify textual information is to tokenize the text, calculate the frequency of each token in the pool, and store these frequencies in a vector, called the feature

[6]For instance, "Words With Friends" is one of Zynga's most popular crossword game apps in Apple app store. This game is categorized as both a "Board" and a "Word" games; however, the "Word" category is this game's primary market and competition in the primary market should dominate developers' platform choice decisions.

[7]The discrete choice model can also be used for classification purposes. However, compared with other classification tools (such as logit/probit regression, decision tree, random forest, and stochastic gradient descents), the SVM has better non-linear performance in high-dimensional problems, and is especially suitable for text classifications.

vector. The most common approach tokenizes the text word by word, which is called bag-of-words. A contiguous sequence of n terms, or n-gram, can also be applied to tokenize the text corpus. In most cases, however, it is impractical to extract features through the simple vectorization of the text corpus. The biggest concern is the dimension of the feature vector. A moderate collection of texts can contain tens of thousands of different words. Supposing that there are thousands of entries, such a large feature matrix is very difficult to store and analyze.

In the literature of natural language processing, latent semantic analysis (LSA) can extract the most important information from the feature matrix so that it can transform a very high-dimensional feature matrix to relatively low-dimensional feature space. Another issue in feature extraction is the bias introduced by high-frequency words. As feature vectors are usually very sparse, unusual high-frequency words would result in considerable bias. For example, a document on financial systems may contain "bank" or "finance" many times. Although these words are very important tags in such a document, their high frequency in the feature vector would undercut the efficacy of other features. The normalization of the feature vector is thus necessary. Term frequency-inverse document frequency (TF-IDF) is used to normalize the raw feature vector.

Suppose that $D = \{d_i, i = 1, \ldots, N\}$ is the set containing N documents d_i and t is some word included in these documents. Term frequency is then defined as the frequency of t in text d, represented by $tf(t, d)$. Inverse document frequency is the inverse of the number of documents containing term t, which is:

$$idf(t, D) = \frac{N}{|\{d_i \in D : t \in d_i\}|} \tag{1.1}$$

TF-IDF is the product of these two terms, $tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$. To smooth TF-IDF further, logarithms are taken on $tf(t, d)$ and $idf(t, D)$.

Suppose that the normalized feature matrix is represented by an $N \times M$ matrix X with rank r, which is usually very large. We need to find another $N \times M$ matrix X' with a much lower rank, k, to approximate the original high-dimensional feature matrix. Given the matrix X and the rank k, this low-rank approximation is defined as an optimizing problem:

$$X_k^\star = argmin_{X_k} \|X - X_k\|_F \tag{1.2}$$

$\|\cdot\|_F$ is the Frobenius norm, where $\|Y\|_F = \sqrt{\sum_i \sum_j y_{ij}^2}$. $X_k^\star$ is the low-rank approximation of X. This optimized equation can be solved by singular value decomposition (SVD).

For the $M \times N$ matrix X, suppose that the $M \times M$ matrix U consists of standardized orthogonal eigenvectors of XX^T, and that $N \times N$ diagonal matrix V consists of all eigenvectors of X^TX. $M \times N$ matrix Σ is defined as:

$$\sigma_{ij} = \begin{cases} \sqrt{\lambda_i} & \text{for } i = j = 1, \ldots, r \text{ and } \lambda_i \geq \lambda_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Matrix X can then be expressed as:

$$X = U\Sigma V^T \tag{1.3}$$

Proof of Equation (1.3) is straightforward. As XX^T is a symmetric real-valued $M \times M$ matrix, XX^T can be decomposed as $XX^T = U\Sigma^2 U^T$. As V is composed of normalized orthogonal eigenvectors, then $V^T V = I_N$. Therefore,

$$XX^T = U\Sigma^2 U^T = U\Sigma V^T V \Sigma U^T \tag{1.4}$$

From the equation above it is easy to see that $X = U\Sigma V^T$.

Given SVD, the best k-rank approximation can be established by the Eckart-Young theorem(Eckart and Young 1936) below.

THEOREM 1. *Given an $M \times N$ matrix X and its SVD $X = U\Sigma V^T$, the optimal k-rank approximation of X is:*

$$X_k^\star = U\Sigma_k V^T \tag{1.5}$$

Σ_k is defined as:

$$\sigma_{ij} = \begin{cases} \sqrt{\lambda_i} & \text{for } i = j = 1, \ldots, k \text{ and } \lambda_i \geq \lambda_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

and the optimal k-rank approximation error is:

$$min_{X_k} \|X - X_k\|_F = \sqrt{\lambda_{k+1}} \tag{1.6}$$

Theorem 1 states that the best k-rank approximation is the SVD of the biggest k eigenvalues and the corresponding eigenvectors, and the approximation error is the k-th largest eigenvalue. The intuition behind this theorem is straightforward. As eigenvalues may represent the importance of the corresponding eigenvectors, it is natural to use the largest k eigenvalues and their eigenvectors to approximate the original matrix.

Not only does this low-rank approximation reduce the dimensions of the original feature matrix, but the approximation itself also has semantic meaning. Because row i in feature matrix X captures the occurrence of term i across all documents, element (i, j) in matrix XX^T then stands for the co-occurrence of term i and term j. While SVD reduces the dimension of X from r to k, it also concentrates the terms with similar co-occurrences, which is called latent semantic analysis (LSA). LSA has the potential to improve the synonymy and polysemy problem which simple vectorization cannot solve. For example, “leap,” “bounce,” “hop” and “vault” share the similar meaning of “jump.” Latent semantic analysis can gather these terms if the frequency of their co-occurrence is high. Conversely, “bank” could mean either a financial institute or the ground bordering a river. If “bank” occurs together with “finance” or “money” in some documents and with “river” or “lake” in others, LSA can separate these two different meanings.

SVM Classification

There are many models dedicated to classification, such as the discrete choice model, regression trees and the random forest derived from them, and the SVM. The biggest difference between the logit or probit model and other classification methods is that the discrete choice model is a linear model. However, in reality the relationship between input and output may be much more complicated than a linear relationship. Despite logit or probit models being able to accommodate sophisticated specifications, the approximation of a non-linear system with lots of input variables will render the estimation impossible. Regression trees classify different cases by building a tree structure decision process. While easy to interpret and to scale up, regression trees often overfit the training data. Algorithms such as pruning and random forest may correct this over-fitting problem. The SVM is a non-linear classification algorithm and does not have an over-fitting problem. Linearly non-separable data can be dealt with by the appropriate choice of a kernel function. The SVM is widely applied to high-dimensional problems, especially those involving natural language processing.

Supposing that we observe the input x and output y, a typical classification problem can be generalized as $y = 1$ if $f(x) > c$, otherwise $y = -1$. Function $f(x)$ is the classifier and c is the threshold value for the classifier. In the familiar discrete choice model, $f()$ could be a logit or probit function and the threshold value c is usually set to 0.5. The most important question is how to find the classifier $f(x)$. In the discrete choice model, the choice probability derives from the underlying behavior model which compares the utilities of different alternatives. Given the distribution assumption of the idiosyncratic component in the utility function, it is easy to derive the choice probability, which can also be considered to be the classifier. In the SVM, there is no such behavior model. Instead, the idea is simply to find the boundary which can best separate different groups. Suppose that $\{(x_i, y_i)\}$ for $i = 1, \ldots, n$. $x_i \in R^D$ is the input vector and $y \in \{-1, 1\}$ is the variable representing alternatives. The classifier in SVM can be refined as:

$$f(x) = w^T x + b \tag{1.7}$$

where w and b are two unknown parameters in the classifier. As x_i is a point in a n-dimensional space, $f(x) = w^T x + b$ actually is a hyper-plane in this n-dimensional space. The essence of SVM is finding the hyper-plane that could separate different groups. Without certain constraints, there would be infinite such hyper-planes; therefore, this hyper-plane needs to separate the closest points from different groups as far as possible. In other words, SVM maximizes the margins to the closest support points. Let $d(x_i, f(x))$ be the distance from point x_i to the hyper-plane $f(x)$. The SVM can be characterized by the optimizing problem below.

$$\begin{aligned} &\max_{w,b} \min_i d(x_i, f(x)) \\ \text{such that } \ &y_i f(x_i) \geq 0, \forall i = 1, \ldots, n \end{aligned} \tag{1.8}$$

It is easy to see that the hyper-plane $f(x)$ is only characterized by these closest points, and x_i such that $i = arg\min_i d(x_i, f(x))$ is called the support vector. $d(x_i, f(x)) = \frac{|w^T x_i + b|}{||w||}$

is invariant with a proportional change in w and b, which indicates that the distance should be normalized. For $i = arg\min_i d(x_i, f(x))$, let $|w^T x_i + b| \equiv 1$. The objective function in (1.8) then becomes $\max_{w,b} \frac{1}{||w||}$, which is equivalent to $\min_{w,b} \frac{1}{2}||w||^2$. The original optimizing problem then becomes:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||w||^2 \\ \text{such that } \ &y_i f(x_i) \geq 0, \forall i = 1, \ldots, n \end{aligned} \tag{1.9}$$

This is a quadratic programming problem and a global minimum is guaranteed. In many practical implementations, however, data are not linearly separable. Slack variables are thus added to accommodate data irregularity.

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||w||^2 + C\sum_i \epsilon_i \\ \text{such that } \ &y_i f(x_i) \geq 1 - \epsilon_i, \forall i = 1, \ldots, n \end{aligned} \tag{1.10}$$

C is the punishment parameter, which is often tuned according to the cross-validation. Some non-linear kernel function can be applied to x, such as the commonly used Gaussian radial basis function kernel (RBF kernel), $k(x, z) = \exp(-\gamma||x - z||)$. In this case, the separating hyper-plane becomes curvy. The feature dimension of the RBF kernel is infinite, which scales the SVM up to very large samples with a large number of features. The punishment parameter C and the RBF kernel parameter γ need to be tuned according to cross-validation and the hyper-plane parameters w and b, and the slack variable ϵ_i can be solved from the optimization. Once we obtain the classifier $f(x)$, the prediction will be obtained according to the sign of $f(x)$ on a test sample.

Performance Evaluation

Two aspects of the performance of the classifier are evaluated. First, a good classifier needs to collect as many entities of a true type as possible. Second, misclassifications should be eliminated. In other words, the perfect classifier should recognize all of the true types and exclude any samples not of that type. Suppose that $G = \{(g_i, j_i), i = 1, \ldots, N, j_i \in \{0, 1\}\}$ is the set containing entities g_i with type 1 and type 0. A classifier $F(g_i) \in \{1, \ldots, J\}$ maps entities to the binary, and $F(g_i) = 1$ means that g_i is classified as type 1. The detection ability of this classifier is measured by the recall:

$$recall = \frac{\sum_i F(g_i) \cdot 1(j_i = 1)}{|\{g_i, j_i) \in G : j_i = 1\}|} \tag{1.11}$$

The misclassification is measured by the precision:

$$precision = \frac{\sum_i F(g_i) \cdot 1(j_i = 1)}{\sum_i F(g_i)} \tag{1.12}$$

However, the recall and the precision contradict each other. A strict classifier tends to have high precision but low recall, while a slack classifier is the opposite. Therefore, only targeting one measure usually returns a very poor classifier. The F1 score balances these two measures and attempts to reach a trade-off:

$$F1 = 2 \times \frac{precision \cdot recall}{precision + recall} \tag{1.13}$$

$F1$ score is restricted to $(0, 1)$; the higher the score, the better the classification performance. $F1$ score can only be used to evaluate a classifier's performance on a training data set. The best use of the $F1$ score is to guide model section in cross-validation.

Redefine Apple Market

There are 18 genres of Apple App Store game apps, describing a very detailed spectrum of games, but Apple game apps are usually classified into more than one genre. Table 1.2 shows that about 80% of apps are classified into more than one genre, with the majority classified into two genres. While multiple genres may reach more potential users, in most cases the primary genre is obvious. For instance, "Words With Friends" is Zynga's very popular crossword game app on Apple's App Store. This game is classified as both a "Board" and a "Word" game, but the primary market is as a "Word" game and the competition in this primary market should dominate developer's platform choice decisions.

The SVM can be used to distinguish an app's primary genre from secondary genres, but first needs to be fitted to a training data set where the primary genre is obvious. 21% of Apple's game apps are uniquely assigned to a single sub-genre and hence are used for the SVM training sample. The estimated classifier is also used to predict the primary sub-genre of the apps with multiple genres.

In our data, the number of multiple genre combinations is as high as 1879. Each combination needs to train an individual SVM. Moreover, a combination of more than 2 genres is a multiclass classification problem. Not only does the training sample involve more apps, but the complexity of the SVM is also more than doubled due to the multiclass nature. However, the incidence of complex combinations is very low. Most multi-genre cases are 2-genre or 3-genre combinations. To reduce the computational burden, SVM is only trained on 2-genre combinations with more than 100 samples or 3-genre combinations with more than 30 samples, which covers 196 combinations and accounts for 94% of the apps with multiple genres. The primary genre of other multi-genre apps is randomly selected. Table 1.4 compares the distribution of apps with the original multiple genres and the result of SVM classification. In general, the SVM preserves the original distribution across genres.

The performance of the classifier is evaluated by the F1 score. A higher F1 score means fewer errors are made by the classifier. Figure 1.1 shows the histogram of the F1 score for the 196 genre combinations, calculated for the training sample during cross-validation. The distribution is skewed to the right and more than 50% of the SVMs have F1 scores of 0.9 or higher.

Table 1.2: Distribution of Apple's Game Apps Cross Combinations

Combinations	Combination Distribution		App Distribution	
	Freq.	Percent	Freq.	Percent
1	18	0.95%	39789	21.01%
2	153	8.07%	135323	71.46%
3	616	32.47%	11350	5.99%
4	756	39.85%	2445	1.29%
5	265	13.97%	361	0.19%
6	60	3.16%	68	0.04%
7	19	1.00%	19	0.01%
8	8	0.42%	8	0.00%
9	2	0.11%	2	0.00%
Total	1897	100.00%	189365	100.00%

This table reports the distribution of apps assigned to multiple genres. The second and the third columns list the distribution of different genre combinations. The fourth and the fifth columns report the distribution of apps in these genre combinations.

Redefine Google Market

Unlike Apple app store, Google app store only has six very broad genres for game apps. Table 1.3 lists the distribution of Apps across these genres. Although each Google game app is uniquely assigned to only one genre, some genres, such as "Casual" and "Brain & Puzzle" are so coarsely defined that they may contain very differentiated games In addition, Google's genres are not aligned with Apple's App Store genres. Classifying Google game apps according to Apple's App Store game genres, which cover a wide spectrum of game apps, would align the apps on the two platforms and break the broad Google genres down into finer genres.

As there are 18 Apple game genres, a convenient approach would be to disregard Google's existing genres and build an 18-class SVM to classify all of the Google game apps. However, training an SVM with such a complex structure would be very costly. Even if the SVM could be successfully trained, its performance would not be satisfactory. Binary classification usually has better performance than multiclass classification, and multiclass classification

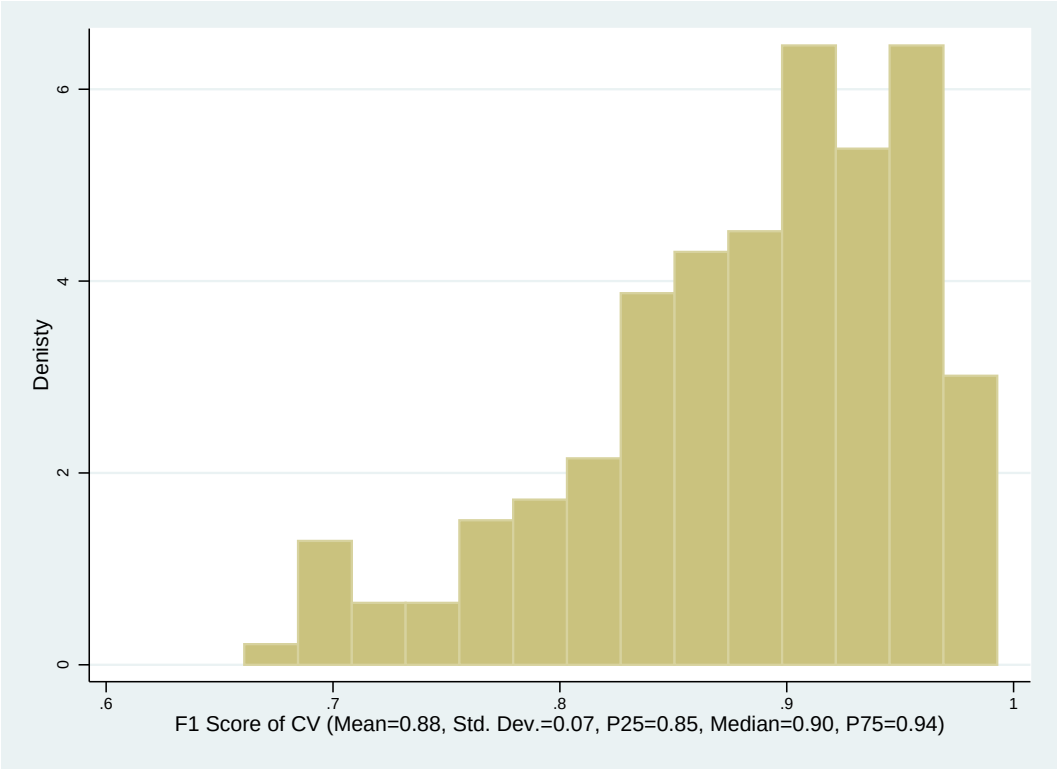


Figure 1.1: Apple App Classification F1 Score from Cross-Validation

with fewer classes is more reliable than a bulky multiclass classification. To optimize this classification problem, we first find the potential classification genres for each Google genre from the cross-platform apps and then implement the SVM independently for each Google genre. Table 1.3 lists the corresponding Apple genres for each broad Google genre. The final column in Table 1.3 shows the F1 score obtained from cross-validation. The SVM results are given in Table 1.4. Google app store has many more Puzzle apps than Apple app store, but otherwise the game genres are similarly distributed in the two platforms.

Table 1.4 lists the distribution of game apps across genres after market re-definition. The "Multi-genre" column gives the genre distribution of the raw data, and the "Unique" column only shows Apple game apps that have a unique genre. "SVM Results" gives the new distribution after market re-definition. For most genres, the distribution of Apple game apps is similar before and after market re-definition. In general, the app distributions of the two platforms are in line with each other, except for Google having more puzzle apps. "Puzzle", "Action", "Arcade", "Family" and "Educational" are the top genres, accounting for more than half of all game apps.

Table 1.3: Distribution of Google Game Apps Cross Genre and Corresponding Apple Genres

Genre	Freq	Percent	Corresponding Apple Genres	SVM F1 Score
Arcade & Action	39517	25.50%	Action, Adventure, Arcade	0.71
Brain & Puzzle	55587	35.80%	Board, Educational,Family, Puzzle,Strategy, Trivia, Word	0.81
Cards & Casino	8661	5.60%	Card, Casino	0.94
Casual	37287	24.00%	Action, Adventure, Arcade, Educational,Family, Puzzle, Role Playing, Simulation, Strategy, Trivia	0.70
Racing	6398	4.10%	Racing	-
Sports Games	7744	5.00%	Sports	-
Total	155194	100.00%	-	-

This table reports the app distribution cross original Google game genres. The fourth column lists the Apple genres for classification, which is determined by the corresponding genres revealed in cross-platform apps.

Descriptive Statistics

To better demonstrate the platform growth during the data period, Figure 1.2 shows the numbers of downloadable game apps on the two platforms and the global smartphone shipments in the 2-year period of the study. From 2011-2013, the game market size in both Apple and Google tripled. Before 2012, Google game app fell far behind Apple, but Google starts to catch up since 2012. The gap between the two platforms in terms of game apps is gradually narrowing during the course of the data period. Notably, however, Android phone shipments are growing much more quickly than iPhone shipments, widening the gap between smartphone user bases on the two platforms.

In order to examine the market evolution cross heterogeneous apps, apps are separated into three groups according to the number of ratings. Figure 1.3 plots the ratio of the market size in October 2013 to the market size in November 2011 for each rating group. During the two-year period, apps without any ratings increased by 7.5 times in Google, but that ratio in Apple is only 3.4. Google also experienced a rapid growth in the high-end segment. The market size of apps with more than 1000 ratings became 6 times larger in Google, but during the same period Apple's high-end segment only doubled. Apps receiving medium ratings (less than 1000 ratings) increased by a similar scale in two platforms. The market evolution in two platforms exhibits very distinctive patterns. Namely, in Google the influx of low-type apps was accompanied with a great increase of high-type apps and the bar graph is in the "U" shape. On the contrary, low-type apps increased the most in Apple and high-type apps had the smallest growth. The increasing ratio is decreasing with the rating groups.

Table 1.5 reports the distribution of apps in three rating groups at November 2011 and

Table 1.4: Distribution of Game Apps across Genres

Genre	Apple						Google	
	Multi-genre		Unique		SVM Results		SVM Results	
	Freq.	Percent	Freq.	Percent	Freq.	Percent	Freq.	Percent
Action	48305	13.50%	2967	7.50%	24683	13.00%	17108	11.0%
Adventure	25251	7.10%	1384	3.50%	9036	4.80%	3388	2.2%
Arcade	48517	13.60%	4588	11.50%	28917	15.30%	20087	12.9%
Board	19038	5.30%	1070	2.70%	6835	3.60%	3473	2.2%
Card	9084	2.50%	998	2.50%	5756	3.00%	4680	3.0%
Casino	6915	1.90%	536	1.30%	4143	2.20%	3832	2.5%
Dice	3252	0.90%	179	0.40%	1249	0.70%	-	-
Educational	20097	5.60%	6208	15.60%	12017	6.30%	11080	7.1%
Family	33351	9.40%	8760	22.00%	19038	10.10%	11815	7.6%
Music	5431	1.50%	679	1.70%	2575	1.40%	136	0.1%
Puzzle	52792	14.80%	7153	18.00%	38068	20.10%	53184	34.3%
Racing	9412	2.60%	757	1.90%	5723	3.00%	6097	3.9%
Role Playing	8901	2.50%	575	1.40%	3357	1.80%	1264	0.8%
Simulation	14354	4.00%	1054	2.60%	5200	2.70%	2040	1.3%
Sports	12409	3.50%	856	2.20%	6276	3.30%	7428	4.8%
Strategy	19834	5.60%	741	1.90%	5376	2.80%	3544	2.3%
Trivia	10933	3.10%	866	2.20%	5947	3.10%	4446	2.9%
Word	8817	2.50%	418	1.10%	5169	2.70%	1592	1.0%
Total	356693	100.00%	39789	100.00%	189365	100.00%	155194	100.0%

The last two columns list the app distribution of Google apps cross SVM classified game genres. The rest columns are for Apple apps. "Multi-genre" refers to the distribution of app assigned to multiple genres. "Unique" is the distribution of uniquely labeled apps.

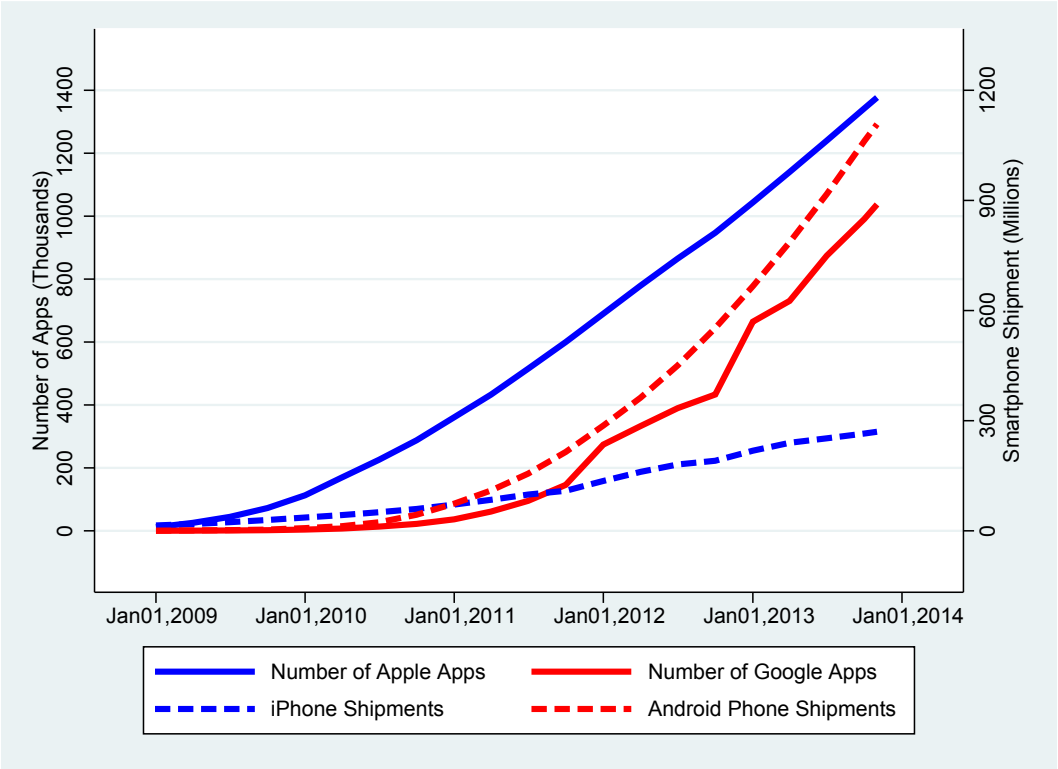


Figure 1.2: The Evolution of Mobile App and Smartphone Market Size

Phone shipments data are from Gartner and on quarterly basis. The number of apps is on monthly basis. The lines of smart phone shipments are smoothed.

October 2013 respectively. In general, app compositions in Apple are very stable during the data period, but the share of apps with more than 1000 ratings dropped by 2 percentages. Since popular apps only account for a very small share, the 2-percentage drop actually is very significant. App compositions in Google changed drastically. On the one hand, high-end apps increased steadily, but on the other hand app compositions deteriorated rapidly in low-end and middle level segments. For example, the share of no rating apps skyrocketed and accounted for more than half of entire apps. Meanwhile the share of middle class apps shrunk greatly. Google's market evolution shows a tendency of bipolarization.

1.4 App Quality Inference

For apps even in the same category, they are still different on many dimensions, but some app characteristics are either hard to measure or not observable at all. However, it is crucial to consider heterogeneous apps in the platform choice decision, since the same set of market characteristics probably brings very different or even opposite impacts on heterogeneous apps. We assume app's high dimensional heterogeneity could reflect on one single variable

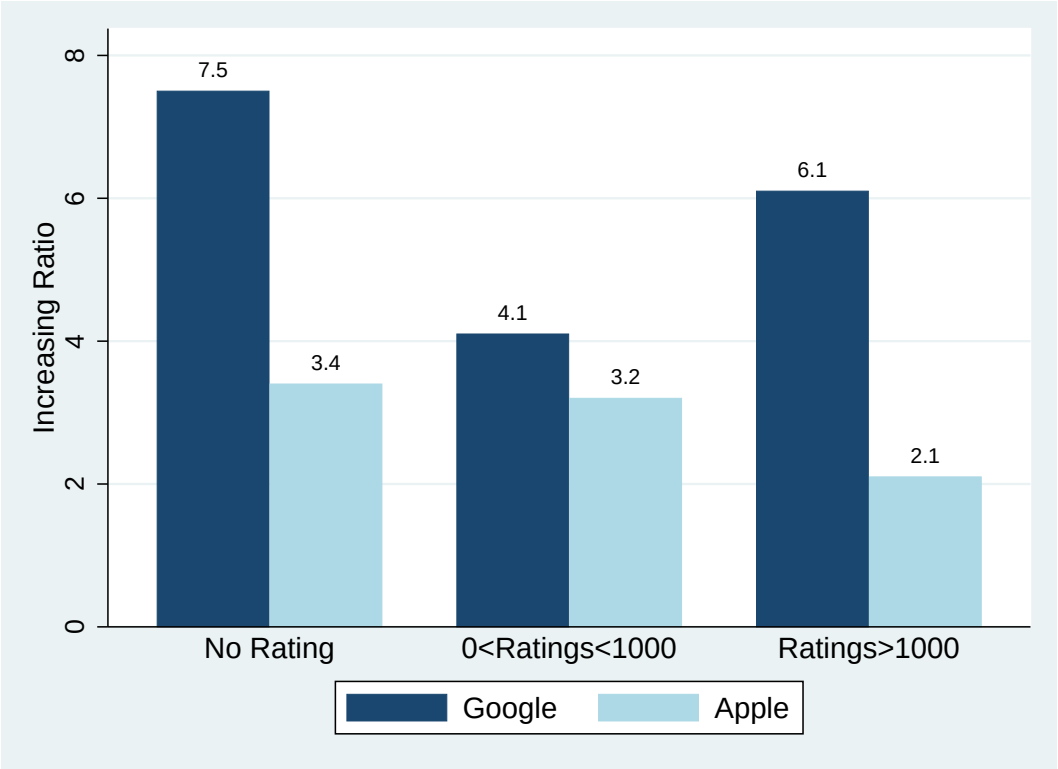


Figure 1.3: The Evolution of Mobile App by Rating Group

This figure plots the ratio of the number of apps at the end of data period to the number of apps at the beginning of data period in different rating groups. Therefore, it shows the market expansion in different market segments.

Table 1.5: Apps Distribution by Rating Groups (%)

App Group	Apple		Google	
	Begin	End	Begin	End
No Rating	65.5	68.0	39.4	53.2
0<Ratings<1000	28.4	28.0	55.3	41.0
Ratings>1000	6.2	4.0	5.3	5.9

"app quality".

App quality is central to the platform choice model, but is not directly observed in the data. In principle, high-quality apps tend to be downloaded more, but download numbers can be affected by factors other than quality. For instance, top chart rankings greatly influence app downloads, and established apps and newly released apps reveal distinctive downloading patterns. Therefore, app quality cannot simply be approximated by the number

of downloads, and hence we develop an app adoption model in this section to recover app quality.

App Adoption Model

Because of the enormous number of apps on the market, the search cost is very high for consumers, and not every app will be discovered by potential users. Therefore, the app adoption process is assumed to be driven by two forces: app visibility and adoption probability conditional on exposure. The app visibility is a probability modeled by a sampling process. The conditional adoption problem is a logit style demand system (see Ghose and Han 2014; Kim 2013), where the high-dimensional app heterogeneity is approximated by the mean consumer utility, denoted as app quality.

We start by introducing the consumer utility associated with apps. Based on this utility, we then discuss in turn the discovery process and the conditional adoption process. The actual adoption model combines the two processes.

Consumer n's indirect utility from downloading app i is:

$$u_{ni} = \zeta_i + \epsilon_{ni}, \tag{1.14}$$

where ζ_i is the average utility that users get from app i and is normalized by outside goods. Hereafter, this average user utility stands for app quality. ϵ_{ni} is a user-app-specific idiosyncratic shock following an i.i.d. type I extreme value distribution.

Discovery Process

The process of app discovery can be viewed as a sampling process. Because of the enormous number of apps, there is very little difference between sampling with replacement and sampling without replacement. For simplicity, we assume that the process is sampling with replacement. The equation (1.15) represents the discovery probability for app i in period t. If all apps were homogeneous, in a market with I apps each app would have an equal discovery probability of $\frac{1}{I}$. However, due to heterogeneity in app characteristics and performance, discovery probability in fact varies greatly.

$$p_{it} = \frac{exp(v_{it})}{\sum_{l=1}^{I} exp(v_{lt})} \tag{1.15}$$

Due to the high search cost, apps compete strongly for listing on top charts, which can drastically increase the opportunity for exposure to potential users. Top chart ranking is therefore one of the most important factors determining app discovery. In many cases, app adoption falls into a virtuous cycle. High adoption in one period increases the app's discovery probability and boosts its adoption in the next period. The adoption process therefore assumes that app performance in the previous period affects app adoption in the current period.

Surveys of smartphone users (e.g. Nielsen 2011; Surikate 2012; Corti 2013) confirm that users rely heavily on top charts and social networks to discover new apps. Rankings on top charts have the most direct and the largest effect on app discovery probability. In addition, the more an app is adopted, the more likely it is to be mentioned in social media and the more likely it is to be discovered by other potential users.

Therefore, we assume the factor v_{it} determining app i's sampling probability is characterized as:

$$v_{it} = \gamma_1 Q_{i,t-1}^2 + \gamma_2 Q_{i,t-1} + \gamma_3 k_{it} + \gamma_4 t + \varsigma_{it}, \tag{1.16}$$

where $Q_{i,t-1}$ is app i's total number of adoptions up to period $t-1$, and k_{it} is app i's ranking in period t. It is likely that the adoption rate decrease as the user base gets larger, so the quadratic term of $Q_{i,t-1}$ wants to capture this pattern. We assume that an app's novelty follows a geometric decay process captured by γ_4, which is similar to findings in the advertising literature (e.g. Mela, Gupta, and Lehmann 1997; Dubé, Hitsch, and Manchanda 2005). ς_{it} captures unobservable factors that affect the sampling probability, such as in-app advertising, being featured in an app store, or being mentioned in a TV show or in a celebrity's blog.[8]

Conditional Adoption Problem

Apps are highly differentiated, and users may thus install multiple apps for the same purpose (Ghose and Han 2014). This app demand model does not assume that app adoption is exclusive. Therefore, consumers could discover more than one app in each period. Competition in our app adoption model therefore mainly comes from the app discovery process. After an app is exposed to users, it is endowed with great market power and its only competitor is an outside option. The conditional adoption process is therefore considered to be a discrete choice between the focal app and the outside option, without involving comparison with other apps.

Suppose that a user is exposed to app i at period t with probability p_{it}. The user installs app i if and only if the indirect utility from this app is greater than the utility from the market-specific outside option.

The conditional adoption probability of app i is therefore:

$$\bar{s}_i = P(\zeta_i + \epsilon_i > \epsilon_0) = \frac{exp(\zeta_i)}{1 + exp(\zeta_i)}. \tag{1.17}$$

Users who download this app become adopters. Those who have not yet adopted the app and new users coming to the platform in this period are the potential users in the next period. Users may be exposed to the same app multiple times. We assume that exposures

[8] v_{it} does not contain any time-invariant components, simply because the discovery probability constantly changes over time. The app market is very new and still evolving, and therefore even factors that are usually assumed to be time-invariant in other markets (e.g. the branding effect) are unlikely to be time-invariant in the app market.

and adoption decisions in previous periods do not affect a user's adoption decision in the current period. In other words, users who were exposed to a given app but did not adopt it in previous periods will make an adoption decision in the same way as new users. This assumption makes the adoption model much more tractable; moreover, it is not as strong as it sounds. First, most apps are regularly updated and app characteristics evolve quickly; many apps, for example, have their icons or interfaces re-designed every few months. Hence, the carry-over impression from previous exposure provides limited information about the app's current quality, and should not have much effect on the user's current adoption decision. Second, the number of potential users is much larger than the number of available apps. Statistically, the probability that an app is exposed to the same users multiple times is small and any errors introduced by this assumption are likely to be essentially negligible.

Actual Adoption Rate

Multiplying (1.15) and (1.17) gives the actual adoption rate s_{it}:

$$s_{it} = p_{it} \cdot \bar{s}_i = \frac{exp(v_{it})}{\sum_{l=1}^{I} exp(v_{lt})} \times \frac{exp(\zeta_i)}{1 + exp(\zeta_i)}. \tag{1.18}$$

This app adoption model reveals how apps diffuse over time. One of the dynamics embedded in the model suggests that downloads in previous periods determine the app discovery probability in the current period and further affect the app adoption in the current period.

Adoption Model Estimation

The interpretation of this adoption rate is not straightforward. Assume that the total number of potential users in market m in period t is N_{jt}, and that q_{it} users adopt app i in period t. The adoption rate in (1.18) is then:

$$s_{it} = \frac{q_{it}}{N_{jt} - Q_{i,t-1}}, \tag{1.19}$$

where $Q_{it} = \sum_{k=1}^{t} q_{ik}$ is the total number of downloads realized up to period t. Usually, $Q_{it} \ll N_{jt}$, so that:

$$s_{it} \approx \frac{q_{it}}{N_{jt}}. \tag{1.20}$$

Replace the adoption rate s_{it} with (1.20). The adoption function in (1.19) is transformed into:

$$\frac{q_{it}}{N_{jt}} = \frac{exp(v_{it})}{\sum_{l=1}^{I} exp(v_{lt})} \times \frac{exp(\zeta_i)}{1 + exp(\zeta_i)}, \tag{1.21}$$

and logarithm transformation results in:

$$ln(q_{it}) = v_{it} + \underbrace{\ln(N_{jt}) - \ln\left(\sum_{l=1}^{I} exp(v_{lt})\right)}_{\text{constant within period } t} + \ln\left(\frac{exp(\zeta_i)}{1+exp(\zeta_i)}\right) \tag{1.22}$$

Noting that N_{jt} and $\sum_{l=1}^{I} exp(v_{lt})$ in Equation (1.22) are constant within period t, define:

$$\mu_t \equiv \ln(N_{jt}) - \ln\left(\sum_{l=1}^{I} exp(v_{lt})\right). \tag{1.23}$$

And $\ln\left(\frac{exp(\zeta_i)}{1+exp(\zeta_i)}\right)$ is a monotonic transformation of quality ζ_i, so define:

$$\eta_i \equiv \ln\left(\frac{exp(\zeta_i)}{1+exp(\zeta_i)}\right). \tag{1.24}$$

Finally, v_{it} is characterized by Equation (1.16). Substituting Equations (1.16), (1.23) and (1.24) into Equation (1.22) gives:

$$ln(q_{it}) = \gamma_1 Q_{i,t-1}^2 + \gamma_2 Q_{i,t-1} + \gamma_3 k_{it} + \gamma_4 t + \eta_i + \mu_t + \varsigma_{it}. \tag{1.25}$$

However, our data contain very limited information on app downloads. No download information is available for Apple app store. For Google app store, only intervals of cumulative app downloads are observed. Although the download interval cannot be used directly to estimate the adoption model, descriptive evidence shows that download intervals are highly correlated with the number of ratings. We therefore assume that the number of downloads is proportional to the number of ratings and that $q_{it} \approx \alpha r_{it}$, which leads to:

$$ln(r_{it}) = -\ln(\alpha) + \alpha_1 R_{i,t-1}^2 + \alpha_2 R_{i,t-1} + \alpha_3 k_{it} + \alpha_4 t + \eta_i + \mu_t + \varsigma_{it}, \tag{1.26}$$

where $R_{i,t}$ is app i's total cumulative number of ratings at the beginning of period t.

After these transformations, the original app adoption model is simplified to a linear equation with period fixed effects and app fixed effects. Parameters of interest are the transformed app qualities ζ_i, $i = 1, ..., I$, captured by the app fixed effect in (1.26). The original 2-year daily panel data are cut into bi-weekly panel data in which estimation is performed. The app ranking during each window is represented by the number of days the app is ranked at different positions on the top chart (top 50, top 50-100, or outside top 100).

In order to estimate so many fixed effects consistently, we take the first difference of (1.26) to get rid of the fixed effect.

$$ln(r_{it}) - ln(r_{it-1}) = \alpha_1(R_{i,t-1}^2 - R_{i,t-2}^2) + \alpha_2(R_{i,t-1} - R_{i,t-2}) + \alpha_3(k_{it} - k_{it-1}) + \alpha_4 + (\mu_t - \mu_{t-1}) + \varsigma_{it} - \varsigma_{it-1}. \tag{1.27}$$

In the equation above $(R^2_{i,t-1} - R^2_{i,t-2})$, $(R_{i,t-1} - R_{i,t-2})$ and $(k_{it} - k_{it-1})$ are correlated with $\varsigma_{it} - \varsigma_{it-1}$. However, R_{i1}, ..., R_{it-2}, R^2_{i1}, ..., R^2_{it-2} are all exogenous with $\varsigma_{it} - \varsigma_{it-1}$, which can be used to construct many moment conditions. Equation (1.27) can be consistently estimated by GMM. Once coefficients α are estimated consistently, the fixed effect η_i in equation(1.26) could be easily recovered.

Apple and Google apps are estimated separately, so the estimated quality is platform-specific. To make sure the quality measures in Apple and Google are comparable, we use the cross-platform apps to calibrate estimated quality levels. [9] For the majority of apps that are present only on one platform, only one quality is recovered and this is used to proxy the app's quality on the other (not entered) platform. However, this assumption may cause a selection problem as the Android and iOS operating systems represent two different platforms in terms of application programming interfaces, development tools and programming architecture. Because developers might be more proficient on one platform than the other, the same app idea may eventually have different realized quality levels on the two platforms. This may incentivize developers to enter the platform with the higher realized quality level and unobservable quality might therefore be overestimated. If, however, an entrant's expected profit depends on its competitors' highest realized quality levels across both platforms, then the selection problem will not cause bias in the estimation. We consider this to be the case: in general, entrants are not aware of their competitors' development proficiency in iOS and Android, so their natural strategy is to assume competitive apps to be of equally high quality on both platforms.

Estimation Results

The summary statistics of the app adoption models are presented in Table 1.6. As revealed, the total number of ratings and the incremental number of ratings are highly skewed in both Apple and Google app stores. The majority of apps have very few ratings. Google game apps have relatively more ratings. This table also shows that it is extremely difficult to be ranked on top charts especially on Top100, but on average the chance is slightly higher for Google apps. Since Google app store is catching up, its apps are newer than Apple game apps.

Table 1.7 and Table 1.8 show the demand estimation results for Apple and Google apps, respectively.[10] Figure 1.4 depicts the kernel density of estimated app qualities.

When the total adopter base is very large, the overall impact of the total number of ratings on app adoption is positive, suggesting that a large user base benefits app discovery (e.g. through word of mouth (WOM)). While the adopter base is very large, the overall impact

[9] A cross-platform app is essentially the same app present on two platforms, so the quality estimates on the two platforms should equal or nearly equal. We regress cross-platform apps' quality estimates based on Google on those based on Apple. The slope of the simple linear regression is 0.87 and the intercept is 0.10. Based on this equation, Apple and Google's app qualities are transformed to a comparable scale.

[10] As top charts are specific to original genres, the demand is estimated on the original genres instead of the redefined market.

Table 1.6: Summary Statistics of Variables in the App Adoption Estimation

Apple

Variables	Obs	Mean	Std. Dev.	Min	Max
Incremental Number of Ratings	13,184,933	11	613	0	475,688
Total Number of Ratings	13,184,933	825	12,236	0	1,688,434
Days Since Launch	13,184,933	545	396	0	1,930
Days Ranked on Top 50	13,184,933	0.042	0.695	0	14
Days Ranked on Top 50-100	13,184,933	0.042	0.651	0	14
Days Ranked off Top 100	13,184,933	0.826	3.092	0	14

Google

Variables	Obs	Mean	Std. Dev.	Min	Max
Incremental Number of Ratings	3,576,405	46	1,444	0	496,140
Total Number of Ratings	3,576,405	1,269	18,921	0	2,780,605
Days Since Launch	3,576,405	298	249	0	1,829
Days Ranked on Top 50	3,576,405	0.116	0.540	0	14
Days Ranked on Top 50-100	3,576,405	0.116	1.219	0	14
Days Ranked off Top 100	3,576,405	0.799	3.055	0	14

The original data are on daily basis. To reduce the computation burden, we choose a wider biweekly span for for the adoption estimation. Essentially, it is a biweekly panel data set and this table reports the summary statistics of major variables on the pooled data.

Table 1.7: Apple App Adoption Estimation Results

Variables	Action	Adventure	Arcade	Board	Card	Casino
Log(TotalNumberofRatings)^2	-0.1681***	-0.1530***	-0.1722***	-0.1421***	-0.1074***	-0.0899***
	(0.0007)	(0.0010)	(0.0007)	(0.0011)	(0.0012)	(0.0013)
Log(Total Ratings)	0.9306***	0.8754***	0.9032***	0.8525***	0.6144***	0.5455***
	(0.0066)	(0.0087)	(0.0066)	(0.0119)	(0.0148)	(0.0152)
Log(Present Days)	0.0441***	0.0697***	0.0457***	0.0011	0.0297***	0.0545***
	(0.0016)	(0.0023)	(0.0015)	(0.0024)	(0.0037)	(0.0048)
Days Ranked on Top50	0.2094***	0.2638***	0.2379***	0.4060***	0.3110***	0.3425***
	(0.0047)	(0.0048)	(0.0045)	(0.0043)	(0.0042)	(0.0043)
Days Ranked on Top50-100	0.3345***	0.2664***	0.3449***	0.2749***	0.1967***	0.1921***
	(0.0048)	(0.0049)	(0.0048)	(0.0046)	(0.0047)	(0.0045)
Days Ranked off Top100	0.2128***	0.1779***	0.2057***	0.0634***	0.0595***	0.0407***
	(0.0014)	(0.0016)	(0.0013)	(0.0013)	(0.0014)	(0.0015)
Constant	0.1204***	(0.0016)	0.1135***	0.0816***	-0.0518***	-0.1536***
	(0.0054)	(0.0084)	(0.0053)	(0.0080)	(0.0122)	(0.0172)
Observations	1,823,918	882,181	1,802,665	744,548	373,920	238,435
Number of App	53,580	28,527	51,887	21,757	11,895	8,360

VARIABLES	Educational	Family	Music	Puzzle	Racing	Role Playing
Log(TotalNumberofRatings)^2	-0.0941***	-0.1384***	-0.1227***	-0.1524***	-0.1222***	-0.1512***
	(0.0012)	(0.0009)	(0.0020)	(0.0008)	(0.0016)	(0.0014)
Log(Total Ratings)	0.3414***	0.7519***	0.6124***	0.8153***	0.6749***	0.8390***
	(0.0114)	(0.0085)	(0.0170)	(0.0068)	(0.0140)	(0.0116)
Log(Present Days)	0.0447***	0.0625***	0.0649***	0.0359***	0.0762***	0.0796***
	(0.0017)	(0.0018)	(0.0045)	(0.0013)	(0.0043)	(0.0052)
Days Ranked on Top50	0.3847***	0.4482***	0.2555***	0.3026***	0.3663***	0.2881***
	(0.0035)	(0.0049)	(0.0039)	(0.0044)	(0.0047)	(0.0058)
Days Ranked on Top50-100	0.2691***	0.3346***	0.1317***	0.3719***	0.2366***	0.2806***
	(0.0036)	(0.0048)	(0.0042)	(0.0044)	(0.0053)	(0.0055)
Days Ranked on off Top100	0.0571***	0.1657***	-0.0058***	0.1899***	0.0435***	0.1004***
	(0.0011)	(0.0016)	(0.0016)	(0.0013)	(0.0018)	(0.0020)
Constant	-0.0447***	-0.0653***	0.0720***	-0.0153***	0.1518***	0.1975***
	(0.0060)	(0.0065)	(0.0179)	(0.0049)	(0.0176)	(0.0174)
Observations	781,328	1,206,584	174,096	1,989,339	250,813	328,078
Number of App	24,367	37,640	6,103	59,195	9,923	11,617

VARIABLES	Simulation	Sports	Strategy	Trivia	Word
Log(TotalNumberofRatings)^2	-0.1346***	-0.1468***	-0.1410***	-0.1200***	-0.1144***
	(0.0011)	(0.0013)	(0.0010)	(0.0014)	(0.0013)
Log(Total Ratings)	0.7870***	0.8429***	0.7689***	0.6840***	0.7742***
	(0.0095)	(0.0130)	(0.0094)	(0.0148)	(0.0169)
Log(Present Days)	0.1124***	0.0707***	0.0581***	0.0000	-0.0463***
	(0.0036)	(0.0033)	(0.0027)	(0.0029)	(0.0039)
Days Ranked on Top50	0.3315***	0.4002***	0.3866***	0.3213***	0.3398***
	(0.0054)	(0.0045)	(0.0050)	(0.0033)	(0.0033)
Days Ranked on Top50-100	0.2497***	0.3312***	0.3429***	0.2091***	0.2017***
	(0.0054)	(0.0046)	(0.0051)	(0.0036)	(0.0038)
Days Ranked on off Top100	0.1403***	0.0670***	0.1182***	0.0322***	0.0253***
	(0.0020)	(0.0017)	(0.0015)	(0.0010)	(0.0011)
Constant	0.0028	0.0123	0.0834***	0.0239**	0.2350***
	(0.0126)	(0.0117)	(0.0092)	(0.0104)	(0.0119)
Observations	529,819	453,086	756,085	400,708	323,848
Number of App	16,864	13,871	22,535	13,170	11,063

Standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Table 1.8: Google App Adoption Estimation Results

Variables	Arcade & Action	Brain & Puzzle	Cards & Casino	Casual	Racing	Sports Games
Log(TotalNumberofRatings)^2	-0.0705***	-0.0798***	-0.0500***	-0.0693***	-0.0558***	-0.0474***
	(0.0008)	(0.0008)	(0.0012)	(0.0008)	(0.0015)	(0.0013)
Log(Total Number of Ratings)	0.1820***	0.2125***	0.2173***	0.1918***	0.1811***	0.0667***
	(0.0099)	(0.0081)	(0.0166)	(0.0093)	(0.0211)	(0.0177)
Log(Days Since Launch)	-0.0139***	-0.0142***	-0.0775***	-0.0114***	-0.0373***	-0.0794***
	(0.0017)	(0.0012)	(0.0040)	(0.0017)	(0.0074)	(0.0050)
Days Ranked on Top 50	0.0894***	0.1168***	0.0801***	0.1562***	0.1750***	0.1324***
	(0.0041)	(0.0041)	(0.0059)	(0.0046)	(0.0045)	(0.0049)
Days Ranked on Top 50-100	0.0683***	0.1281***	0.1145***	0.1312***	0.1238***	0.1019***
	(0.0039)	(0.0040)	(0.0046)	(0.0042)	(0.0043)	(0.0047)
Days Ranked off Top 100	0.0478***	0.0727***	0.0556***	0.0756***	0.0616***	0.0589***
	(0.0022)	(0.0020)	(0.0020)	(0.0020)	(0.0022)	(0.0022)
Constant	1.2992***	0.6830***	0.7947***	0.8941***	1.0435***	1.0063***
	(0.0148)	(0.0099)	(0.0234)	(0.0123)	(0.0401)	(0.0269)
Observations	877,677	1,215,926	265,800	919,854	103,803	189,928
Number of Apps	47,741	64,675	12,509	49,715	6,586	9,332

Standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

becomes negative. This is because the very large adopter base erodes the potential users which leads to slower adoptions. The total number of ratings plays a more important role in app adoption in Apple app store. Ranking affects adoption very strongly, particularly in Apple app store. This difference between the Apple and Google ranking effects suggests that different platforms might be associated with different user behavior or ranking mechanisms. Generally speaking, the stimulating effect of being ranked on top charts decreases with the rank. On average, the effect of being ranking in the top 50 is 2-4 times larger than that of being ranked beyond the top 100.

The adoption model also suggests very different impacts of tenure days on app adoption between two platforms. In Google app store, the effect of longer tenure days is decaying over time, but in Apple app store, this effect is accumulating. The opposite impacts imply that popular apps in Apple store gets even more popular, while market attention in Google app store is volatile. If Apple app store keeps promoting already established apps, it is very likely to have this accumulating effect.

The kernel densities of the estimated game app quality are shown in Figure 1.4; both Apple and Google apps are skewed leftwards. This indicates that high-quality apps account for a small proportion of game apps and the majority of apps having low to moderate quality levels. Despite this similarity, the quality distributions of two platforms are very different on some aspects. For example, Apple's distribution is more concentrated, with a very high spike right above 0 and two thin tails, while that of Google is more spread out and is skewed to the left, resulting in the lower mean and median quality of Google apps. Although Google has

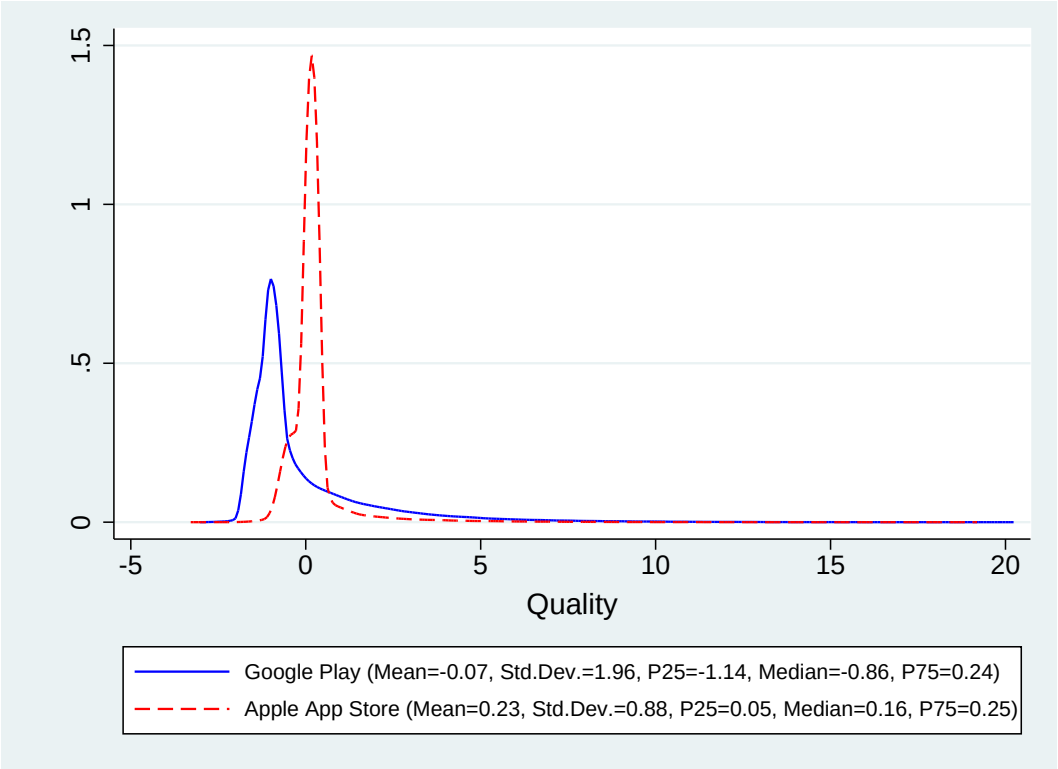


Figure 1.4: Apple and Google App Quality Distribution

a much larger share of low-quality apps, the two platforms have very close 75% percentiles. This implies than Google is not necessarily inferior to Apple in the high-quality segment. The estimated app quality will be used as an important variable the in app's platform choice model.

1.5 Platform Choice Decision

This section develops the empirical model of app's platform choices, in which potential entrants make platform choices based on their pre-determined quality level, observable platform characteristics, and expectations about post-entry platform characteristics. The app quality recovered in §1.4 will be used to estimate the platform choice model.

In each period, a finite number of potential entrants with pre-determined quality levels arrive, and simultaneously decide which platform (Apple or Google) to enter. Observing the platform characteristics and the incumbent composition, each entrant forms an expectation about other potential entrants' entry probabilities, and further updates its beliefs about the post-entry distribution of app quality for each platform. Based on these expectations and beliefs, the potential entrant makes a platform choice decision by comparing the expected profits associated with each platform. The platform with the higher profit will be entered.

Incumbents' qualities, the number of potential entrants, and the distribution of the quality levels of potential entrants are all public information. We adopt the Bayesian Nash equilibrium concept, in which potential entrants' actual choices should coincide with the expected choices.

In this model, entry is a one-time action. As is common in the entry literature, potential entrants that do not enter are not observed in the data. Because we focus on how apps choose which platform to enter, a model that accommodates no entry will not provide extra insights. Thus, no entry is not an option in this platform choice model.

Further, we adopt a static rather than a dynamic entry model, for four reasons. First, the main advantage of a dynamic entry model lies in the fact that it is able to achieve a better estimate of entry cost by modeling entry profit as the expected discounted future profits. However, entry cost is not the central question in this research. Second, a dynamic entry game inherently assumes that entrants can project the long-term state transition, which is unrealistic in a fast-evolving area such as the app market. Entry decisions are therefore more likely to be myopic. Third, the dynamic entry model assumes that developers can wait to enter, but it is not true in reality. Fourth, estimating a dynamic entry game for a large market is also computationally infeasible, unless we assume a new equilibrium concept (e.g. the oblivious equilibrium; see Weintraub, Benkard, and Van Roy 2008 and Benkard, Jeziorski, and Weintraub 2013). Overall, a static entry model is appropriate for addressing the research questions in this chapter.

Basic Setting

Let the finite number of potential entrants be I, and denote Google app store and Apple app store as platforms 0 and 1 respectively. Potential entrant i chooses an action $a_i \in \{0, 1\}$. All potential entrants make their choices simultaneously, and $A = \{0, 1\}^n$ denotes the set of all possible actions of potential entrants. Also, let $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_I)$ denote the action vector of all potential entrants except entrant i.

$s_i \in S_i$ is the state variable for potential entrant i, and similarly $s = (s_1, \ldots, s_I) \in S$ is the state vector of all potential entrants, where $S = \prod_{i=1}^{I} S_i$. s_i includes a set of variables influencing entrant i's profit, such as market size, entrant i's own quality level and the quality distribution of its competitors. We assume that s is common knowledge in the sense that it is not only known to all potential entrants but is also observable to researchers.

Besides the state variable s_i, each potential entrant has a choice-specific state vector, $\epsilon_i = (\epsilon_i(0), \epsilon_i(1))$, which can be considered to be the idiosyncratic shocks to entrant utility. The random shock may include some unobservable developer know-how knowledge or personal preference towards one platform, so this state is only known to entrant i itself and the private state vector is then $\epsilon_i = (\epsilon_i(0), \epsilon_i(1))$. We assume that $\epsilon_i(a_i)$ follows i.i.d. extreme value distribution.

Potential entrant i obtains the following utility from taking action a_i:

$$u_i(a, \epsilon_i, s; \theta) = \pi_i(a_i, a_{-i}, s; \theta) + \epsilon_i(a_i), \tag{1.28}$$

where $\pi_i(a_i, a_{-i}, s; \theta)$ is the profit associated with action a_i and $\epsilon_i(a_i)$ is the idiosyncratic shock of taking action a_i. θ is a vector of the underlying model parameters and we will discuss the specification of $\pi_i(a_i, a_{-i}, s; \theta)$ in detail in §1.5.

Entrant i's policy function is a mapping of the state variable to the optimal action, denoted as $a_i = \nu(s, \epsilon_i)$. Then the probability of entry, $\sigma_i(a_i = k \mid s)$, $k = 0, 1$, can be expressed as

$$\sigma_i(a_i = k|s) = \int \mathbf{1}(\nu(s, \epsilon_i) = k) f(\epsilon_i) d\epsilon_i. \tag{1.29}$$

Entrants take actions simultaneously, so they do not know competitors' actual actions. In period 0, entrant i's expected utility from action a_i is as follows:

$$\begin{aligned} U_i(a_i, \epsilon_i, s; \theta) &= E_{a_{-i}}\left(u_i(a, \epsilon_i, s; \theta)\right) \\ &= \sum_{a_{-i}} \pi_i(a_i, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s) + \epsilon_i(a_i) \\ &\equiv \Pi_i(a_i, s; \theta) + \epsilon_i(a_i), \end{aligned} \tag{1.30}$$

where $\sigma_{-i}(a_{-i}|s) = \prod_{j \neq i} \sigma_j(a_j|s)$, and $\Pi_i(a_i, s; \theta)$ is the expected profit.

Entrant i takes action $a_i = k \in \{0, 1\}$ if action k provides a higher expected utility than action $1 - k$. Entrant i's policy function therefore should satisfy:

$$\begin{aligned} \sigma_i(a_i = k|s) &= Prob\left(U_i(a_i = k, \epsilon_i, s; \theta) > U_i(a_i = 1 - k, \epsilon_i, s; \theta)\right) \\ &= Prob\left(\Pi_i(a_i = k, s; \theta) + \epsilon_i(k) > \Pi_i(a_i = 1 - k, s; \theta) + \epsilon_i(1 - k)\right) \\ &= \Lambda\left(\Pi_i(a_i = k, s; \theta) - \Pi_i(a_i = 1 - k, s; \theta)\right), \end{aligned} \tag{1.31}$$

where $\Lambda(\cdot)$ is the logistic function.

Profit Function

In general, an app's expected profit is affected by both its own quality level and market characteristics. Market characteristics include the size of the user base, the distribution of app quality in the focal market and other unobservable market characteristics.

Let ζ_i denote app quality. As the number of apps is finite, we have a finite set $\Theta = \{\zeta_i, i = 1, 2, \ldots, \bar{I}\}$ containing all possible levels of app quality, where $\zeta_1 \leq \zeta_2 \leq \ldots \leq \zeta_{\bar{I}}$ with $\bar{I} < \infty$ being the largest possible number of apps available on a platform. It is computationally prohibitive to conduct analyses directly on this set. Similarly to BOY and to Kim 2013, we observe from the data that the app market is highly skewed toward a few killer apps with a fat left tail, which is also consistent with the rating distribution. Because of this polarized market structure, app quality is discretized into two types: low-type and high-type.

Define the set $\omega_l = \{\zeta_i : \zeta_i < d \text{ for } \forall\ \zeta_i \in \Theta\}$ as the set of low-type quality levels, and the set $\omega_h = \{\zeta_i : \zeta_i \geq d \text{ for } \forall\ \zeta_i \in \Theta\}$ as the set of high-type quality levels, where d denotes the cutoff value of low/high type. Also denote $\Omega = (\omega_l, \omega_h)$. Apps in the same subset are homogeneous in terms of strategic interactions, meaning that they receive and exert the

same competitive effects. Depending on the actual quality level, apps in the same subset may still receive different profits.

Applying the counting measure on Ω results in the vector $H = (h_l, h_h)$, where $h_j = |\omega_j|$, $j = l, h$. The vector H characterizes the distribution of apps in low- and high-quality bins. At period 0, the market state is (Ω, H_0). After entry, the market state transits to (Ω, H_1) in period 1. The relationship between H_0 and H_1 is defined as:

$$H_1 = H_0 + \Delta, \tag{1.32}$$

where $\Delta \equiv (\Delta_l, \Delta_h)$ is the quality distribution of entrants who choose to enter the focal platform, and depends on all potential entrants' platform choice decisions. It is easy to find:

$$E(\Delta) = \left(\sum_{j \in \omega_l} \sigma_j(a_j = k|s), \sum_{j \in \omega_h} \sigma_j(a_j = k|s) \right). \tag{1.33}$$

Following Berry 1992, Seim 2006 and Ciliberto and Tamer 2009, we assume a reduced-form profit function for entrant i ($\zeta_i \in \omega_j$) on platform k ($k = 0, 1$) as:

$$\begin{aligned} \pi_i(a_i = k, a_{-i}, s; \theta) &= \pi_i(a_{-i}, s^k; \theta^k) \\ &= \underbrace{\theta_1^k N^k + g_j\left(H_1^k, \theta_2^k\right) + \theta_3^k \zeta_i}_{revenue} - \underbrace{C_i^k}_{entry\ cost} + \rho_k, \end{aligned} \tag{1.34}$$

where

$$C_i^k = c_0^k + c_1^k b_i + c_2^k b_i \cdot \zeta_i \tag{1.35}$$

.

$\theta^k \equiv \left\{\theta_1^k, \theta_2^k, \theta_3^k\right\}$ is the vector of the parameters of interest. For simplicity of notation, the platform superscript k is suppressed hereafter. N is the platform user base and ρ_k represents market unobservable effect.

Function $g_j(H_1, \theta_2)$ reflects the effect of the post-entry quality distribution on app i's profit, which only depends on i's quality type, j. It is parameterized as $g_j(H_1, \theta_2) = \theta_{jl} h_{1l} + \theta_{jh} h_{1h}$, where h_{1l} is the number of low-quality apps post-entry and h_{1h} is the number of high-quality apps post-entry. The coefficients θ_{jl} and θ_{jh} capture the effect of low-type entrants and high-type entrants, respectively, on a type-j entrant. This assumption implies that the effects of apps with heterogeneous quality types are separately additive.

The platform-specific entry cost has two components: a fixed entry cost c_0, a variable entry cost $c_1 b_i + c_2 b_i \cdot \zeta_i$, where b_i is the size of app i. Larger apps have more content, and high-quality apps require more effort, both of which are associated with higher variable costs. In the end, (1.34) becomes:

$$\pi_i(a_i, a_{-i}, s; \theta) = \theta_1 N + h_{1l}\theta_{jl} + h_{1h}\theta_{jh} + \theta_3 \zeta_i - C_i + \rho_k. \tag{1.36}$$

As $H_1 \equiv \{h_{1l}, h_{1h}\}$ is realized only after entry, it is a stochastic term in the profit function, and the expected profit function is:

$$\begin{aligned} \Pi_i(a_i, s; \theta) &\equiv E(\pi_i(a_i, a_{-i}, s; \theta)) \\ &= \theta_1 N + E(h_{1l}|s)\theta_{jl} + E(h_{1h}|s)\theta_{jh} + \theta_3 \zeta_i - C_i + \rho_k, \end{aligned} \tag{1.37}$$

where $E(H_1|s) = H_0 + E(\Delta|s)$.

Equilibrium

This platform choice decision is an incomplete information static game. The existence of an equilibrium can be demonstrated by the Brouwer fixed-point theorem. Plugging Equation (1.33) into Equation (1.36) results in:

$$\Pi_i(a_i, s; \theta) = \theta_1 N + \left(h_{0l} + \sum_{j \in \omega_l} \sigma_j \right) \theta_{jl} + \left(h_{0l} + \sum_{j \in \omega_h} \sigma_j \right) \theta_{jh} + \theta_3 \zeta_i - C_i + \rho_k. \tag{1.38}$$

Then, if we replace the expected profit in (1.31) by (1.38), it is easy to see that an entrant's strategy is a mapping of other entrants' strategies, $\sigma = \Lambda(\sigma; s, \theta)$. As the logistic function $\Lambda(\cdot)$ is a continuous mapping, the equilibrium of this entry game must exist for any finite state variable.

The uniqueness of this equilibrium does not hold in general settings. Given the same state variable s, a different quality distribution vector H_1 may lead to the same $g_j(H_1, \theta_2)$. Multiple equilibria are very likely in such a setting.[11] However, the identification of this game does not rely on uniqueness. The estimation strategy proposed later only requires the data generating process to be from a single equilibrium.

1.6 Identification and Estimation

This section discusses, in turn, the identification and estimation strategies for the app's platform choice model.

Identification

Structural primitives θ in the profit function need to be identified. The profit function $\pi_i(a_i, a_{-i}, s; \theta)$ is a linear function. Once this is identified, the identification of θ is straightforward. The discussion thus focuses on identification of the profit function.

The first step is to identify the difference between the expected profits on each platform from the equation (1.31) as below,

$$\sigma_i(a_i = k|s) = \Lambda\left(\Pi_i(a_i = k, s; \theta) - \Pi_i(a_i = 1 - k, s; \theta)\right).$$

ASSUMPTION 1. *The idiosyncratic shock $\epsilon_i(a)$ follows some known continuous distribution and is i.i.d. distributed across action a and entrant i.*

[11] Seim 2006 discusses equilibrium uniqueness in a similar setting.

Assumption 1 ensures the existence of inverse Λ, so it is easy to get that

$$\Pi_i(a_i = k, s; \theta) - \Pi_i(a_i = 1 - k, s; \theta) = \Lambda^{-1}\left(\sigma_i(a_i = k|s)\right).$$

The next step is to identify the profit function $\pi_i(a_i, a_{-1}, s; \theta)$ from $\Pi_i(a_i = k, s; \theta) - \Pi_i(a_i = 1 - k, s; \theta)$. It is similar to solving a system of simultaneous equations, in which the number of equations must be larger than or equal to the number of unknowns.

ASSUMPTION 2. *A set of variables, z, satisfies the exclusion restrictions, and z contains at least one continuous variable.*

Exclusion restrictions require the existence of a set of state variables that only affect the entrant's profit and not that of its competitors', which is analogous to bringing more equations into a system of simultaneous equations to solve for unknowns. A developer's platform experience is essential to the success of its apps because of accumulation of knowledge and learning by doing. Hence, a developer's platform experiences are likely to affect its platform choice decision for the next app. However, it is unlikely to affect other developers' platform choice decisions. Therefore, developers' platform experiences (measured by the number of days operating on each platform) are used as exclusion restrictions in our estimation.

THEOREM 2. *If Assumption 1 and Assumption 2 hold, the profit function $\pi(a_i, a_{-i}, s; \theta)$ can be identified.*

Proof. If $\nu(s, \epsilon_i) = k_1$ is the equilibrium, there must be:

$$\Pi_i(a_i = k_1, s; \theta) + \epsilon_i(k_1) > \Pi_i(a_i = k_2, s; \theta) + \epsilon_i(k_2) \tag{1.39}$$

Following the previous assumption of extreme value distribution, the implication of this equilibrium condition is:

$$\sigma_i(a_i = k_1|s) = \Lambda\left(\Pi_i(a_i = k_1, s; \theta) - \Pi_i(a_i = k_2, s; \theta)\right)$$

The logit function $\Lambda(\cdot)$ gives a one-to-one mapping, so the difference in expected profit can be recovered from the optimal strategy:

$$\Pi_i(a_i = k_1, s; \theta) - \Pi_i(a_i = k_2, s; \theta) = \Lambda^{-1}\left(\sigma_i(a_i = k_1|s)\right) \tag{1.40}$$

This is not only the case for extreme value distribution. $\Pi_i(a_i = k_1, s; \theta) - \Pi_i(a_i = k_2, s; \theta)$ can be recovered from equilibrium by $\epsilon_i(a_i)$ with other continuous distribution assumptions, such as the normal distribution. Therefore, Assumption 1 guarantees the recovery of the difference in the expected profit function.

If we disregard strategic interactions, this entry game becomes a single-agent discrete choice problem that can easily be identified, similarly to BLP. Suppose that entrant i chooses to enter platform k_1 where $a_i = k_1$, and we obtain:

$$\begin{aligned}\Lambda^{-1}\left(\sigma_i(a_i = k_1|s)\right) &= \Pi_i(a_i = k_1, s; \theta) - \Pi_i(a_i = k_2, s; \theta) \\ &= \sum_{a_{-i}} \pi_i(k_1, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s) - \sum_{a_{-i}} \pi_i(k_2, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s)\end{aligned} \tag{1.41}$$

The right-hand side of (1.41) can be further expressed as:

$$\sum_{a_{-i}} \pi_i(k_1, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s) = \sum_{a_{-i}} \left(\pi_i(k_1, a_{-i}, s; \theta) \prod_{j \neq i}^{n} \sigma_j(a_j = k_1|s) \right) \tag{1.42}$$

and

$$\sum_{a_{-i}} \pi_i(k_2, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s) = \sum_{a_{-i}} \left(\pi_i(k_2, a_{-i}, s; \theta) \prod_{j \neq i}^{n} (1 - \sigma_j(a_j = k_1|s)) \right) \tag{1.43}$$

The identification of $\pi(a_i, a_{-i}, s; \theta)$ could be viewed as solving $\pi(a_i, a_{-i}, s; \theta)$ uniquely from the system of simultaneous equations structured by (1.41), (1.42) and (1.43) given the state variable s. Equation (1.42) has $n \times 2^{n-1}$ unknowns, because there are n entrants whose utility of entering k_1 depends on 2^{n-1} possible actions of $n-1$ other entrants. Accordingly, there are $n \times 2^{n-1}$ unknowns in (1.43). The left-hand side of (1.41) contains information about $n \times 2$ scalars, as there are n entrants and each has two actions. The right-hand side of (1.41) has $n \times 2^n$ unknowns. It is not possible to solve $\pi(a_i, a_{-i}, s; \theta)$ uniquely from this system of equations.

To identify $\pi(a_i, a_{-i}, s; \theta)$, some exclusion restrictions are introduced. The basic idea of exclusion restriction is to find some variables that only affect $\sigma_{-i}(a_{-i}|s)$ and not $\pi_i(a_i, a_{-i}, s; \theta)$, so that there are sufficient variations in $\Lambda^{-1}(\sigma_i(a_i|s))$, the left-hand side, to uniquely determine $\pi(a_i, a_{-i}, s; \theta)$. Suppose that a set of variables $\tilde{s}$ satisfies the exclusion restrictions. In other words, z influences entrant i's profit only through its effect on the platform choices of other entrants. Commonly used exclusion variables are productivity shocks. Entrant i's expected profit should depend on the platform choices of other competitors, and not on the productivity shocks of its competitors. With exclusion restrictions, (1.41) becomes:

$$\Lambda^{-1}(\sigma_i(a_i = 1|s, z)) = \sum_{a_{-i}} \pi_i(1, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s, z) - \sum_{a_{-i}} \pi_i(2, a_{-i}, s; \theta)\sigma_{-i}(a_{-i}|s, z) \tag{1.44}$$

In the system of simultaneous equations, there are $n \times 2$ left-hand scalars and $n \times 2^n$ unknowns, so the necessary condition for identification is that the support of the distribution of $\tilde{s}$ conditional on the state variable s has at least 2^{n-1} points, which is Assumption 2.

This necessary condition can be easily satisfied if z includes at least one continuous variable and $\Lambda^{-1}(\sigma_i(a_i|s, z))$ is the sufficient variable of z so that the support of z is still preserved.

In summary, if Assumption 1 and Assumption 2 hold, profit function $\pi(a_i, a_{-i}, s; \theta)$ can be identified and the identification is proven.

□

Estimation Strategy

ASSUMPTION 3. *Data from different markets are generated by a single equilibrium.*

The two-step approach for game estimation does not require uniqueness of the equilibrium nor an equilibrium selection mechanism. However, data need to be generated from the same equilibrium over different periods. In general, multiple equilibria are very common in entry games. This single-equilibrium assumption implies that a stable equilibrium is repeatedly played.[12]

Entrant i's decision rule is governed by:

$$\sigma_i(a_i = k|s) = Prob\left(\Pi_i(a_i = k, s; \theta) + \epsilon_i(k) > \Pi_i(a_i = 1 - k, s; \theta) + \epsilon_i(1 - k)\right). \quad (1.45)$$

Given the distribution of ϵ_i, it is easy to derive the likelihood function:

$$\mathcal{L}(\theta, \sigma) = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{N_{mt}} \left(a_{imt} ln\left(\Lambda(a_{imt}|s_{imt}; \theta, \sigma)\right) + (1 - a_{imt}) ln\left(1 - \Lambda(a_{imt}|s_{imt}; \theta, \sigma)\right)\right), \quad (1.46)$$

with the constraint that the action should be the equilibrium $\sigma = \Lambda(\theta, \sigma; S)$. N_{mt} is the number of potential entrants entering genre m in period t.

Consequently, the MLE is defined as

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \mathcal{L}(\theta, \sigma) \qquad \text{subject to} \quad \sigma = \Lambda(\theta, \sigma; S). \quad (1.47)$$

A common approach to estimating this problem is to nest a fixed-point algorithm in the MLE (e.g. Seim 2006). For a given set of θ, a vector of equilibrium choice probabilities, $\sigma^\star$, is easily calculated by finding the fixed point of $\sigma = \Lambda(\theta, \sigma; S)$. $\sigma^\star$ is then used to construct the likelihood function $\mathcal{L}(\theta, \sigma^\star)$. Therefore, for every set of θ evaluated in MLE, a corresponding vector of equilibrium choice probability σ needs to be calculated via the fixed-point algorithm. When the number of players is large, the domain of the policy function becomes very large. As a result, this estimation procedure can be extremely computationally intensive.

A feasible pseudo maximum likelihood is therefore applied to estimate this entry game, and is implemented in two steps. In the first step, a nonparametric estimate of the entry probability $\hat{\sigma}_i(a_i|s)$ is estimated, using a sieve logit (see Ai and Chen 2003). A sieve logit with cubic splines is used for the first step. The splines are interpolated between 1 to 3 equally space percentiles of each state variable. In the end, all of the power terms and splines are interacted with high quality dummy. The sieve logit also includes market fixed effect.

Given the predicted entry probability $\hat{\sigma}_i$, we have:

$$\hat{\Pi}_i(a_i, s; \theta) = \sum_{a_{-i}} \pi_i(a_i, a_{-i}, s; \theta) \prod_{j \neq i} \hat{\sigma}_j(a_j|s, \tilde{s}). \quad (1.48)$$

[12] A general approach allowing for multiple equilibria in the data-generating process is developed by Aguirregabiria and Mira 2007. Sweeting 2009 shows that with particular data structures, multiple equilibria may aid identification of model primitives.

Following the distribution of ϵ_i, a pseudo likelihood function could be constructed using $\hat{\Pi}_i$:

$$\mathcal{L}(\theta; a_i, s, \hat{\sigma}) = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{N_{mt}} \sum_{k=0}^{1} \mathbf{1}(a_{imt} = k) ln \left(\Lambda \left(\hat{\Pi}_i(a_{imt} = k, s; \theta) - \hat{\Pi}_i(a_{imt} = 1-k, s; \theta) \right) \right). \tag{1.49}$$

Primitives θ can be estimated by maximizing the pseudo likelihood function, the global maximum of which is guaranteed.

To estimate the model, we split the data into 24 monthly intervals. State variables are taken to be the values at the beginning of each period. Based on recovered quality levels, apps are categorized into low-type and high-type. Different cutoff points for low- and high-type apps are used to check robustness. A 2-year rolling sum of global smartphone shipments is used as a proxy for the size of the user base.[13]

The error in this two-step estimation mainly arises from the first-step estimation of choice probabilities. As long as $\hat{\sigma}$ is estimated consistently, with some weak assumptions, the consistency of the two-step estimation is guaranteed (see Aguirregabiria and Mira 2007; Bajari et al. 2010).[14] As a matter of fact, the big data allow very flexible nonparametric estimation for the first stage, which in turn guarantee the accurate prediction of $\hat{\sigma}$. Due to this reason, two-step approach is a very suitable method for the big data.

Estimation Results

The summary statistics of the main variables in the platform choice model are presented in Table 1.9. During the data period, there are slightly more apps entering Google. The average developer tenure on Google is 88 days, much shorter than the 272 days for Apple developers. This implies that many Google entries occurred recently. Slightly more apps chose Apple during the data period. Table 1.10 breaks down entrants into high-type and low-type. More high-type apps chose Google, but slightly more low-quality apps entered Apple.

Table 1.11 reports the estimation results for structural parameters. To investigate the interactive effects of heterogeneous apps more closely, the estimated competitive effects coefficients are separated into Table 1.12 and impacts of other state variables are gathered in Table 1.13.

[13] In the past two years, more than 80% of global smartphone shipments were running on the Android operating system.

[14] However, this two-step estimation is less efficient than the nested fixed-point estimation. Suppose that $\hat{\sigma}$ is the first step estimator and $\hat{\theta}_{2s}$ is the second step estimator. Also define $l_{mt} \equiv \sum_{i=1}^{N_m} a_{imt} ln\left(\Lambda(a_{imt}|s_{imt}; \theta, \sigma)\right)$. Then the Fisher information matrix of the pseudo maximum estimation is $\Omega = E(\nabla_\theta l_{mt} \nabla_\theta l'_{mt})$. Following the properties of MLE, it is easy to show that the infeasible MLE $\hat{\theta}$ has $\sqrt{MT}(\hat{\theta} - \theta) \to_d N(0, \Omega^{-1})$. For the two-step estimation, suppose that the first-step estimator satisfies $\sqrt{MT}(\hat{\sigma} - \sigma) \to_d N(0, \Sigma)$, then $\sqrt{MT}(\hat{\theta}_{2s} - \theta) \to_d N(0, V_{2s})$ where $V_{2s} = \Omega^{-1} + \Omega^{-1}\Omega_{\theta\sigma}\Sigma\Omega'_{\theta\sigma}\Omega^{-1}$ and $\Omega_{\theta\sigma} = E(\nabla_\theta l_{mt} \nabla_\sigma l'_{mt})$. As $\Omega^{-1}\Omega_{\theta\sigma}\Sigma\Omega'_{\theta\sigma}\Omega^{-1}$ is a positive definite matrix, V_{2s} is greater than Ω^{-1}, meaning that the two-step estimation is less efficient than the infeasible MLE.

Table 1.9: Summary Statistics of Variables in Entry Estimation

Variable	Obs	Mean	Std. Dev.	Min	Max
Entry Platform (1=Apple)	245,081	0.48	0.50	0	1
App Quality	245,081	0.14	0.92	-2.61	8.23
App Download Size (Megabyte)	245,081	19.15	48.79	0	2,560
Developer Tenure on Apple	245,081	88.65	187.89	0	1,735
Developer Tenure on Google	245,081	271.67	426.58	0	1,928
iPhones 2-Yr Global Shipments (Billion)	245,081	0.22	0.04	0.12	0.27
Android Phones 2-Yr Global Shipments (Billion)	245,081	0.71	0.28	0.24	1.16

This table reports the summary statistics of game apps entering either platform during the data period. Smart phone shipments are the sum of smart phone shipments in the past 2 years since the focal app entered either platform. They are used to capture the size of smart phone user base.

Table 1.10: Entrants Type Distribution by Platforms

Type		Android Entrant %	iTunes Entrant %	No. of Entrants
Quality> 1	H	8.4	1.7	245,081
	L	43.1	46.8	
Quality> 1.5	H	6.8	1.1	
	L	44.7	47.4	
Quality> 2	H	5.4	0.8	
	L	46.1	47.7	

For robustness, the high type and low type apps are categorized according to three cutoff points. As the cutoff point gets larger, the proportion of high type entrants becomes smaller.

Table 1.11: Estimated Structural Parameters

Variables	Quality>1	Quality>1.5	Quality>2
θ_{ll}^1	0.0538	0.078	0.1060**
	(0.0574)	(0.0564)	(0.0537)
θ_{lh}^1	0.2369	-0.314	-1.2153
	(1.2455)	(1.5555)	(1.7505)
θ_{hl}^1	-0.2260**	-0.3837***	-0.4667***
	(0.1042)	(0.1353)	(0.1550)
θ_{hh}^1	-4.4954***	-4.7051***	-5.1338***
	(0.9617)	(1.3053)	(1.7657)
θ_{ll}^2	0.1883***	0.1319***	0.0940**
	(0.0526)	(0.0486)	(0.0446)
θ_{lh}^2	-0.9524**	-0.7407	-0.3212
	(0.4400)	(0.5867)	(0.7434)
θ_{hl}^2	-0.4365***	-0.5124***	-0.4975***
	(0.1043)	(0.1374)	(0.1661)
θ_{hh}^2	1.1966*	1.8999*	1.8894
	(0.6150)	(1.0856)	(1.9687)
iPhones 2-Yr Global Shipments (B)	12.1301**	9.9490**	7.4472
	(4.9217)	(4.9616)	(4.8862)
Android Phone 2-Yr Global Shipments (B)	-4.1813***	-3.8127***	-3.4397***
	(0.7675)	(0.7943)	(0.7802)
App Quality	0.9115***	0.8912***	0.8172***
	(0.0483)	(0.0474)	(0.0410)
Log(App Download Size) (Mb)	0.5980***	0.5973***	0.6007***
	(0.0124)	(0.0123)	(0.0124)
App Quality×Log(App Download Size) (Mb)	-0.0809***	-0.0665***	-0.0549***
	(0.0079)	(0.0086)	(0.0088)
Ln(Developer Tenures on Apple)	0.5431***	0.5458***	0.5483***
	(0.0112)	(0.0113)	(0.0110)
Ln(Developer Tenures on Google)	-0.6230***	-0.6265***	-0.6295***
	(0.0118)	(0.0117)	(0.0113)
Constant Entry Cost	-1.4269*	-1.0418	-0.5911
	(0.8350)	(0.8600)	(0.8300)
Market Fixed Effect	Yes	Yes	Yes
Observations	245,081	245,081	245,081

Bootstrapped standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Table 1.12: Estimated Competitive Effects

Platform	Type	Quality>1	Quality>1.5	Quality>2
Apple	H_{ll}	0.0538	0.078	0.1060**
		(0.0574)	(0.0564)	(0.0537)
	H_{lh}	0.2369	-0.314	-1.2153
		(1.2455)	(1.5555)	(1.7505)
	H_{hl}	-0.2260**	-0.3837***	-0.4667***
		(0.1042)	(0.1353)	(0.1550)
	H_{hh}	-4.4954***	-4.7051***	-5.1338***
		(0.9617)	(1.3053)	(1.7657)
Google	H_{ll}	0.1883***	0.1319***	0.0940**
		(0.0526)	(0.0486)	(0.0446)
	H_{lh}	-0.9524**	-0.7407	-0.3212
		(0.4400)	(0.5867)	(0.7434)
	H_{hl}	-0.4365***	-0.5124***	-0.4975***
		(0.1043)	(0.1374)	(0.1661)
	H_{hh}	1.1966*	1.8999*	1.8894
		(0.6150)	(1.0856)	(1.9687)

Bootstrapped standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Interestingly, for both platforms low-type incumbents make the platform inferior so that it is less attractive for high-type entrants. This negative externality is stronger in Google. Google and Apple also present very different competition structures. In Google, the presence of low-type incumbents encourages low-type entrants to enter, but in Apple the presence of high-type incumbents exerts a very negative effect on entrants of both types. The lower competition level in Google to some extent explains why many high-type apps chose to enter Google when Google fell far behind Apple on the high-end segment.

The user base has a positive effect on an entrant's expected profit, but the scale of the effect is larger for Apple than for Google. The estimates suggest that on average the profitability of an additional iPhone user is about 3-4 times that of an additional Android phone user. This finding is consistent with the industry consensus that Google app store is less profitable than Apple app store, but the larger Android user base offsets the lower

Table 1.13: Impacts of Other State Variables

Variables		Quality>1	Quality>1.5	Quality>2
User Base	iPhones 2-Yr Global Shipments (B)	12.1301**	9.9490**	7.4472
		(4.9217)	(4.9616)	(4.8862)
	Android Phone 2-Yr Global Shipments (B)	-4.1813***	-3.8127***	-3.4397***
		(0.7675)	(0.7943)	(0.7802)
	Constant Entry Cost	1.4269*	1.0418	0.5911
		(0.8350)	(0.8600)	(0.8300)
Entry Cost (Apple - Google)	Log(App Download Size) (Mb)	-0.5980***	-0.5973***	-0.6007***
		(0.0124)	(0.0123)	(0.0124)
	App Quality×Log(App Download Size) (Mb)	0.0809***	0.0665***	0.0549***
		(0.0079)	(0.0086)	(0.0088)
Platform Experience	Ln(Developer Tenures on Apple)	0.5431***	0.5458***	0.5483***
		(0.0112)	(0.0113)	(0.0110)
	Ln(Developer Tenures on Google)	0.6230***	0.6265***	0.6295***
		(0.0118)	(0.0117)	(0.0113)
Quality	App Quality	0.9115***	0.8912***	0.8172***
		(0.0483)	(0.0474)	(0.0410)
Market Fixed Effect		Yes	Yes	Yes
Observations		245,081	245,081	245,081

Bootstrapped standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

user engagement. For instance, based on data from two big app analytic firms, Cole 2013 finds that for every $1.00 in app download revenue earned by iOS developers, their Android counterparts earn just $0.19, and Apple's apps bring developers 5 times the per-download revenue of Google apps.

Google has a lower fixed entry cost than Apple, but a much higher variable entry cost for a typical app with the average quality level. A developer need only pay a registration fee of $25 to publish apps on Google app store, while Apple charges $100. Furthermore, new apps published on Apple's App Store have to undergo a strict review process that usually takes days or even weeks. Apps that violate the rules of the app store are rejected. In contrast, Google uses an automatic screening process. New apps submitted to Google are available on Google app store within 20 minutes unless they contain malware. These features of the Apple and Google app ecosystems result in a lower fixed entry cost for Google. On average Apple's App Store has a lower variable cost. Currently, there are only four mainstream iPhone models (iPhone4, 4S, 5 and 5S) with minimal variations in screen size and resolution. In contrast, there are many more Android phone models with various display specifications. causing the development cost to increase to accommodate the different Android phones. Consequently, it is generally more costly for an app with a mean or median download size to enter Google.

The structural estimation also shows that developer's platform experience has a strong impact on app's platform choices, but developer's platform experience on Google is more valuable. This is again because the ecosystem in Google is more complicated, which causes the stickier platform choices.

1.7 Policy Experiments

A booming platform needs to be attractive to potential entrants, especially to high-type apps. App stores could adopt certain market designs to improve the app market evolution. Based on the structural estimation in §1.6, counterfactual experiments in this section explore which market policies can benefit the platform expansion as well as improve apps composition.

The policy experiments are achieved by solving the new equilibrium. The structure parameters have been estimated. When a policy alters certain variables or parameters, the new equilibrium represents the updated market outcome and the new equilibrium could be solved by the fixed point algorithm. App's platform choice decision characterized by an incomplete information game is defined by the equation

$$\sigma_i(a_i = k|s) = \Lambda\left(\Pi_i(a_i = k, s; \theta) - \Pi_i(a_i = 1 - k, s; \theta)\right), \tag{1.50}$$

where

$$\Pi_i(a_i, s; \theta) = \theta_1 N + H_{1l}\theta_l + H_{1h}\theta_h + \theta_3\zeta_i - C_i + \rho_k. \tag{1.51}$$

H_{1l} and H_{1h} are the number of post-entry low-type and high-type apps respectively. $H_{1l} = h_{0l} + \sum_{j\in\omega_l}\sigma_j$ and $H_{1h} = h_{0h} + \sum_{j\in\omega_h}\sigma_j$. The equilibrium is a set of policy function $\{\sigma_1^\star, \ldots, \sigma_n^\star\}$ satisfying the equation above. Counterfactual experiments require solving the new equilibrium for either a new set of state variable s or another set of coefficients θ.

Given the state variable s and coefficient θ, the equilibrium is solved by the following algorithm.

The number of low-type potential entrants $\bar{H}_l$ and the number of high-type potential entrants $\bar{H}_h$ are assumed to be publicly known in this game. Let $\Delta_l = \sum_{j \in \omega_l} \sigma_j$ denote the number of low-type entrants to the focal platform and similarly Δ_h is the number of high-type entrants choosing this platform. Therefore, once Δ_l and Δ_h are realized, we immediately know how many apps choose the other platform.

1. Given Δ_l and Δ_h, update post-entry players, $H_{1l} = H_{0l} + \Delta_l$ and $H_{1h} = H_{0h} + \Delta_h$.
2. Calculate σ_i. Every variable in (1.50) is known, so it is easy to calculate $\hat{\sigma}_i$ through the logit function stated above. Alternatively, I draw two independent Type I extreme random variables ϵ_1 and ϵ_2, and calculate $\hat{\sigma}_i = 1\left(\Pi_i(a_i = k, s; \theta) - \Pi_i(a_i = 1 - k, s; \theta) > \epsilon_2 - \epsilon_1\right)$. This indicator function makes this iteration an integer programming problem. Alternative, I can calculate the acutal probability by the logit function and let $\hat{\sigma}_i = 1$, if the probability is greater than 0.5.
3. If $\mid \Delta_l - \sum_{j \in \omega_l} \hat{\sigma}_j \mid + \mid \Delta_h - \sum_{j \in \omega_h} \hat{\sigma}_j \mid \leq 1$, this set of $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_n\}$ is reported as an equilibrium.
4. Repeat 1- 3 for $\{(\Delta_l, \Delta_h) : 0 \leq \Delta_l \leq \bar{H}_l, 0 \leq \Delta_l \leq \bar{H}_l, \Delta_l \in N, \Delta_h \in N\}$.

Large Smartphone User Base and Consumer Willingness to Pay

The estimation results already show that the smartphone user base has a very strong impact on the platform choice. The relatively small user base is a serious issue for new app platforms. For instance, before Google started to take off in 2011, small Android phone user base was associated with the slow growth of Google app store. The similar correlation can also be found in Windows phone and its app store. There are several effective strategies allowing platforms to expand its smartphone user base. For instance, Android operating system is free for phone manufactures. In 2013, Microsoft acquired Nokia which makes it possible to subsidize the Microsoft phone. Given everything else unchanged, this policy experiment examines how the potential entrants adjust their platform choices facing a 5% larger user base in one of the platforms.

This policy is equivalent with increasing user profitability by 5%, so this experiment can also simulate the effect of higher user engagement. Google app store is widely believed to be less profitable than Apple, mainly because of Android phone user's low willingness to pay and lower user engagement. In fact, Google has been trying to improve user experience through enhancing the payment system, cleaning up malware and defragging different Android operating systems. Given that Android phone user base is four times as large as iPhone user base, slight changes in users' willingness to pay can result in substantial changes in developers' platform choice decision.

Table 1.14: Change of Entrants by Policy Experiments

Policies	Platforms	Apple Entrant			Google Entrant		
		L	H	ALL	L	H	ALL
5% More User	Apple	2.1%	2.5%	2.1%	-1.9%	-1.1%	-1.8%
	Google	-2.6%	-3.4%	-2.6%	2.4%	1.4%	2.3%
5% Fewer Low-Type Incumbents	Apple	-1.1%	5.3%	-0.7%	1.1%	-2.2%	0.6%
	Google	1.1%	-3.9%	0.8%	-1.1%	1.6%	-0.7%
5% More High-Type Incumbents	Apple	-0.1%	-2.2%	-0.2%	0.1%	0.9%	0.2%
	Google	0.1%	-0.4%	0.1%	-0.1%	0.2%	-0.1%
5% More User &	Apple	0.9%	8.0%	1.4%	-0.9%	-3.4%	-1.2%
5 % Fewer Low-Type Incumbents	Google	-1.5%	-7.2%	-1.8%	1.4%	3.0%	1.6%
5% Less High-Type Competition &	Apple	-1.2%	7.8%	-0.6%	1.1%	-3.3%	0.5%
5 % Fewer Low-Type Incumbents	Google	1.1%	-3.2%	0.8%	-1.0%	1.3%	-0.7%

The policy experiments results are summarized in Table 1.14. Increasing iPhone users (or consumer willingness to pay) by 5% leads to overall 2.1% more apps choosing Apple, in which the high-type entrants increase by 2.5%. Similarly, 5% larger Android phone user base (or consumer willingness to pay) results in 2.3% more apps entering Google, but only 1.4% more high-type entrants. Although Android phone user is more effective on increasing the overall entrants, iPhone user is particularly useful to improve the platform composition, such as increasing the share of high-type apps.

Regulate Low-Type Apps

The profit of an app platform largely depends on high-quality apps, since market demand is highly skewed toward elite apps. However, a great number of low quality apps increase the search cost and impose negative externality on the entire platform. The estimation results show that high-type entrants prefer a platform with fewer junk apps. Moreover, in Google, the presence of low-quality apps induces more entry of low-quality apps. Therefore, it is essential for app platforms to clean up junk apps and make platform more attractive to high-quality entrants . The following two counterfactual experiments concern the effects of app selection mechanisms on the market composition.

The first policy experiment along this line imposes a hypothetical regulation that removes apps having downloads lower than a certain threshold. It is found that the elimination of low-quality incumbents is followed by more high-type entrants and fewer low-type entrants. This policy is much more effective on Apple. 5% fewer low-type incumbents results in 5.3% more high-type entrants, much higher than the 1.6% increase in Google. This policy could reduce low-type entrants by 1.1% in both platforms.

Woo High-Type Apps

Parallel with the regulations on low-quality apps, platform can also woo high-type apps. For example, at the early stage of Android platform, Google hired many talented developers to create Android apps. Microsoft subsidizes developers up to hundreds of thousands of dollars for developing apps for the Microsoft phone. Such actions immediately boost high-type apps available in the platform. However, more high-type apps could increase the competition level, which may have negative impacts on the market evolution.

This policy experiment explores the changes of the platform choices when the high-type apps are increased by 5% respectively in Apple and Google. 5% more high-type incumbents reduce the overall entry slightly (0.1%-0.2%) in both platforms. Interestingly, Apple and Google show some opposite impacts on high-type entrants. This policy brings 0.2% more high-quality entrants to Google, but reduces Apple high-type entrants by 2.2%. This result would be explained by the competition structures in high-end segment in Google and Apple. The estimation results show a considerable competitive effect among Apple high-quality apps, but that competitive effect does not exist in Google. During the data period, Apple is already a very well established platform with a large collection of high-quality apps. On

the contrary, Google had far fewer high-quality apps. Therefore, the same stimulating policy works differently on platforms at different stages. For a platform to takeoff, it would be effective to improve the app composition by introducing more high-type apps, but more high-quality apps in a prosperous platform would exacerbate the already high competition level.

More users and fewer low-type incumbents

The counterfactual experiments above discuss the effect of each single market policy. However, the platform could adopt multiple policies and get the optimal market designs. Some policies combined with others can reach the best efficacy and minimize the "side effect".

This policy bundle examines the effect when smartphone user base increases by 5% and meanwhile 5% low-type incumbents are removed from the app store. In this setting, platform promotes the adoption of hardware while regulating the low-type apps through certain selection mechanisms. Larger user base leads to more apps to enter, but the influx of low-type apps is an undesirable side effect, which can be alleviated by suppressing the entry of low-type apps.

This policy bundle can boost high-type entrants by 8% in Apple and low-type entrants only increase by 0.9%. The policy on promoting hardware adoption induces more low-type entrants than high-type entrants in Google app store, and actually market composition will deteriorate. However, when the promotion of hardware is combined with regulations on low-type apps, high-type apps increase by 3% and low-type entrants only increase by 1.4% in Google.

Lower high-type competition and fewer low-type incumbents

This policy bundle improves app compositions from two directions. On the one side, lower competition level among high-type apps will attract more high-type entrants and on the other side the regulation on low-type apps suppresses the low-type entrants. This policy is very effective in Apple. High-type entrants are boosted by 7.8% while low-type entrants decrease by 1.2%. This policy bundle can also reduce Google low-type entrant by a similar scale, but it is not very effective on inducing high-type apps, which only increase by 1.3%.

1.8 Conclusion

Since Apple and Google launched their online app stores in 2008, the market for mobile apps has experienced rapid growth and represents an enormous business opportunity. The expansion of an app platform relies on the presence of a great variety of high-quality apps. Given the existence of multiple app platforms, a fundamental question in the app industry is how app developers choose which app platform to enter.

This chapter studies the platform choice decisions of app developers and the implications for the evolution of the overall app market, using a unique and big daily-level panel dataset that contains information on every app in the two leading app stores, Apple and Google, over a 2-year period. Based on these data, we construct and estimate a structural model for heterogeneous app developers' platform choice decisions within an incomplete information static game framework. As the platform choice is largely driven by app quality, which is not directly observed in the data, we develop an app adoption model (which consists of an app discovery process and a conditional adoption decision) to recover app quality.

The entry model shows that in both platforms the high-type entrants dislike the platform with more low-type incumbents. It is also found that smartphone user base has a strong impact on the platform choice. However, since these two platforms are at different growth stage, Apple and Google app stores also exhibit some distinctive competition structures. In Google, the presence of low-quality apps induces more low-quality apps to enter, but in Apple the presence of high-quality apps suppresses the entry of other high-quality apps.

These market characteristics and competition structures uncovered by the structural model provide very rich implications regarding the platform designs. The policy experiments find that eliminating 5% low-quality incumbents can increase high-quality entrants by 5.3% in Apple and 1.6% in Google. And the regulation on low-quality incumbents also discourages low-quality entrants. Boosting the smartphone user base or consumer willingness to pay are both effective to expand the platform, but these stimulating policies also induce the entry of more low-quality apps. The optimal market designs targeting at expanding the collection of high-quality apps should supplement stimulating measures with app elimination mechanisms so that the market evolution will not be undermined by the influx of junk apps.

The contributions of this research are as follows. Theoretically, we provide insights into both app demand and app supply. On the demand side, we supplement the existing literature on app demand by considering search cost. On the supply side, this research adds to the few studies on the mobile app entries. Substantially, this structural model offers policy recommendations for app platforms regarding various design mechanisms. Methodologically, we integrate state-of-the-art machine learning techniques with structural estimation techniques to derive model primitives, reducing the computational burden associated with big data.

There are several limitations to the present study that also represent opportunities for future research. First, this chapter reveals the different market and competition structures between Apple and Google app stores, but it has not explained causes of these differences. The understanding of competing app platforms will be enriched through further explorations on the relationship between ranking or featuring mechanisms and the platform evolution. Second, we assume that app heterogeneity is compressed into one single variable, app quality. A better measure of app heterogeneity would bring deeper insights into this market. User reviews contain rich information about app characteristics and relying on text mining techniques, app characteristics on more dimensions could be extracted and quantified.

Chapter 2

Do Firms Benefit From Disclosing Innovations?

2.1 Introduction

Patents and trade secrets are the two major means of intellectual property protection. In the patent system, inventors place inventions to the public domain in exchange for legal protection of the intellectual property. Trade secrecy secures intellectual property by keeping it from the public. However, owners of knowledge sometimes do not file patents or keep their innovations secret. Instead they disclose their private knowledge by exposing them to the public domain. Such behaviors are called knowledge disclosure. Once disclosed, the innovation is no longer patentable and could be accessed by others free of charge. Private knowledge can be disclosed in various forms, such as publications on research journals, conference presentations, or publications on the Internet. In academia, there has been a long history of publishing knowledge, which has greatly expedited the diffusion of new knowledge and driven following innovations. In the very competitive business world, however, it is hard to explain the large scale of knowledge disclosures by conventional wisdom, since firms can always choose to apply for patents or to keep innovations to themselves.

Existing literature offers several explanations for open knowledge disclosure. For instance, strategic disclosure could deter potential competitors in innovation competition. By partially disclosing innovation or private knowledge, a firm sends signals to its rivals, that the disclosing firm has advantages on this technology and is confident of its innovation. As a result, rivals may stop competing with the disclosing firm. Besides the strategic deterrence, disclosure can extend patent races, because disclosure prevents anyone, including the disclosing firm itself, from applying for patents based on this disclosed innovation unless a significant progress is made. Therefore, by disclosing part of an ongoing innovations, firms make it more difficult for both themselves and competitors to obtain patents on the disclosed innovations. This action further may extend any ongoing patent race and earn extra time for the disclosing firm. Innovation disclosure can also send signals to potential buyers,

by revealing valuation of a given innovation to potential buyers. Without knowing the details, potential buyers cannot correctly evaluate a piece of innovation. However, once buyers get to know the idea, they no longer need to purchase it. This creates a paradox for the unpatentable innovations. Therefore, a seller may choose to disclose part of its ideas or innovations to help potential buyers make an evaluation. Disclosure can also create spillovers to upstream suppliers or downstream producers. For instance, if an automobile manufacturer discloses an innovation on steel production, this will enhance the efficiency of its upstream supplier's steel production process. In turn, the automobile manufacturer can also benefit from the enhanced quality or efficiency in production. Likewise, a spillover effect could also be realized for the downstream producers. When an innovation has profound industry-wide influence, then disclosures could expedite the adoption of certain technologies or even establish industry standards. Disclosing firms may in turn benefit from demand increase and/or cost reduction. Among these explanations, strategic disclosure has been extensively studied using theoretical models. However, under many circumstances, strategic disclosure might not be possible. Henkel and Pangerl 2008 interviewed IP managers in several multinational firms and revealed that it is very hard for firms to know the details of competitors' R&D, such as the stages, obstacles or progress. Without such information, it is unrealistic to take strategic moves or disclose strategically. In many cases, therefore, firms will not be able to disclose innovations strategically.

Most existing studies on knowledge disclosure take a theoretical perspective, and there have been few empirical studies on this issue. We supplement this literature by providing empirical evidence on knowledge disclosure, which is made feasible by a very novel data set covering all IBM's innovation disclosures and patents. By matching this data set with the NBER patent database, we not only observe several important patents characteristics, but can also infer how knowledge diffuses, how the innovation evolves and more importantly, how firms utilize patents and innovation disclosures differently.

Relying on these unique data, this chapter first empirically tests the strategic explanations for innovation disclosures. If firms disclose their innovations due to strategic reasons, a likely consequence is that disclosures undercut other firms' related patents. In particular, under strategic explanations, patents citing disclosures would be weaker than other patents in the same technical field. Therefore, this research will examine the strength of patents built upon IBM disclosures and the strength of other comparable patents not citing IBM disclosures. Further this research will study the diffusion of IBM disclosures and the utilization of disclosures along the diffusing path. Based on several empirical findings, we suggest an alternative explanation for innovation disclosures. That is, innovation disclosures could elicit follow-up innovations, and disclosing firms can benefit from follow-up innovations built upon their disclosures. Through selectively disclosing peripheral innovations and learning from follow-up innovations, disclosing firms can extend and strengthen their patent portfolios.

This chapter is organized as follows. The next section reviews related literature. The following section documents the data used for this study and essential data preparations. The section after provides data summary statistics, followed by the empirical section. The last section concludes.

2.2 Related Literature

There has been considerable prior work on knowledge disclosure. A major strand of the literature uses theoretical models to explain why firm discloses innovation for free. One explanation is that firms disclose innovations for defensive purposes, including strategic deterrence (e.g., Johnson 2004; Gill 2008) and extension of patent races (e.g., Baker and Mezzetti 2005, Johnson 2004, Gill 2008). For instance, Johnson 2004 illustrates disclosures as a defensing mechanism against "patent pirates" and guarding against bad patents. In addition, he finds innovations that are less-technically challenging and easier to innovate around tend to be better candidates for publishing. Gill 2008 examines the incentives for a leading firm in a research competition to release its intermediate R&D results. He finds disclosures signal commitment to the research project which would induce a lagging rival to exit the contest. Baker and Mezzetti 2005 shows that by making the innovation more difficult to patent, disclosure extends the patent race, and gain the disclosing firm advantages in a patent race.

Disclosures can also reveal the value of innovations to potential buyers or financial markets (e.g., Bhattacharya and Ritter 1983; Anton and Yao 2002; Anton and Yao 2004). For instance, Bhattacharya and Ritter 1983 show that disclosure can be valuable as it acts as a credible signal to financial markets of the firm's innovation prospects. Anton and Yao 2002 argue that disclosure of some knowledge serves as a signal to the value of non-disclosed knowledge, which helps buyers evaluate the innovation.

Disclosure can also create spillovers to upstream suppliers or downstream producers, which in turn benefits disclosing firms (e.g., Pénin 2007)

A second strand of literature studies the knowledge disclosure through surveys, interviews or descriptive analysis. For instance, Henkel and Pangerl 2008 surveyed and interviewed IP managers in 37 German firms cross different industries and several EPO patent examiners and judges. They conclude that disclosures are widely used to secure firms' free right to operate, and firms rarely disclose private knowledge out of strategic concerns. This finding implies that disclosures usually have close connections to the disclosing firm's existing patents, and these disclosures are protected by the firm's patent portfolio. Bhaskarabhatla and Pennings 2012 compare IBM disclosures and patents before and after several antitrust lawsuits, and claim that IBM was likely to disclose its innovations to deal with the antitrust lawsuits initiated by the Department of Justice.

The existing literature offers a great start for our foray into the problem of knowledge disclosure, but our work differs from existing literature in several aspects. First, to our best knowledge, this research is is the first to address the question of knowledge disclosure through econometric methods. Relying on the very unique data, this research empirically tests the strategic defensive disclosures. Second, we provide an alternative explanation for knowledge disclosure. Namely, the disclosing firm gains by learning from innovations built upon its knowledge disclosures. This claimed mechanism is supported by rigorous empirical evidence.

2.3 Data Description

Since 1958, IBM has been disclosing its innovations on IBM Technical Disclosure Bulletin (IBM TDB hereafter), a journal published by IBM itself. These published innovations are called IBM disclosures. Once these innovations are published to the public, they become part of the "prior art" and IBM automatically gives up its rights on filing patents on them. These disclosures can also be used by others freely. As described by (Bhaskarabhatla and Pennings 2012), the in-house selection process on disclosing is implemented by a committee of IBM innovative directors and IP attorneys. IBM engineers send their innovations to this committee, and this committee decides which ones to be published on IBM TDB and which ones to send to the US Patent Office. Of course, the innovations selected for patent filing need to be rewritten and prepared for the following patent application. The innovations for publication are usually modified slightly and then published in a length of 2-4 pages. IBM published the first issue of IBM TDB in 1958. Up to 2000, around 80,000 pieces of innovations had been published on IBM TDB. Unlike patents, IBM disclosures tend to be informal and only contain title, inventor name, disclosing year and the content. IBM disclosures published on IBM TDB do not include the International Patent Classification, which is generally revealed in patents. In addition, IBM disclosures do not list references, so the citations of disclosures are also unavailable. Even though IBM disclosures do not include references, patents citing these disclosures provide citations. Therefore, the missing information could be partially supplemented by connecting IBM disclosures with patents.

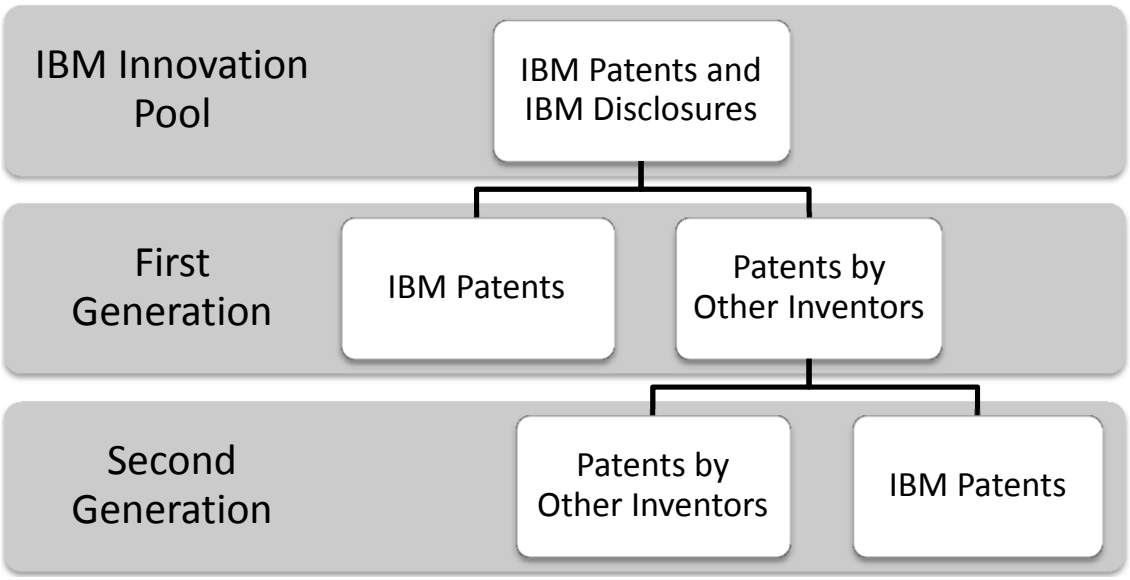
A citing tree originated in IBM innovations is shown in Figure 2.1. This citing tree also enables us to study the evolution and influence of IBM disclosures. This figure illustrates the citing relationship between IBM patents, IBM disclosures and other US patents. IBM patents and IBM disclosures comprise the IBM innovation pool, the root of the citing tree. The first generation of the citing tree includes patents that cite either IBM patents or IBM disclosures. The second generation is derived from the first generation, and includes patents that cite the first-generation patents. We focus on the second-generation patents filed by firms other than IBM. [1]

This citing tree relies on two databases. The first one is Thompson's patent database. Besides many other features, this database covers the complete references cited by each patent. These references include both patents and non-patent references, such as academic papers and innovation disclosures. The second database is the NBER patent database, which provides rich information on patents filed from 1976 to 2006. The same database also records patent citations from 1976 to 2006. Because citations take time to accumulate, we drop the latest 6 years and use the data from 1976 to 2000.

To create this tree-structured data, we need to connect disclosures with US patents, which also will enable us to infer several essential characteristics of IBM disclosures. This match is achieved by two steps. First, we search non-patent citations of all US patents

[1]This citing tree could keep growing. However as the tree becomes bigger, patents in lower generation become less relevant to the original innovation pool. For the purpose of studying knowledge diffusion, the citing tree is restricted to the first two generations.

Figure 2.1: The Citing Tree



through Thompson's patent database and extract all US patents citing IBM disclosures. This step establishes the basic linkage between IBM disclosures and patents. Based on the patent citation data in NBER patent database, we could further build the second generation patents originated in IBM disclosures. The IBM innovation pool on the very top of the citing tree also consists of IBM patents, so we separate patents filed by IBM from NBER patent database. Similarly, using the citation data in NBER patent database, we could build the first generation and the second generation patents originating in IBM patents.

Table 2.1 provides summary statistics on IBM disclosures and IBM patents. In terms of quantity, IBM patents are more than twice as many as IBM disclosures and cited more.The average citation is the number of citations averaged cross all IBM patents or IBM disclosures. The average citation received by IBM patents is much higher than the average citation by IBM disclosures, suggesting that IBM disclosures are likely to be less valuable and less significant. In other words, IBM preserves the high quality innovation for patents and only discloses less substantial ones. The citing gap is defined as the number of years between a patent's application year (a disclosure's publishing year) and the application year of patents citing this patent (disclosure). This gap reflects the diffusion speed. The average citing gap for IBM disclosures is about 15 months longer than IBM patents, indicating a higher diffusion speed for IBM patents. Even though the descriptive statistics show that IBM disclosures are less substantial and of lower quality than IBM patents, it does not mean that disclosures are not valuable. IBM disclosures were published regularly and systematically for about 50 years. The volume of IBM disclosures alone exceeds that of patent applications by any other firms during the same period, except Canon. The average citation received by these disclosures is above 60% of US patents applied during the same period.

As shown in Table 2.2, more than 60% of IBM patents are cited by IBM. In comparison,

Table 2.1: Summary Statistics of IBM Disclosures and Patents

	No.	No. of Citing Patents	Average Citations	Average Citing Gap (Year)
IBM Disclosures	14,008	23,004	2.13	7.64
IBM Patents	31,424	150,496	15.68	6.42

This first column is the number of IBM disclosures and IBM patents used for this research. The second column reports the number of patents citing IBM disclosures or IBM patents. The third column calculates the average citations received by IBM patents or IBM disclosures. The last column shows the average years taken to be cited by other patents.

only about 40% IBM disclosures are cited by IBM. This implies that IBM disclosures are less central to IBM's overall innovation portfolio, which is consistent with the low-quality nature of disclosures. Moreover, patents cited by IBM receive more citations from others. For instance, on average, IBM patents cited by IBM receive 16.61 citations from other inventors, compared with the 7.76 non-self citations received by patents not cited by IBM. The same pattern exists for IBM disclosures, but at smaller magnitudes. This may indicate that innovations concentrate in some directions and technologies.

Table 2.2: Citation of IBM Disclosures

		Percentage	Average Non-self Citations
IBM Disclosures	Cited by IBM	44.40%	2.41
	Not Cited by IBM	55.60%	1.91
IBM Patents	Cited by IBM	62.63%	16.61
	Not Cited by IBM	37.37%	7.76

The "Percent(%)" in the second column represents the ratio of IBM disclosures or IBM patents cited by IBM itself. The last column shows the average citations which are not made by IBM.

Table 2.3 reports the summary statistics of patents in the first generation. More than 90% of patents in the first generation are derived from IBM patents, even though IBM patents are only about twice as many as IBM disclosures. More detailed analyses reveal that IBM cites patents in the first generation differently. The first generation patents that cite both IBM patents and disclosures are utilized by IBM the most. Patents derived from only IBM patents are also frequently cited by IBM as well, but about 20% less. The first generation patents originated in only IBM disclosure is the least used by IBM.

Because some patents in the first generation are IBM's own patents, to better examine the innovation diffusion in other firms, the bottom panel in Table 2.3 excludes the first generation patents filed by IBM itself. The general pattern found in the full sample does not change. Noticeably the number of patents citing both IBM patents and disclosures reduces by over 40%, from 11205 to 7142, but the number of patents in other categories only decreases by around 10%. It seems that patents originated in both IBM patents and disclosures are more

Table 2.3: Patents Citing IBM Patents or IBM Disclosures

Including Self Citations

Group	No. of Patents	Percentage of Patents Cited by IBM
Patents Citing IBM Disclosure only	11,799	29.28%
Patents Citing IBM Patents only	139,291	37.05%
Patents Citing Both	11,205	55.53%

Excluding Self Citations

Group	No. of Patents	Percentage of Patents Cited by IBM
Patents Citing IBM Disclosure only	10,668	25.69%
Patents Citing IBM Patents only	122,579	33.28%
Patents Citing Both	7,142	44.34%

extensively cited by IBM. IBM patents and disclosures seem to be complements for IBM, as patents citing both are most intensively used by IBM.

One important aspect of this research is to study the innovation diffusion originated in IBM disclosures and patents. Thus it is essential to measure the relative closeness of follow-up innovations to IBM patent portfolio. To do so, we create a distance measure called the technical distance. There are 883 IPC technical sub classes in patent classification system. To determine the related technical fields of one particular patents group, we calculate the distribution of technical classes for some patents group. Then for each patents group, the vector with 883 elements (technical vector) represents its position in the IPC technical space. Based on this vector, it is easy to calculate the Euclidean distance between any two groups. Suppose $V_t = (v_{1t}, v_{2t}, \ldots, v_{nt})$ represents the technical distribution of IBM patent portfolio in year t, where v_{it} is the percentage of IBM patents in IPC class i in year t. Similarly, $X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})$ is the technical vector of patent j. Then the technical distance is defined as $d_{jt} = ||V_t - X_j|| = \sqrt{\sum_{i=1}^{n}(v_{it} - x_{ij})^2}$.

Table 2.4 lists the distance matrix between different groups covering all levels. Among these pairs, IBM patents are very close to patents citing IBM patents, the first generation patents in the citing tree. In contrast, IBM patents are very far away from patents citing IBM disclosure with a technical distance of 14.95. This suggests that IBM disclosures are not very essential in IBM patent portfolio. The technical distance gets smaller to 7.12 between IBM patents and the first generation patents citing IBM disclosures and meanwhile cited by IBM. Most interestingly, although both citing IBM disclosures, patents citing disclosures are very distant from patents that cite IBM disclosures and also cited by IBM disclosures, with a distance of 19.46. This distance matrix shows that IBM disclosures lie in the peripheral

Table 2.4: Technical Distance Between Patents Groups

	IBM Patents	Patents Citing IBM Patents	Patents Citing IBM Disclosure	Patents Citing Disclosures and Cited by IBM
IBM Patents	0	4.66	14.95	7.12
Patents Citing IBM Patents		0	18.58	4.40
Patents Citing IBM Disclosures			0	19.46
Patents Citing Disclosures and Cited by IBM				0

region of the entire IBM innovative realm. When IBM selectively uses patents built upon its own disclosures, the distance gets much closer but not as close as the distance between IBM patents and patents cited by IBM patents.

2.4 Empirical Strategies

This section first tests the strategic explanation of innovation disclosures. Existing literature shows that defensive disclosures hurt competitors in patent races or innovation competition. In the strategic context, disclosures could keep competitors from getting patents or undercut the strength of their patents on the related technical field. Unfortunately, we could not observe a firm's innovation process, so it is impossible to examine if disclosures block any competitor's patents. On the other hand, competitors' innovations which already filed patents are observable along with patent characteristics. By identifying disclosure's impacts on these patents, we will be able to verify the strategic explanation of innovation disclosures.

The descriptive analyses in the preceding section show that IBM discloses its innovations on a large scale. Usually the peripheral innovations are published on IBM TDB. Later on, IBM cited the follow-up patents which were built on its disclosures and not distant from IBM patent portfolio. Based on these descriptive findings, the empirical section further provides support on this finding and offers an alternative explanation for firms' innovation disclosures.

The first part of this section provides empirical tests on the strategic explanations of disclosures, and the second part explains how IBM could benefit from its disclosures.

Are IBM Disclosures Offensive?

Claim is a measure of patent's scope and efficacy. The more claims a patent has, the stronger and wider the patent is. Therefore, if disclosures undercut competitors in patent races, competitors' patents built on IBM disclosures will end up with fewer claims, compared with other similar patents but not built on disclosures. For patents citing IBM disclosures, we construct a control group which includes patents by the same assignees, same application year and same IPC sub-class as those patents citing IBM disclosures, but not citing IBM disclosures.

Table 2.5: The Number of Claims of the First Generation Patents

Variables	Number of Claims		
	Full Sample	IBM Patents Excluded	IBM Patents Only
Cite Disclosure	0.729	0.846	0.818
	(0.096)***	(0.113)***	(0.176)***
No. of Backward Citations	0.131	0.131	0.114
	(0.003)***	(0.003)***	(0.011)***
No. of Forward Citations	0.092	0.094	0.059
	(0.004)***	(0.004)***	(0.004)***
Constant	9.101	8.999	10.258
	(0.165)***	(0.178)***	(0.424)***
Application Year Dummy	Yes	Yes	Yes
IPC Class Dummy	Yes	Yes	Yes
R^2	0.10	0.10	0.11
N	567,033	536,175	30,858

Robust standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

The empirical equation is as follows:

$$cl_i = \alpha + \beta_1 ifdc_i + \gamma X_i + \lambda_t + \eta_j + \epsilon_i, \tag{2.1}$$

where cl_i is the number of claims of patent i. $ifdc_i$ is a binary variable indicating whether patent i cites any IBM disclosures. X_i is a vector of control variables, including backward and forward citations of patent i. λ_t is the application year dummy and η_j is the IPC class dummy. ϵ_i is an idiosyncratic error. The OLS estimation results are listed in Table 2.5. The first column shows the estimation for the full sample. Compared with the control group, the first generation patents citing IBM disclosures have more claims, which does not support the strategic explanation. The second column removes all IBM patents and the third column is the estimation on only IBM patents. Across these three sample groups, patents citing IBM disclosures always have more claims.

A patent could cite IBM patents, IBM disclosures, both, or neither, so there are four different groups: patents citing both IBM patents and disclosures, patents only citing IBM patents (*ifimbpat*), patents only citing IBM disclosures (*ifboth*) and patents citing neither (*ifnone*). In the second group of regressions, the citing relation is broken down into these four cases and patents only citing IBM disclosures are set as the base. $ifimbpat$, $ifboth$ and $ifnone$ are binary indicators of these cases.

$$cl_i = \alpha + \beta_1 ifibmpat_i + \beta_2 ifboth_i + \beta_3 ifnone_i + \gamma X_i + \lambda_t + \eta_j + \epsilon_i \quad (2.2)$$

The OLS estimation results are listed in Table 2.6. Patents only citing IBM disclosures have more claims than patents citing neither IBM patents nor disclosures. Moreover, we find that patents built upon IBM disclosures or IBM patents also have more claims than those not citing IBM innovations. This result does not show any evidence that IBM disclosures jeopardize other firms' patents. In contrast, IBM disclosures lead to higher quality patents. Among patents citing IBM innovations, patents citing both IBM patents and IBM disclosures have the most claims, followed by patents citing only IBM patents. Patents citing only IBM disclosures have fewer claims than patents citing IBM patents, and this is mainly because innovation contained in IBM patents is more substantive than IBM disclosures. The second column of Table 2.6 presents the results when IBM patents are excluded from the first generation. The effects become stronger. The last column shows the results on IBM patents. IBM patents citing its own patents and disclosures are stronger than IBM patents not built on previous IBM innovations. However, citing IBM patents or IBM disclosures do not have distinct impacts on the quality of IBM patents. After all, empirical results provide no evidence showing that IBM disclosures weaken citing patents, so that the strategic explanation of disclosures does not hold in our data.

How IBM Benefits from Disclosing Its Innovations?

This section compares the first-generation patents originated in IBM disclosures and those originated in IBM patents. Based on these analyses, we propose an alternative explanation of innovation disclosures. Further, we examine how IBM utilizes the first-generation patents differently and its impact on the second-generation IBM patents. The first set of regressions in this part examines the technical distances of the first-generation patents among different groups. The sample used for these regressions is the same as before, which includes all first-generation patents and all other patents by the same assignees with the same IPC class and the same application year.

Using this sample, we estimate the following equation:

$$d_i = \alpha + \beta_1 ifibmpat_i + \beta_2 ifboth_i + \beta_3 ifnone_i + \gamma X_i + \lambda_t + \eta_j + \epsilon_i, \quad (2.3)$$

where d_i is the technical distance between patent i and IBM patent portfolio in patent i's application year. The explanatory variables are the same as (2.2). The regression results are summarized in Table 2.7. The first-generation patents citing IBM patents and IBM disclosures have the shortest technical distance to IBM patent portfolio. Particularly, patenting citing both are the closest to IBM patent portfolio. Patents citing only IBM disclosures are relatively distant from patents citing IBM patents. Patents citing neither IBM patents nor IBM patents have the greatest technical distance to IBM patent portfolio. This pattern does not change when IBM patents are removed from the sample, as shown in the second column. For IBM patents, we do not observe differential effects on the distance to IBM

Table 2.6: The Number of Claims of the First Generation Patents

Variables	Number of Claims		
	Whole Sample	IBM Patents Excluded	IBM Patents Only
Cite Neither IBM patents nor Disclosure	-0.757	-0.780	-0.862
	(0.115)***	(0.123)***	(0.317)***
Only Cite IBM patents	0.226	0.271	-0.199
	(0.120)*	(0.129)**	(0.312)
Cite Both IBM patents and Disclosure	0.665	1.036	0.543
	(0.188)***	(0.241)***	(0.348)
No. of Backward Citations	0.124	0.124	0.106
	(0.003)***	(0.003)***	(0.011)***
No. of Forward Citations	0.090	0.092	0.059
	(0.003)***	(0.004)***	(0.004)***
Constant	9.767	9.710	10.944
	(0.198)***	(0.213)***	(0.507)***
Application Year Dummy	Yes	Yes	Yes
IPC Class Dummy	Yes	Yes	Yes
R^2	0.10	0.10	0.11
N	567,033	536,175	30,858

Robust standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

patent portfolio between the two groups citing IBM disclosures. However, IBM patents not built upon any IBM innovations are significantly further away from IBM patent portfolio, compared with patents citing any IBM innovations.

IBM disclosures do not contain IPC classes. Otherwise we could compute the technical distance between IBM disclosures and IBM patents directly. However, by comparing the technical distances of patents citing IBM disclosures and those citing IBM patents, we could infer that IBM disclosures are less central to IBM patent portfolio. In other words, IBM discloses peripheral innovations. Or at least, IBM selectively discloses innovations which are less likely to be used by other firms to create patents close to the core of IBM patents. However, IBM's own patents are not impacted by selective disclosure, because the effects disappear in the regression on IBM patents.

The first generation patents built on IBM disclosures and patents will become inputs of other patents in the second generation. We have shown that IBM patents and IBM disclosures lead to different patents in the first generation, hence we next explore if these innovations are exploited by IBM in different ways. The citing relations between the first-

Table 2.7: Technical Distance of The First Generation Patents

Variables	**Technical Dist**		
	Full Sample	IBM Patents Excluded	IBM Patents Only
Only Cite IBM patents	-0.449	-0.442	0.083
	(0.074)***	(0.077)***	(0.261)
Cite Both IBM patents and Disclosure	-0.852	-0.831	-0.082
	(0.106)***	(0.123)***	(0.283)
Cite No IBM patents nor Disclosure	0.405	0.341	0.77
	(0.071)***	(0.073)***	(0.268)***
No. of Backward Citations	0.005	0.004	0.011
	(0.001)***	(0.001)***	(0.006)*
No. of Forward Citations	0.016	0.013	0.031
	(0.001)***	(0.001)***	(0.003)***
Constant	95.571	95.917	86.612
	(0.219)***	(0.229)***	(0.699)***
Application Year Dummy	Yes	Yes	Yes
IPC Class Dummy	Yes	Yes	Yes
R^2	0.85	0.84	0.86
N	567,030	536,173	30,857

Robust standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

generation patents and the second-generation patents could reveal how IBM learns from innovations built upon its patents and disclosures. We still use the same sample as before. The dependent variable is a binary variable indicating if a patent is cited by IBM. Besides the explanatory variables used before, this regression also includes cross terms of technical distance and the citation binary variables in order to capture differential effects of technical distance on IBM's citing decisions cross patent groups.

$$\begin{aligned} ifcite_i &= \alpha + \beta_1 ifibmpat_i + \beta_2 ifboth_i + \beta_3 ifnone_i + \beta_4 d_i + \\ &\quad \beta_5 d_i * ifibmpat_i + \beta_6 d_i * ifboth_i + \beta_7 d_i * ifnone_i + \gamma X_i + \lambda_t + \eta_j + \epsilon_i \quad (2.4) \end{aligned}$$

The equation above is estimated as a linear probability model and the results are presented in Table 2.8. We see that among the first-generation patents, patents citing IBM's

patents or disclosures are more likely to be cited by IBM. This effect is particularly strong for the patents citing both IBM patents and disclosures. Along with the previous evidence, this finding once again implies that IBM disclosures and IBM patents are complements. Together, they not only lead to stronger patents, but also elicit more relevant and useful patents for IBM. Patents citing only IBM disclosures are cited by IBM more than patents not built upon any IBM innovations at all.

The coefficients of interactive terms suggest that if the first-generation patents cite IBM patents, IBM is more likely to cite those patents closer to its patent portfolio., compared with the first-generation patents citing IBM disclosures only. This effect does not appear for the first-generation patents citing only IBM disclosures. In other words, even though patents citing IBM disclosures are more distant from IBM patent portfolio, when IBM cites these patents, it does not necessarily cite those closer to IBM patent portfolio. Such selective citing will help IBM extend the boundary of its innovation domain. The scale of these effects becomes larger when IBM patents are excluded. However, most of these effects disappear in the regression on IBM patents.

So far, the empirical evidence has shown that IBM disclosures are peripheral to IBM patents, and IBM is more likely to cite patents built upon IBM disclosures, especially those not very close to IBM patent portfolio. A natural question that follows is the consequence of the selective citing and its impact on IBM patents in the second generation. For example, will the peripheral patents in the first generation lead to less coherent IBM patents in the second generation? The last set of regressions in this section focuses on the technical distances of IBM patents in the second generation. Sample used for these regressions include all IBM patents in the second generation, that is, all IBM patents that cite the first-generation patents citing IBM patents or IBM disclosures in the root.

According to the citing relations of the first generation patents, we can create series of binary variables indicating whether the second-generation IBM patents cite patents citing IBM patents or disclosures in the root. Therefore, $ifimbpat_i$, $ifboth_i$ and $ifnone_i$ in (2.5) have the same meaning as in previous equations. They refer to the citation relationship of the first-generation patents instead of the direct citation relationship of IBM patents in the second generation. d_i is the technical distance calculated before. OLS estimation is reported in Table 2.9.

$$d_i = \alpha + \beta_1 ifibmpat_i + \beta_2 ifboth_i + \beta_3 ifnone_i + \gamma X_i + \lambda_t + \eta_j + \epsilon_i \tag{2.5}$$

However, IBM patents in the second generation could cite IBM patents or IBM disclosures directly. Therefore, we create three groups of samples. The first sample includes all IBM patents in the second generation regardless of their direct citations. The second group excludes the second-generation IBM patents which cite IBM disclosures directly, as well as the second-generation IBM patents citing the first-generation IBM patents which cite IBM disclosures in the root. This group is labeled as "Indirect Citation Excluded" and reported in the second column of Table 2.9. A more restricted sample group is created in which any second-generation IBM patents directly citing IBM patents or IBM disclosures are excluded.

Table 2.8: How IBM Cites the First Generation Patents

Variables		**If IBM Cites**	
	Full Sample	IBM Patents Excluded	IBM Patents Only
Tech Distance	-0.036	0.001	-0.126
	(0.029)	(0.030)	(0.078)
Only Cite IBM patents	10.637	14.382	-7.459
	(2.813)***	(2.966)***	(6.931)
Cite Both IBM patents and Disclosure	22.261	22.970	2.983
	(3.266)***	(3.686)***	(7.274)
Cite Neither IBM patents Nor Disclosure	-16.607	-11.859	-14.327
	(2.802)***	(2.948)***	(7.159)**
Distance * Only Cite IBM patents	-0.057	-0.098	0.118
	(0.028)**	(0.029)***	(0.074)
Distance * Cite Both IBM patents and Disclosure	-0.021	-0.106	0.079
	(0.034)	(0.038)***	(0.078)
Distance * Cite Neither IBM patents nor DC	0.064	0.036	0.102
	(0.028)**	(0.029)	(0.076)
No. of Backward Citations	-0.094	-0.067	-0.084
	(0.005)***	(0.005)***	(0.031)***
No. of Forward Citations	0.569	0.55	0.65
	(0.021)***	(0.022)***	(0.020)***
Constant	33.059	23.199	70.662
	(2.967)***	(3.110)***	(7.634)***
Application Year Dummy	Yes	Yes	Yes
IPC Class Dummy	Yes	Yes	Yes
R^2	0.23	0.21	0.17
N	567,030	536,173	30,857

Robust standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Table 2.9: How IBM Cites the Second Generation Patents

Variables	**Technical Distance**		
	Full Sample	Indirect Citation Excluded	Direct Citation Excluded
Only Cite IBM patents	-0.435	-1.834	-1.834
	(0.337)	(0.427)***	(0.490)***
Cite Both IBM patents and Disclosure	-0.923	-1.970	-2.040
	(0.338)***	(0.464)***	(0.529)***
Cite No IBM patents nor Disclosure	-0.165	-0.333	-0.430
	(0.347)	(0.423)	(0.484)
No. of Backward Citation	0.013	0.133	0.137
	(0.006)**	(0.019)***	(0.023)***
No. of Forward Citation	0.031	0.030	0.027
	(0.003)***	(0.005)***	(0.005)***
Constant	87.135	92.820	93.185
	(0.660)***	(0.673)***	(0.778)***
Application Year Dummy	Yes	Yes	Yes
IPC Class Dummy	Yes	Yes	Yes
R^2	0.86	0.81	0.82
N	31,423	12,534	9,518

Robust standard errors in parentheses, * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Patents in the second group could cite the first-generation IBM patents as long as these IBM patents do not build on IBM disclosures. However, the third sample group excludes any direct or indirect connections with IBM disclosures and IBM patents. The third group is labeled as "Direct Citation excluded" in the third column of Table 2.9.

The results indicate that the second-generation IBM patents originated in different innovations in the root have different technical distances to IBM patent portfolio. Even in the most restricted sample, where all direct connections with IBM innovations are excluded, the effects of IBM disclosures and IBM patents could still reach the second generation. For instance, the second-generation IBM patents originated in both IBM patents and IBM disclosures in the root are very close to IBM patent portfolio, followed by IBM patents originated in only IBM patents. IBM patents that originated in only IBM disclosures and IBM patents that are totally unrelated with any IBM innovations sit on the very outer layer of IBM patent portfolio. This pattern becomes sharper in the restricted samples, because contamination through other channels is ruled out. The last empirical evidence proves that IBM disclosures do help IBM extend its innovation domain by learning from follow-up patents from other firms.

2.5 Conclusions

This chapter studies competitive firms' voluntary innovation disclosures. Building upon existing literature, this research is one of the very few empirical studies on innovation disclosures. This research benefits from a very novel and unique set of data with a complete coverage of IBM disclosures, IBM patents and patents citing these IBM innovations. We first examines the strategic explanation of innovation disclosures. Existing literature shows that in a competitive context, a strategic firm could deter competitors or undercut competitors' patents by innovation disclosures. Our empirical analyses indicate that patents citing IBM disclosures in fact have more claims than comparable patents that do not cite IBM disclosures. Thus IBM disclosures do not appear to undercut citing patents, instead, they lead to stronger patents. We do not find empirical evidence supporting the strategic and defensive disclosures.

We further investigate how the disclosures diffuse and their impacts on firms' IP management. The empirical study reveals that disclosures published by IBM tend to be distant from IBM patents and less substantial, but IBM is actively learning from follow-up patents originated in its disclosures. As a result, some IBM patents are on the peripheral of IBM patent domain. Therefore, IBM could extend its innovation boundary and strengthen the its innovation portfolio by selectively disclosing its innovations.

Bibliography

[1] Victor Aguirregabiria and Pedro Mira. "Sequential estimation of dynamic discrete games". In: *Econometrica* 75.1 (2007), pp. 1–53.

[2] Chunrong Ai and Xiaohong Chen. "Efficient estimation of models with conditional moment restrictions containing unknown functions". In: *Econometrica* 71.6 (2003), pp. 1795–1843.

[3] James J Anton and Dennis A Yao. "Little patents and big secrets: managing intellectual property". In: *RAND Journal of Economics* (2004), pp. 1–22.

[4] James J Anton and Dennis A Yao. "The sale of ideas: Strategic disclosure, property rights, and contracting". In: *The Review of Economic Studies* 69.3 (2002), pp. 513–531.

[5] Patrick Bajari, C Lanier Benkard, and Jonathan Levin. "Estimating dynamic models of imperfect competition". In: *Econometrica* 75.5 (2007), pp. 1331–1370.

[6] Patrick Bajari et al. "Estimating static models of strategic interactions". In: *Journal of Business & Economic Statistics* 28.4 (2010).

[7] Scott Baker and Claudio Mezzetti. "Disclosure as a Strategy in the Patent Race*". In: *Journal of Law and Economics* 48.1 (2005), pp. 173–194.

[8] C Lanier Benkard, Przemyslaw Jeziorski, and Gabriel Y Weintraub. "Oblivious Equilibrium for Concentrated Industries". In: *NBER Working Paper* w19307 (2013).

[9] Steven Berry, James Levinsohn, and Ariel Pakes. "Automobile prices in market equilibrium". In: *Econometrica: Journal of the Econometric Society* (1995), pp. 841–890.

[10] Steven Berry and Peter Reiss. "Empirical models of entry and market structure". In: *Handbook of industrial organization* 3 (2007), pp. 1845–1886.

[11] Steven T Berry. "Estimation of a Model of Entry in the Airline Industry". In: *Econometrica: Journal of the Econometric Society* (1992), pp. 889–917.

[12] Ajay Bhaskarabhatla and Enrico Pennings. *Defensive Disclosure under Antitrust Enforcement*. Tech. rep. Tinbergen Institute Discussion Paper, 2012.

[13] Sudipto Bhattacharya and Jay R Ritter. "Innovation and communication: Signalling with partial disclosure". In: *The Review of Economic Studies* 50.2 (1983), pp. 331–346.

[14] Paul A Bjorn and Quang H Vuong. *Simultaneous Equations Models for Dummy Endogenous Variables: A Game Theoretic Formulation with an Application to Labor Force Participation*. Tech. rep. California Institute of Technology, Division of the Humanities and Social Sciences, 1984.

[15] Timothy Bresnahan, Joe Orsini, and Pai-Ling Yin. *Platform choice by mobile apps developers*. Tech. rep. Working paper, National Bureau of Economic Research, Cambridge, MA, 2014.

[16] Timothy F Bresnahan and Peter C Reiss. "Empirical models of discrete games". In: *Journal of Econometrics* 48.1 (1991), pp. 57–81.

[17] Timothy F Bresnahan and Peter C Reiss. "Entry and competition in concentrated markets". In: *Journal of Political Economy* (1991), pp. 977–1009.

[18] Octavian Carare. "The impact of bestseller rank on demand: evidence from the app market". In: *International Economic Review* 53.3 (2012), pp. 717–742.

[19] Federico Ciliberto and Elie Tamer. "Market structure and multiple equilibria in airline markets". In: *Econometrica* 77.6 (2009), pp. 1791–1828.

[20] Shane Cole. *Apple's iOS brings developers 5x more revenue per download than Android*. Nov. 2013. URL: `http://appleinsider.com/articles/13/11/27/apples-ios-brings-developers-5x-more-revenue-per-download-than-android`.

[21] Kevin Corti. *Mobile games discovery - survey findings*. 2013. URL: `http://evil27games.wordpress.com/2014/02/11/mobile-games-discovery-survey-findings/`.

[22] Jean-Pierre Dubé, Günter J Hitsch, and Puneet Manchanda. "An empirical model of advertising dynamics". In: *Quantitative marketing and economics* 3.2 (2005), pp. 107–144.

[23] Carl Eckart and Gale Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218.

[24] Per Engstrom and Eskil Forsell. "Demand effects of consumers' stated and revealed preferences". In: *Available at SSRN 2253859* (2013).

[25] Rajiv Garg and Rahul Telang. "Inferring app demand from publicly available data". In: *MIS Quarterly* 37.4 (2013).

[26] Gartner. *Gartner Says by 2017, Mobile Users Will Provide Personalized Data Streams to More Than 100 Apps and Services Every Day*. 2014. URL: `http://www.gartner.com/newsroom/id/2654115`.

[27] Anindya Ghose and Sang Pil Han. "Estimating Demand for Mobile Applications in the New Economy". In: *Management Science, Forthcoming* (2014).

[28] David Gill. "Strategic disclosure of intermediate research results". In: *Journal of Economics & Management Strategy* 17.3 (2008), pp. 733–758.

[29] Joachim Henkel and Stefanie Pangerl. "Defensive publishing: an empirical study". In: *Available at SSRN 981444* (2008).

[30] Justin P Johnson. "Defensive publishing by a leading firm". In: *Available at SSRN 606781* (2004).

[31] Min Jung Kim. "Essays on the Economics of the Smartphone and Application Industry". PhD thesis. University of Minnesota, 2013.

[32] Yongdong Liu, Denis Nekipelov, and Minjung Park. "Timely versus Quality Innovation: The Case of Mobile Applications on iTunes and Google Play". In: *NBER Working Paper* (2014).

[33] Carl F Mela, Sunil Gupta, and Donald R Lehmann. "The long-term impact of promotion and advertising on consumer brand choice". In: *Journal of Marketing research* (1997), pp. 248–261.

[34] Nielsen. *State of the media: mobile media report Q3 2011*. Dec. 2011. URL: `http://www.nielsen.com/us/en/reports/2011/state-of-the-media--mobile-media-report-q3-2011.html`.

[35] A Pakes et al. "Moment Inequalities and Their Application". In: (2011).

[36] Ariel Pakes, Michael Ostrovsky, and Steven Berry. "Simple estimators for the parameters of discrete dynamic games (with entry/exit examples)". In: *The RAND Journal of Economics* 38.2 (2007), pp. 373–399.

[37] Julien Pénin. "Open knowledge disclosure: an overview of the evidence and economic motivations". In: *Journal of Economic Surveys* 21.2 (2007), pp. 326–347.

[38] Katja Seim. "An empirical model of firm entry with endogenous product-type choices". In: *The RAND Journal of Economics* 37.3 (2006), pp. 619–640.

[39] Surikate. *An insight into iPhone user behaviour within the App Store*. Dec. 2012. URL: `http://www.surikate.com/en/etudes.html`.

[40] Andrew Sweeting. "The strategic timing incentives of commercial radio stations: An empirical analysis using multiple equilibria". In: *The RAND Journal of Economics* 40.4 (2009), pp. 710–742.

[41] Elie Tamer. "Incomplete simultaneous discrete response model with multiple equilibria". In: *The Review of Economic Studies* 70.1 (2003), pp. 147–165.

[42] Gabriel Y Weintraub, C Lanier Benkard, and Benjamin Van Roy. "Markov perfect industry dynamics with many firms". In: *Econometrica* 76.6 (2008), pp. 1375–1411.

[43] Pai-Ling Yin, Jason P Davis, and Yulia Muzyrya. "Entrepreneurial Innovation: Killer Apps in the iPhone Ecosystem". In: *The American Economic Review* 104.5 (2014), pp. 255–259.