The ecology of Web browser

Sela Ferdman

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE MASTER DEGREE

University of Haifa

Faculty of Social Sciences

Department of Computer Sciences

March, 2015

June 14, 2015

v1.3

The ecology of Web browser

By: Sela Ferdman

Supervised By: Dr. Einat Minkov and Dr. Ron Bekkerman

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE MASTER DEGREE

University of Haifa

Faculty of Social Sciences

Department of Computer Sciences

March, 2015

Approved by: ________________________________ Date:________

(supervisor)

Approved by: ________________________________ Date:________

(Chairman of M.Sc Committee)

Abstract

TBA

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Web browsers, a software application for retrieving and presenting resources on the World Wide Web, have become a major part of our computer ecosystem. In fact, some recent operating systems are based entirely on browsers (e.g., ChromeOS by Google[1]). Browser extensions (also called add-ons) are computer programs, which allow the user to customize a browser to meet his or her needs. These extensions (as the name suggests) extend, improve and personalize browser capabilities. An extension was developed, for example, that provides visually impaired users with access to the content of bar charts on the Web Elzer et al. (2007). Another example browser add-on addresses security concerns, transparently producing a different password for each Website, and by that defending the user against password phishing and other attacks (Ross et al., 2005). A great number of extensions are installed by users on a daily basis nowadays. It was reported that more than 750 million (non-unique) extensions were downloaded and installed by users of Google Chrome alone as of June 2012[2]. Toolbars are considered to be a particular kind of a browser extension. A browser toolbar is a GUI widget, which typically resides in the upper part of the browser's window. All major Web browsers support toolbar development as means of extending the browser's GUI and functionality. While browser extensions in general, and toolbars in particular, may be installed following an explicit request by the user, these add-ons are often 'silently' installed by a third party as the user downloads some other program from the Web, or add-ons may be included in a 'software bundle'. There are several reasons why software

[1] `http://www.chromium.org/chromium-os`

[2] `http://www.medianama.com/2012/06/223-the-lowdown-google-io-2012-day-2-310m-chrome-users-425m-g`

companies are interested in having specific toolbars installed on the user's machine. First, a special characteristic of toolbars is that they are often used for collecting information about the browsing history of the user (e.g., Yahoo! Toolbar (**?**)). In addition, it is often the case that once the toolbars are installed, all user searches are directed to some dedicated search portal (e.g. MyWebSearch.com). The company which owns the Website (and typically also the toolbar) typically receives payments from ad providers (primary ad providers today are Google and Yahoo!), when user clicks on ads presented. In fact, this model is used extensively nowadays to generate revenue by software companies, which otherwise distribute freeware products (Leontiadis et al., 2012). For example, 45% of AVG Technologies sales were due to its browser toolbar[3]. It was recently estimated that Google, the biggest Web advertising firm, may lose $1.3 billion in 2013 revenue due to its toolbar policy changes, which might cause some of the companies to shift to its competitors[45] . Importantly, a user serves as a source of income to the installing party as long as he or she actively uses the toolbar, where users may choose to remove the software from their system anytime. The ability to estimate the period of toolbar survival is therefore critical in considering a business model for companies that distribute their software as freeware, counting on revenue generated due to installed toolbars. For example, when a user installs Babylon's translation software , he is offered to install also AVG toolbar, where AVG pays Babylon for these installations. If AVG could estimate whether a specific user would keep its toolbar, it could apply a differential payment model to Babylon according to estimated 'user value'. In the proposed research, we will consider a large-scale authentic data that tracks browser extensions installed at users' machines on a daily basis. In addition to daily 'reporting', events of adding new browser toolbars are explicitly recorded. Removal of toolbars can be inferred. We are interested in exploring trends in this large-scale data, and in answering specific questions, such as 'can one predict if a particular add-on survives on the user's machine more than K days'. Providing a prediction model of good quality in response to this question is highly valuable to the industry, as discussed above. We will explore machine learning techniques to create prediction models of interest. While many academic studies have used information about user environment towards the creation of user

[3] `http://seekingalpha.com/article/1147451-avg-feb-1st-google-policy-updates-threaten-avg-s-growt`

[4] `http://finance.yahoo.com/news/google-may-miss-2013-revenue-113926474.html`

[5] `https://support.google.com/adwordspolicy/answer/50423?hl=en`

profiles and personalization of software services, there is no previous research, to the best of our knowledge, which used information about user browser extensions to similarly model user behavior. We will examine the hypothesis by which the existence of an extension on a users computer reflects something about the user's application download history, as well as about his or her general preferences. Specifically, we will evaluate whether clustering users using the underlying data allows one to improve prediction of future behavior. The main research questions and goals are stated in Section 3. An overview of related literature is given below. In addition to previous works on user profiling, we discuss recent learning methods used in large-scale recommendation systems, relating in detail to clustering techniques, which are used to alleviate data sparsity issues. Finally, descriptions of the prediction models that we plan to use, and an account of the special characteristics and challenges involved with the dataset.

1.1 Main Contributions

The main contributions of this work are the following:

-

1.2 Roadmap

Chapter 2

Related Research

2.1 User Profiling

User profiles characterize users according to their personal preferences and skills, as reflected by raw material gathered from their interaction histories with the system (Koch and Wirsing, 2001),(Gauch et al., 2007). According to (Gauch et al., 2007), User profiling is typically either knowledge-based or behavior-based. Knowledge-based approaches engineer static models of users and dynamically match users to the closest model. In this paradigm, questionnaires and interviews are often employed to obtain relevant information about the user. Behavior-based approaches model user behavior directly, typically using machine-learning techniques, to discover useful patterns from behavioral data. In particular, various aspects of user behavior at the desktop are reflected by browser usage (Benevenuto et al., 2009),(Bilenko and Richardson, 2011). (Lieberman et al., 1995) and (Joachims et al., 1997) have inferred user preferences given user-browsing behavior. (Sugiyama et al., 2004) constructed user profiles based on pure browsing history, comparing their model with collaborative filtering. (Lu and Boutilier, 2011) developed a set of algorithms to support efficient learning of user preferences, where the observed data consists of pairwise comparisons of items. A general overview of user modeling techniques can be found in (Leontiadis et al., 2012). (Sebastiani, 2002) (Sebastiani, 2003) provides a survey of current machine learning approaches used for user profiling. Various learning approaches may be applied, including Bayesian classifiers clustering, decision trees and artificial neural networks (Pazzani and Billsus, 1997), multi-class classification Bauer et al. (2014), and so forth. While browsing behavior has been studied in the context of user profiling, in the proposed research we are interested in using of a different type of 'behavioral logs'. In particular, we believe that the logs detailing the add-ons installed on the user's machine over time can be used as another source of meaningful information about the user's preferences. To the best of our knowledge, this type of user history data has not been studied before. The user profiling approach we are interested in is clearly a behavior based one. Accordingly, we will seek to derive patterns from these data logs that are meaningful using machine learning methods.

2.2 Recommender Systems

Recommendation (or, Recommender) system applications (Resnick and Varian, 1997) consider two classes of entities, usually referred to as users and items. Users have preferences for certain items, where recommender systems are aimed at teasing out these preferences out of historical data. The underlying data corresponds to a utility matrix, assigning values to user-item pairs that represent what is known about the degree of preference of that user for that item. In order to construct this matrix, information need to be first collected on the preferences of the users for a set of items (e.g., movies, songs, browser extensions). Such information can be acquired explicitly (typically, by collecting users ratings) or implicitly (typically, by monitoring users behavior, such as songs heard, applications/add-ons downloaded ,web sites visited and books read). Recommender systems produce a list of predictions following one of two main paradigms - through content-based modeling or collaborative filtering. Content-based (CB) systems examine properties of the items and determine inter-item similarity based on their properties. Past predictions are then projected onto similar items. For example, if a Netflix user has watched many sci-fi films, then the system would recommend other movies of the sci-fi genre in the database. In a content-based system, one must therefore construct for each item a profile, which is a record representing important characteristics of that item. Items profiles can be described by vector of Boolean, as well as multinomial or continuous values. It is possible that item features be obtained automatically from tags (Golder and Huberman, 2006). One of the earliest attempts to tag massive amounts of data was the site del.icio.us, later bought by Yahoo!, which invited users to tag Web pages. Notably, browser extensions are often assigned tags by its developers, tags like Sports, Weather Forecasts, Games and Entertainment. Collaborative-Filtering (CF) systems (term coined by (Goldberg et al., 1992)) focus on the relationship between users and items. They use the known preferences of a group of users to make recommendations or predictions for other users, based on their similarity to the identified user groups. That is, past actions of a group of users are tracked in order to make predictions for individual members of the group. The biggest advantage of collaborative-filtering systems over content-based systems is that explicit content description is not required. Two main types of algorithms for collaborative filtering have been researched: memory-based and model-based. Memory-based CF approaches assess inter-user similarity based on common items in the user-item matrix. These approaches are often deployed into commercial systems (e.g. Amazon) because they are easy-to-implement and highly effective (Vapnik, 1999). However, there are several limitations of the memory-based CF techniques, such as the fact that similarity assessments are unreliable when data is sparse and the common items are few. In order to overcome these shortcomings, model-based CF approaches have been developed. Model-based techniques use the collected data to estimate, or learn, a model to make predictions (Breese et al., 1998). Well-known model-based CF techniques include Bayesian belief nets (BNs) (Breese et al., 1998), as well as use of clustering (Ungar and Foster, 1998),(Zhu and Gong, 2009) and latent semantic analysis (Hofmann, 2004). While memory-based techniques alleviate data sparsity issues, they involve an additional expense of model-building. Hybrid techniques, such as the content-boosted CF algorithm [28] and the approach by (Gong et al., 2009), combine collaborative filtering and content-based techniques, hoping to avoid the limitations of either approach and thereby improve recommendation performance. (Breese et al., 1998) proposed a probabilistic approach to collaborative filtering, which computes the probability that user u give a particular rating to item s given the users ratings of the previously rated items. Two alternative probabilistic models were considered: clustering models and Bayesian networks. In the first approach, like-minded users are clustered into classes; given the users class membership, the user ratings are assumed to be independent, i.e., the model structure is that of a nave Bayesian model. The second model represents each item as a node in a Bayesian network, where the states of each node correspond to the possible rating values. Both the structure of the network and the conditional probabilities were learned from the data. Another interesting methodology of adapting the user profile in a dynamic and automatic way is presented in (Marin et al., 2013).

2.3 Similarity in graphs using Personalized PageRank

Personalized PageRank is an extension of the famous PageRank algorithm Page et al. (1999), both of which are based on a random surfer model. To understand Personalized PageRank, we first review the original PageRank briefly. A random surfer starts at any node on the graph. At each step, with a probability of 1α the surfer moves to a neighboring node randomly, and with a probability of α he teleports to a random node on the graph. This process is repeated until the walk converges to a steady-state. The stationary probability of the surfer at each node is taken as the score of the node. However, this form of score is purely based on the static link structure, indicating the overall popularity of each node on the graph, without tailoring to a specific query node.

In contrast, Personalized PageRank enables query-sensitive ranking, in the sense that we can specify a query node to obtain a personalized ranking accordingly. It is based on the same random surfer model of the original PageRank, except when the surfer teleports, he always prefers the query node q. Specifically, at each step, with probability α the surfer teleports to q instead of a random node, thus visiting the neighborhood of q more frequently. Thus, the stationary distribution, called a Personalized PageRank Vector (PPV), is biased towards q and its neighborhood, which can be interpreted as a popularity or relevance metric specific to q. More generally, a query q can comprise multiple nodes on the graph, such that in the teleportation the surfer can jump to any node in q. Fortunately, the computation for a multi-node query is no more difficult than for a single-node query due to the *Linearity Theorem* (Jeh and Widom, 2003), as the PPV a multi-node query is a simple linear combination of the individual PPV each node in the query.

In our case, we think that this biased version of PageRank could be applied as a recommender system for recommending a user addon to specific user given its addons ecosystem.

2.4 Clustering

A known issue of Recommender Systems is the ‘cold-start’ problem, where little historical data is available for new users, or items. Clustering is often employed to alleviate the cold-start problem. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters (Han et al., 2006). When applied to the domain of Web browsing history, clustering can group together a set of Web pages with similar contents, and users with similar navigation behavior or navigation sessions. Ungar and Foster (Ungar and Foster, 1998) clustered users and items separately using variations of k-means and Gibbs sampling (Geman and Geman, 1984), grouping users based on the items they rated and items based on the users that rated them. Using to their model, users were re-clustered based on the number of items they rated, and items were similarly re-clustered. Each user was assigned to a class with a degree of membership proportional to the similarity between the user and the mean of the class. Their recommendation performance on synthetic data was good, but not as good on real data. A flexible mixture model (FMM) extends existing clustering algorithms for CF by clustering both users and items at the same time, allowing each user and item to be in multiple clusters and modeling the clusters of users and items separately (Si and Jin, 2003). Experimental results show that the FMM algorithm has better accuracy than the Pearson correlation-based CF algorithm and aspect model (Hofmann, 2004). Clustering for both users and items is often referred to as multi-way clustering (Bekkerman et al., 2005) or bi-clustering (Zhu and Gong, 2009). Bekkerman at (Bekkerman et al., 2005) have shown that clustering users and items simultaneously provides better results (when the users and the features are inter-dependent). In our focus domain, for example, a user may use one set of browser add-ons for work and another for entrainment. Multi-Way Distributional Clustering (MDC) is an efficient implementation of Combinatorial Markov Random Fields (Comraf), a novel type of undirected graphical models due to (Bekkerman and Sahami, 2006)]. The model is grounded on three principles: (i) Exploiting multi-modality of the data, i.e. the fact that the data can be viewed from different angles or perspectives. For example, consider a dataset of documents that should be clustered into topics. Rather than represent full documents, a different modality of this data would be sets of words that are contained in the documents. Additional modalities would be a set of author names, a set of documents titles etc. Comraf allows one to simultaneously cluster a number of data modalities, having each one potentially improves the quality of all the others. (ii) Most existing clustering methods are based on explicit definition of a pairwise distance measure between data instances. Such a measure is usually chosen heuristically, and can in some cases be inappropriate for particular tasks. MDC refrains from such an explicit definition and instead optimizes a global objective function over the entire data. (iii) Most existing clustering algorithms are either agglomerative (start with small clusters and merge them to compose larger clusters), divisive (start with one large cluster and split it to obtain smaller clusters), or flat, such as k-means (start with k clusters and rearrange data within these clusters). (Bekkerman et al., 2005) propose to combine all the three approaches together, while choosing the most appropriate one for each data modality. A detailed description of the Comraf model is given in (Bekkerman and Sahami, 2006), and the underlying clustering algorithm (MDC) is described in (Bekkerman et al., 2005). Some modalities may be not clustered by Comraf, as explained in (Bekkerman and Jeon, 2007). In the proposed research, the underlying data is highly sparse. We will therefore explore clustering of items (add-ons) in order to get denser historical records for each user. In addition, we may cluster users, in order to associate new users who have little known history with relevant user groups. As described above, concurrently clustering data along both dimensions may be desired.

Chapter 3

Problem definition

This is essentially equivalent to the research goals.

We have decided to investigate the browser ecosystem from three points of view:

1. We simulated an artificial extraction of an addon from its habitat, and we see that the rest of the system works to bring it back.

2. We see that some addon ”species” (i.e. companies) compose a symbiosis, while others clash with each other (which results in extinction of one of the species).

3. We observe how the browser ecosystem evolves over time.

Given a successful resolution of first problem, the system can infer what addon is ”missing” in current user addons ecosystem and thus predict/recommend an additional addon.

As for the second problem we were investigating, these days it becomes more and more relevant. See, for example, recent discovery about SuperFish security breach is a good example about two companies having a major symbiosis (SuperFish and Komodia) and SuperFish and all major anti-viruses addons is a good example of a clash between two companies.

It is also interesting to see how the addons in browser ecosystem evolve day by day.

Chapter 4

Method

In this chapter we will present the algorithms we implemented for this thesis.

4.1 Baseline algorithms

We chose to use two algorithms to test our algorithms, a popularity baseline and an item similarity baseline. The popularity baseline is a simple recommendation technique and the reason that we chose this method was its speed and simplicity. This allowed us to iterate quickly in the early stages of our research. The item similarity baseline is a more complicated algorithm and was used as a more realistic test of the performance the algorithms. Because the largest datasets we worked with contained hundred of thousands of users and tens of millions of items, we had to make sure that the baseline algorithms were implemented in a high performance, parallel or distributed environment. We chose to use igraph (Csardi and Nepusz, 2006), a high-performance, parallel machine learning framework. This allowed us to focus our efforts on the development of our algorithms and not the baseline algorithms.

4.1.1 Popularity

The popularity algorithm is one of the simplest one can employ for recommending items, as it simply recommends the most popular items in the dataset to all the users. The popularity of the items is determined by the total number of user for each addon in the dataset, and all the users are presented with the same recommendation list, where the items are ranked according to their popularity. Despite its simplicity, the popularity algorithm can perform reasonably well. Another advantage for this algorithm is lack of parameters and the fact that due to its simplicity we are able to make recommendations for datasets containing hundreds of thousands of users and tens of millions of items very fast. Both of these factors contributed in making experimentation much easier, an important factor for a baseline algorithm. This benefit however comes at a cost for the quality of the recommendations. Since the algorithm performs no personalization in the recommendations made, all users are presented with the same recommendation list.

4.1.2 Prediction based on PageRank scores

For each addon we have calculated its PageRank score by running the regular PageRank algorithm over the whole user addon graph. Now, like in section 4.2, when we are predicting the missing addon, we are looking at the addons ranking list (the ranking is according to PageRank scores) and check the Recall@X.

4.2 Personalized PageRank addons analysis

The PageRank that is described in (Page et al., 1999) gives a universal score for the nodes in a graph. It is however possible to change the calculations so that the results will reflect someones personal preferences. In the PageRank formula the teleportation vector $\vec{u}$ has a great influence on determining the outcome. The biasing of the PageRank by this vector is first suggested in (Brin et al., 1998), and thereafter explored many times ((Haveliwala, 2002),(Haveliwala, 2003),(Haveliwala et al., 2003)).In this thesis the goal of biasing the PageRank with the teleportation vector is to personalize the user addons of choice results. PageRank exists for any graph and not just the web graph. PageRank on a graph produces an importance score for each node, and this places PageRank amongst a class of network analysis techniques (Brandes and Erlebach, 2005) known as centrality measures or indices (Koschützki et al., 2005). Instead of looking at a randomsurfer on the web, the non-web PageRank models a random walk on the graph. The behavior of the walk is the same as the random surfer: with probability α the walk continues along an edge of the graph and with probability 1-α the walk jumps to a random node in the graph. Personalized PageRank is an important extension of PageRank model, in which the surfer does not randomly restart browsing anywhere on the web after choosing not to follow a link. Rather, the surfer in this new model restarts at one of only a few pages. If the pages relate to one person, then the resulting PageRank vector is called a personalized PageRank vector. If the pages are topically related, then the vector may be called a topic-specific PageRank vector. In (Freschi, 2007), the authors used the Personalized PageRank model, which they call ProteinRank, to predict protein functions. The Personalized PageRank model seems to perfectly apply to our set of problems as well. We are dealing with users and looking for their personal preferences in browsers extensions, we are also interested in Company based addons propagation in users network. All our dataset was transformed to undirected graph, where user nodes are connected to addon nodes according to their relationship in the original dataset, e.g. each user node has an edge connected to addon node if this user has this addon installed in its ecosystem. Since our goal is to determine/predict a missing addon in user ecosystem, the seed nodes for the personalization vector would be the specific user addons, our hypothesis is that a higher the PageRank score suggests a higher relevance of an addon to the specific user. We would run multiple experiments to show the correctness of our hypothesis.

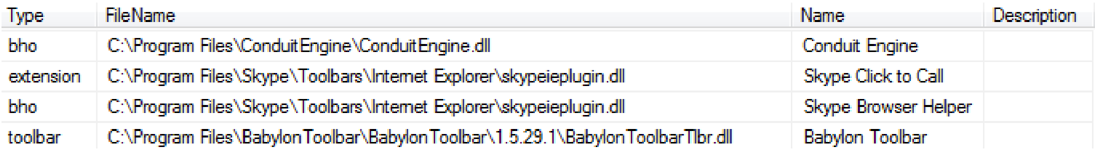
Chapter 5

Datasets

5.1 Data

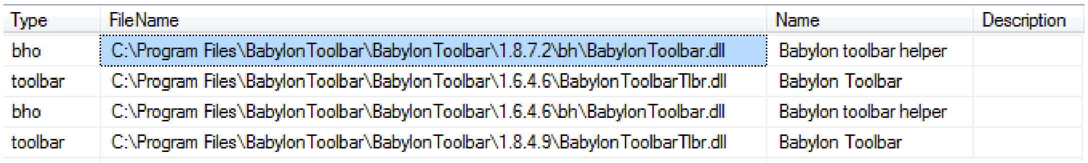
In our experiments we have used large-scale authentic data.The data was collected from various anonymous users that have agreed to share it for improving the product. The users are from all around the globe. This data consists of users, detailing the browser add-ons installed on every user's machine per day, over time. The addons in the database are from three main browsers [citation], Internet Explorer, Mozilla Firefox and Google Chrome. This is a common scenario that user has more than one browser. Since Internet Explorer usually comes pre-installed, many users install an additional browser (typically Chrome or Firefox). The data is stored in relational database in cloud at Amazon RDS , including over 1.5 billion records to date. The data was collected over period of two months from 1/8/2013 to 1/10/2013. This a real data collected from users that have agreed to anonymously send this information. The data was very noisy and special treatment to clean this data was needed.The data in SQL is split into several tables. Following is a summary of information available in this dataset about a specific user:

- A user record specifies his origin: IP, ISP, Country, City and Operating System versio (i.e. Windows XP, Vista, etc.).
- The user's homepage and the default search Website defined per browser are given.
- The installed add-ons are listed for every user. As can be seen at 5.1, the type of add-on is



Type	FileName	Name	Description
bho	C:\Program Files\ConduitEngine\ConduitEngine.dll	Conduit Engine	
extension	C:\Program Files\Skype\Toolbars\Internet Explorer\skypeieplugin.dll	Skype Click to Call	
bho	C:\Program Files\Skype\Toolbars\Internet Explorer\skypeieplugin.dll	Skype Browser Helper	
toolbar	C:\Program Files\BabylonToolbar\BabylonToolbar\1.5.29.1\BabylonToolbarTlbr.dll	Babylon Toolbar	

Figure 5.1: An example SQL table



Type	FileName	Name	Description
bho	C:\Program Files\BabylonToolbar\BabylonToolbar\1.8.7.2\bh\BabylonToolbar.dll	Babylon toolbar helper	
toolbar	C:\Program Files\BabylonToolbar\BabylonToolbar\1.6.4.6\BabylonToolbarTlbr.dll	Babylon Toolbar	
bho	C:\Program Files\BabylonToolbar\BabylonToolbar\1.6.4.6\bh\BabylonToolbar.dll	Babylon toolbar helper	
toolbar	C:\Program Files\BabylonToolbar\BabylonToolbar\1.8.4.9\BabylonToolbarTlbr.dll	Babylon Toolbar	

Figure 5.2: An example SQL table with different versions for same addons

specified (extension, bho, toolbar, etc.), as well as the name of the program, the full path where it resides on the user's machine and the description of the addon; for example, from DB data at figure 5.1 by examining the FileName,Name or Description columns, we can infer that the user has Conduit, Skype and Babylon addons installed.

- The add-ons lists are updated on a daily basis. Thus, we can infer add-on changes (installation and removal) over time. But, since we are not monitoring user actions or other programs actions, we cannot recall whether an addon was removed by the user or by hostile/protecting program. So it is not possible to determine which party initiated either of these status changes.

- One of the issues we had to overcome was identifying similar addons also they did not look exactly the same by looking at their attributes. At 5.2 you can see four addons that should be considered the same but look different - There are often small lexical differences between add-ons with respect to the path, add-on name and also description, in particular, in version number, but essentially it is the same addon.

We are were looking at the data that was collected over two months and included:

- 907,844 users

- 456,458 addons

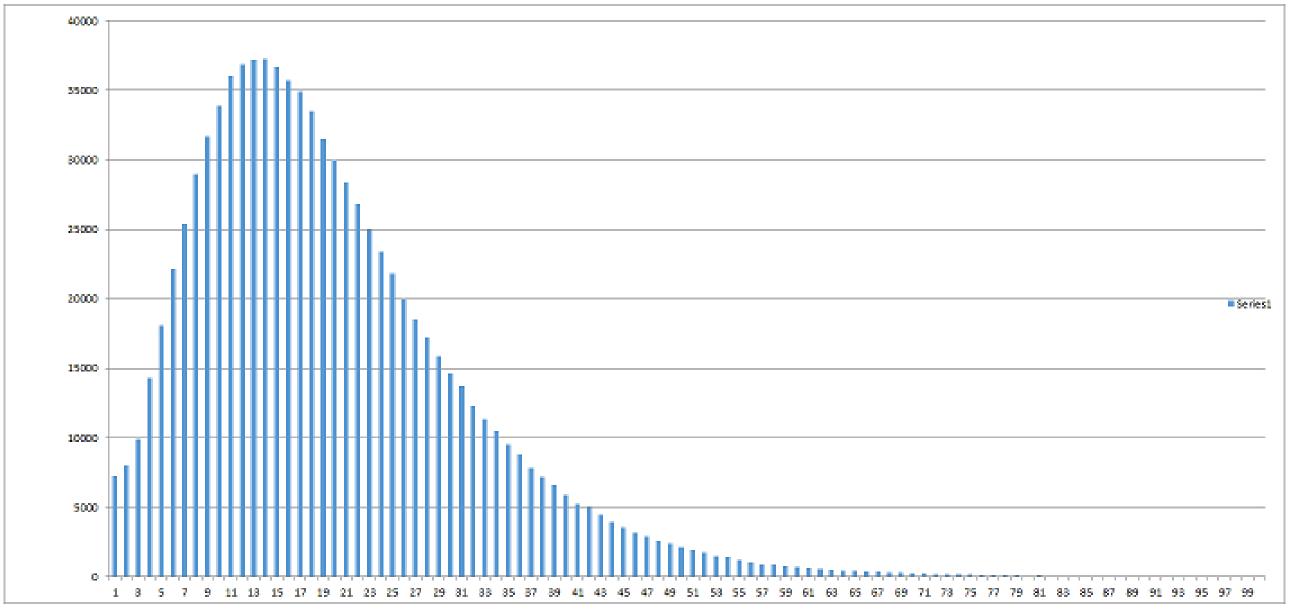


Figure 5.3: Number of addons per user distribution

We have observed that most users have between 9 and 21 addons installed on their PCs.

5.2 Arranging the data

Since the data origins in SQL tables, we needed to turn it to some format that can be accepted by our algorithms. We have started by creating a document which had in every line a user id and all its addons attributes. This has created a document with 17942715 lines and 3.5 GB size. We than broke the addons into collection of words, so that each user is actually represented as it ID and a collection of words from its addons that describes him.

For cleaning the data we first meant to remove all non-alphanumeric characters but it turned to be not good enough. On the one hand dashes and underscore appear to be an integral part of word strings, while on the other hand we had to perform addition clean up before we could use the data.

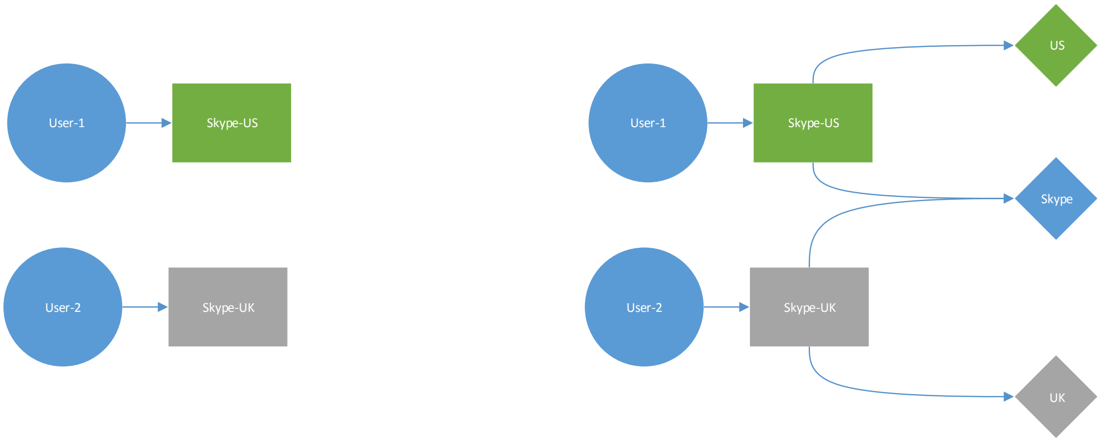
- Removing some non-ASCII characters like in ”chrome/firefox, since the data is sent over network encrypted and decrypted on the server side before storing to SQL tables, some metamorphoses could happen to the text and we need to clean this up, specially because many of the algorithms implementation do not support non-ASCII characters.

- Fixing Windows shortcuts to full paths, Windows sometimes adds 1 as path suffix, so actually VIDEOD 1 and VIDEOD 2 and VIDEOD are the same path to addon.
- We have also converted all strings to lower case since, addon wise, as long as the strings are the same it is the same addon.

5.3 Graph representation

We have decided to transform our data of users and it addons into a relational graph. Each user is defined by a set of its addons, so each user is a node which is connected to its addons with bi-directional edge. Each user has a ’relation’ with other users if they share similar add-ons. There exists another layer of nodes - each addon is to be seen as bag of terms. Each term is a node that connected to addons that contain it. This came to emphasize that addons that are not exactly still might be of same origin. For example, addon that is called Skype-US is most probably well related to addon Skype-UK. Our hypothesis was that,if we did not add the layer of term nodes, see 5.4a, than two users having each one addon would have nothing in common, while with the terms nodes the user will be connected by the ”Skype” term node, see 5.4b.

On the other hand, many addons have a file-system path in their name, such as C://Program Files (x86)//..., thus we have hypothesized that connecting these addons into term nodes will create massively connected addons while they don’t actually have something in common (besides being installed on user PC). We have tested this issue in two ways. First, we have ran experiment comparing results on graph with removed addons prefix that contain a common prefixes like C://Program Files (x86) and on graph where addons remained with full paths. Second, we have ran experiment on a graph when we have removed high degree nodes (higher than 500) and on graph with all nodes, and compared the results. Removing high degree node should only result in only a small approximation error[cite



(a) Without terms layer

(b) With terms layer

Figure 5.4: Terms layer example

Purnamrita Sarkar], while improving the performance and giving equal opportunity for low-degree nodes. Another issue we had to overcome was the same addons, but with different versions. Addons with different version usually resulted in slightly different addon name, this difference could be at the name itself or in path leading to this addon. Splitting the addon nodes into terms should elegantly overcome this obstacle, since the addons will differ only slightly because of its versions.Thus users having the same addon but with different versions will still be strongly connected.

After trying different approaches to represent our data, we've chosen to represent it as bi-directional graph. We have three type of nodes:

- User node - this node represents user by his unique user id. The node is linked to addon nodes
- Addon node - this node represents an addon with all it attributes - file path, addon name and description. The node is linked to user nodes and term nodes.
- Term node - this node contain a term from addon attributes.

Graph construction description:

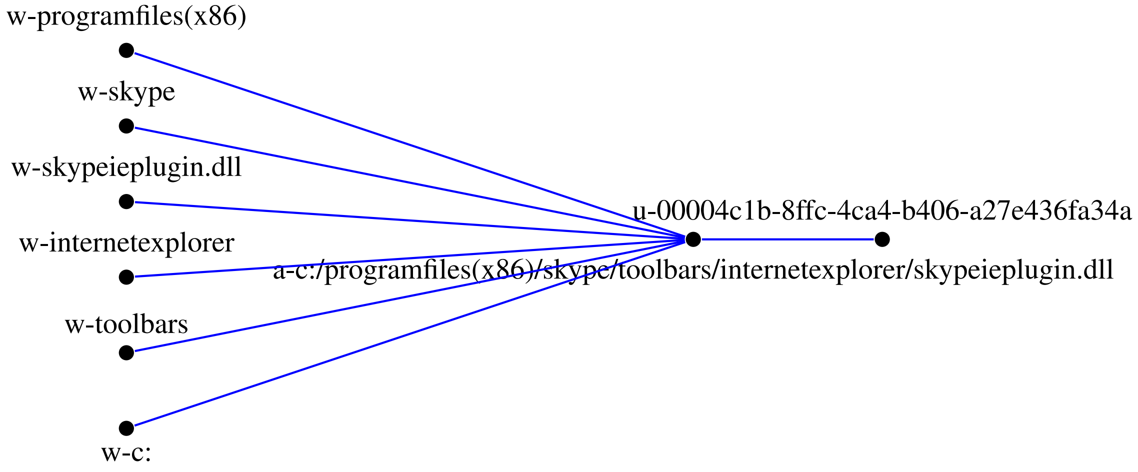

w-programfiles(x86)
w-skype
w-skypeieplugin.dll
w-internetexplorer
w-toolbars
w-c:
u-00004c1b-8ffc-4ca4-b406-a27e436fa34a
a-c:/programfiles(x86)/skype/toolbars/internetexplorer/skypeieplugin.dll

Figure 5.5: An example graph view

(a) Go over all users, for each user create a graph node. Graph node name for a user has a prefix ”u-” appended to user ID, for example u-00004c1b-8ffc-4ca4-b406-a27e436fa34a node.

(b) For each user, go over all its add-ons.

(c) For each addon create a graph node, graph node name has a suffix ”a-” and appended all addon attributes. For example ”a-c:/programfiles(x86)/skype/toolbars/internetexplorer/skypeieplugin.dll”

(d) For each addon create term nodes according to its attributes, each term node has prefix ”w-” and appedned with a term. For example the addon above would create 6 term nodes:

- ”w-c:”
- ”w-programfiles(x86)”
- ”w-skype”
- ”w-toolbars”
- ”w-internetexplorer”
- ”w-skypeieplugin.dll”

User has edges to all its addons and addon has edges to all it terms, all edges are bi-directional.

We have added additional steps for data cleanup.

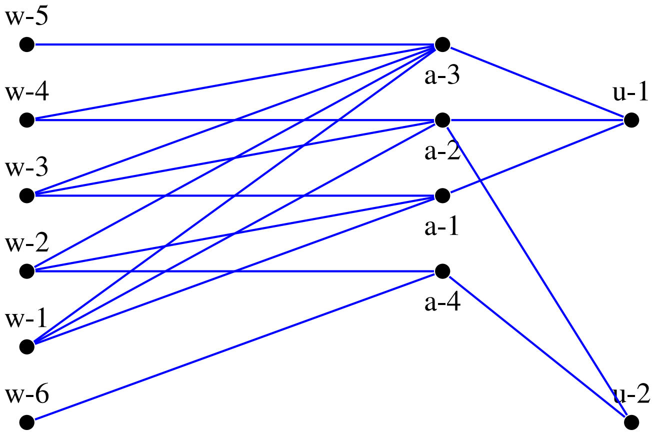


Figure 5.6: An example graph view 2

Sample graph:

In this sample graph, there are two users - u-1 and u-2.The first user has 3 addons and the second user has two add-ons. Both users share one addon - a-2, but also connected via term node w-2.

5.4 Similarity in graphs using Personalized PageRank

Personalized PageRank is an extension of the famous PageRank algorithm Page et al. (1999), both of which are based on a random surfer model. To understand Personalized PageRank, we first review the original PageRank briefly. A random surfer starts at any node on the graph. At each step, with a probability of 1α the surfer moves to a neighboring node randomly, and with a probability of α he teleports to a random node on the graph. This process is repeated until the walk converges to a steady-state. The stationary probability of the surfer at each node is taken as the score of the node. However, this form of score is purely based on the static link structure, indicating the overall popularity of each node on the graph, without tailoring to a specific query node.

In contrast, Personalized PageRank enables query-sensitive ranking, in the sense that we can specify a query node to obtain a personalized ranking accordingly. It is based on the same random surfer model of the original PageRank, except when the surfer teleports, he always prefers the query node q. Specifically, at each step, with probability α the surfer teleports to q instead of a random node, thus visiting the neighborhood of q more frequently. Thus, the stationary distribution, called a Personalized PageRank Vector (PPV), is biased towards q and its neighborhood, which can be interpreted as a popularity or relevance metric specific to q. More generally, a query q can comprise multiple nodes on the graph, such that in the teleportation the surfer can jump to any node in q. Fortunately, the computation for a multi-node query is no more difficult than for a single-node query due to the *Linearity Theorem* (Jeh and Widom, 2003), as the PPV a multi-node query is a simple linear combination of the individual PPV each node in the query.

Chapter 6

The missing addon in User ecosystem

In this chapter we present the experiment results and analyze the performance of the algorithms. Consider a user-addons network with users and addons as nodes, where users are connected to its addons and are connected between them via add-ons. Given a user in the network, how can we detect some potential addons that would be relevant for him? Taking the user addon nodes as user ecosystem and detecting which node is missing in this ecosystem would answer this question. We are using a Personalized PageRank over the a graph of users and addons (seen as global ecosystem) and taking a specific set of user addons as PPR input query, a ranking over all the other addon nodes can be leveraged for the detection of missing addon in user ecosystem.

In the above scenario, the rankings are specific to the dynamic queries, reflecting the relevance of nodes to the query node. As a well-studied graph ranking algorithm, Personalized PageRank (Page et al., 1999) is effective in calculating such query-specific relevance based on the link structure of the graph. In this paper, we use the rankings of Personalized PageRank, as a criteria for recommending an addon.

6.1 Experimental setup

Given the large scale of our graph - 1,331,814 nodes and 18,552,622 edges , and our need to execute the algorithm multiple number of times (thousands), we needed to find a fast and memory wise implementation of Personalized PageRank.

We have chosen igraph (Csardi and Nepusz, 2006) which is a library for creating and manipulating graphs.iGraph uses internally ARPACK package which is a numerical software library written in FORTRAN 77 for solving large scale eigenvalue problems (used also in Mining and Analyzing Social Networks book). ARPACK uses an implicitly restarted Arnoldi method to calculate PageRank. igraph expects as an adjacency list of numeric vertices. We have created a unique number for each node and store the mapping to an actual node name in a separate file. Since the whole graph is loaded into memory, we had to be very careful with memory management (every very small leak would cause a serious global memory leak). We also had to compile a 64bit version of igraph - since 32 bit version failed running - it could not acquire enough memory.

The dataset is constructed as following:

(a) Pick uniformly at random a user node at the graph, find all its addon nodes.

(b) Pick uniformly at random an addon node from the list of nodes found above.

(c) Remove this node - since the only connection between a user node and the addon node is the edge between them, by simply deleting this edge we are removing this addon from user profile.

(d) Now, the remained addons are the seeds of PPR vector.

The output of our algorithm is a ranked list of all add-ons. The addons are ranked according to their PageRank score.

After PPR ran we have every addon (actually every node in graph) has a pagerank score. Obviously the addons that were at the personalized - the user addons - are likely to be ranked very high. But we have removed one addon from the user set of addons, now its addon has no connection to user addons except for connections that pass by other users or terms nodes. So after we have removed this addon node from the user, it is no different than other 200 thousands addon nodes. Now, we want to see what is the rank of this addon that we have removed, the higher it would be the better our recommendation algorithm works. Since, lets say it is ranked in 8th position, not taking into account the ranking position of the original user addons, then we could recommend to user top ten scored addons and with high probability user will find a usefull addon between this ten.

6.2 Ranking measures

Recall at k - R@k Recall-at-k (Recall@k) is the fraction of the documents relevant to the query that are successfully retrieved out of the first k returned. Recall@k is therefore a highly localized evaluation criterion, and captures the quality of rankings for applications where only the first few results matter.For recommendation systems, users usually will only view several top ranked candidates. Therefore whether the correct addon exists in the top ranked candidates is important, and Recall@k is designed for measuring this.

$$Recall = \frac{number\,of\,relevant\,documents\,retrieved}{number\,of\,relevant\,documents}$$

Mean Reciprocal Rank Recall@k considers whether correct addon is in the top k results, but it does not consider the exact ranking position of the correct addon. Mean reciprocal rank (MRR) (Voorhees et al., 1999) is a well-known evaluation metric in Information Retrieval (IR) and is typically used to evaluate the position of the first correct candidate. Given the analogy between query-document search and user-item recommendation, we can define the Reciprocal Rank (RR) for a given recommendation list of a user, by measuring how early in the list (i.e. how highly ranked) is the first relevant recommended item. The MRR is the average of the RR across all the recommendation lists for individual users. MRR is a particularly important measure of recommendation quality for domains that usually provide users with only few but valuable recommendations (i.e., the less-is-more effect (Chen and Karger, 2006)), such as friends recommendation in social networks where top-X performance is important.

$$MRR = \frac{1}{Q} \times \sum_{q=1}^{Q} \frac{1}{rank(1^{st}\,relevant\,result\,of\,query\,q)}$$

Evaluation Metrics For evaluating our model we use the standard information retrieval measures. For every user we compute: (1) recall at rank k (P@k) for our task is defined as the fraction of rankings in which the correct addon is ranked in the top-k positions, (2) Our recommendation is correct when the correct addon is present in the ranked set of add-ons. The mean reciprocal rank (MRR) is the inverse of the position of the first correct addon in the ranked set of addons produced by our model For each user, we have chosen, uniformly at random, one of its addons and completely removed this addon connections to this user. Now, we have added all user addons to PPR vector and executed PPR. We ran the experiments with teleport probability α = 0.85. While we were looking for best setup for addons recommendation system, we have also observed our algorithm behavior in different kind of setups.

Why terms nodes doesn't really matter Every addon has in its neighborhood many nodes which it is connected to. Many of them are USER nodes and only very few are TERM nodes. For example, 203 neighbors with 200 are user nodes and 3 are term nodes. Given that we are running Personalized pagerank and supposedly many users are the same for the vector of current addons, than most of the weight will go via the users and terms will get very small weight to traverse. While term node might be connected to many many other addons - it almost doesn't get any weight from them since we are running Personalized PageRank and not regular PageRank most of the weight is concentrated in the personalized vector - which contains only current user addons and not terms.

Compared methods In every experiment we are comparing the baseline methods to our method.For each of the methods we report Recall@10, Recall@50, Recall@100 and MRR as described at section 6.2

Experiments parameters In every experiment we have a few parameters that we can change in order to tune and compare our method performance:

– Graph with addons that include full path or only the addon file name - as described in chapter 5, we have decided to test our method when using only addon filename as node in the graph and when using full addon path as node in the graph.

– Removing high degree nodes from the graph or not. We hypothesize that in addons/users ecosystem high degree nodes insert more noise in random walk than they contribute to correct pagerank execution. We have observed that high degree nodes are usually common nodes that will contribute more to locality instead of letting pagerank flow reach the hypothesized nodes. While removing high degree node might obviously hurt the popularity baseline performance.

- Minimal number of addons user must have for running our prediction method. Clearly, the absolute minimum is two since when

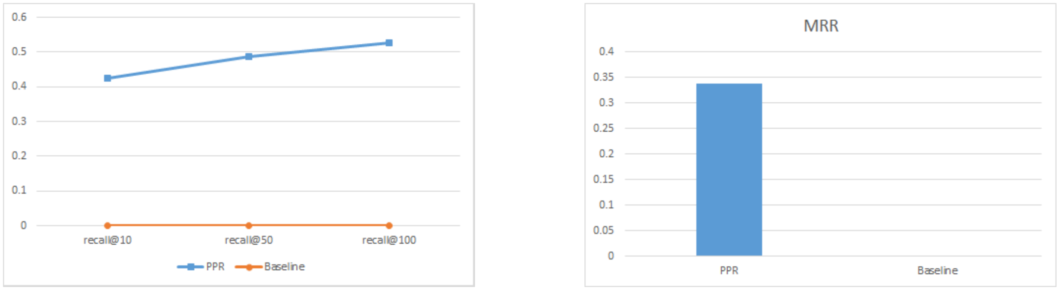
6.3 Analysis

In this thesis a great number of experiments were conducted, each validating a specific aspect of use ecosystem and trying to maximize our prediction accuracy and correctness. We have begun by removing all nodes from the graph that have a degree higher than 500. We hypothesize that in addons/users ecosystem high degree nodes insert more noise in random walk than they contribute to correct pagerank execution. We have observed that high degree nodes are usually common nodes that will contribute more to locality instead of letting pagerank flow reach the hypothesized nodes. In general, a high degree node could mean one of two things:

- If the high degree node is a term node, it means that this term is very popular between addon nodes and most probably it won't contribute for prediction of user addon.
- If the high degree node is an addon node, it means that this addon is very popular among the users and, as with term nodes, it could destruct the weight flow and also it could just recommend an already very popular item to a user.[SELA Improve]

Moreover, it is important to remember that we are experiencing with the Personalized PageRank and our hypothesis is that if the teleportaion vector in some way simulates user ecosystem, the PageRank flow would grant a missing node a high rank, mainly because of the seed nodes which belong to the same user. Thus, due to PageRank weight propagation technique, high degree node would actually reduce the influence of the original seeds.

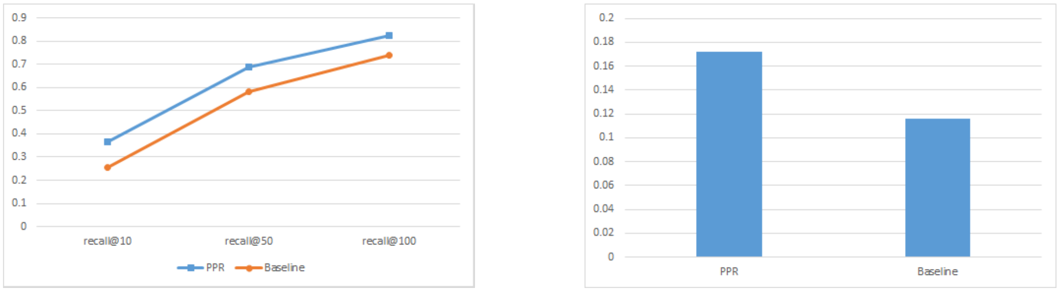
We have also chosen the users that have at least two addons installed on their computer. This restriction is very fundamental since we have no way (at least at our algorithm) to recommend an additional addon to user having only one addon, our way of knowing that we have predicted a correct addon is by removing one addon from all user addons and running PPR from the remaining addons. As with all our experiments, unless stated otherwise, we have picked uniformly at random one thousand users and conducted the experiment. As shown at Figure 6.1a, our algorithm has correctly predicted a missing addon at top-10 ranking list addons, in ∼42% cases. It is also greatly outperforms the baseline benchmark. We have also compared the Mean Reciprocal Rank (MRR) of our algorithm and the baseline as shown in Figure 6.1b.



(a) Recall@K

(b) MRR

Figure 6.1: Graph without paths without popular nodes



(a) Recall@K

(b) MRR

Figure 6.2: Graph without paths with popular nodes

Now, we've wanted to check how the high degree nodes would influence the prediction algorithm, so we've conducted the same experiment as in Figure 6.1a but without removing any node from the original graph, the results are shown at Figure 6.2. As we observe at Figure 6.4a, the Recall@10 prediction has dropped from ∼44% to ∼36%, while Recall@50 and Recall@100 have improved. One possible explanation of this behavior is that Recall@10 is less subtle to popular addons than the Recall@50 and Recall@100, thus while popular nodes can frequently enter the positions in top-

50 or top-100 ranking lists, its harder to enter the top-10 rankings where the actual real prediction lies.[SELA Improve]

High degree nodes High degree nodes in our graph typically represent a very popular addons or a term. We were considering how to treat this kind of nodes, on the one hand these nodes are part of the graph and should be treated like all other nodes. On the other hand, given the addon popularity, it is unfair to predict it - it is somewhat trivial. We've conducted all of the experiments in both ways - with and without high degree nodes.But it is our opinion that the high degree nodes should be treated in a similar way to how the stop words in lexical analysis are treated. Since the stop words has no interest for the prediction algorithms it is usually removed. We've observed, see Figure 6.3, that the nodes degree distribution follows Zipf's law distribution.

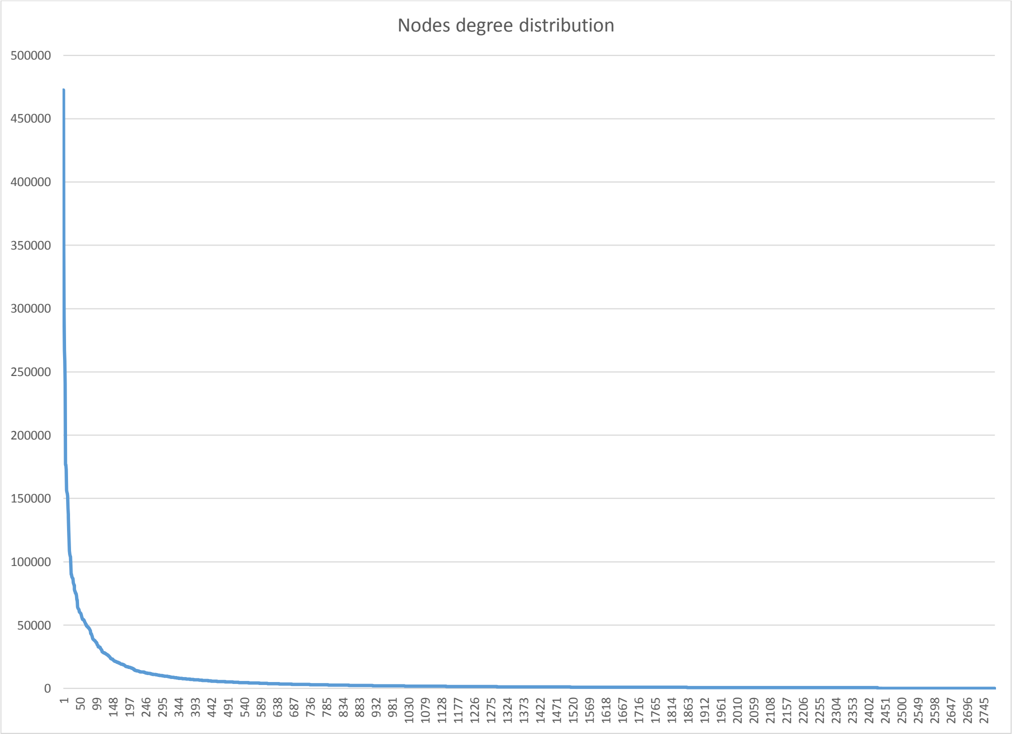


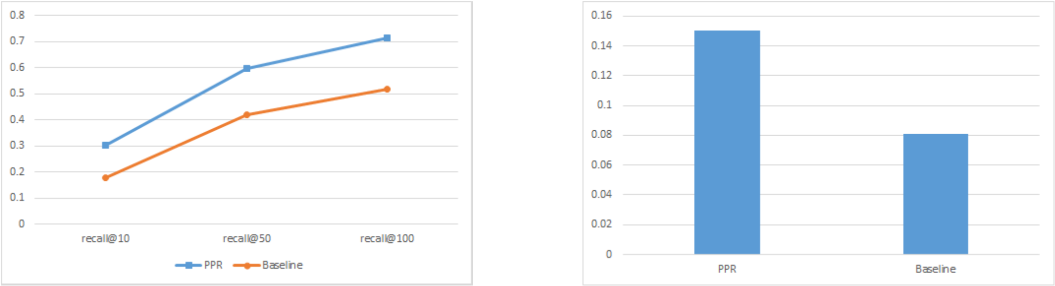
Figure 6.3: Nodes degree distribution

Given the differences between the algorithms scores, it is a good place to mention the SD and the SEM of the experiments, see Table 6.1.

Table 6.1: Standard deviation and Standard error of the Mean

	Recall@10	Recall@50	Recall@100
SD	0.001773	0.002601	0.005279
SEM	0.001024	0.001502	30.003048
MEAN	0.358904	0.661086	0.80593

As mentioned at chapter 4, the addons data as collected from the users refers to addons usually by full file system path, e.g. C:/Program Files/Skype/skype1.dll, While breaking this addon to term nodes this would create four nodes C,$ProgramFiles$, $Skype$ and $skype1$. In Figure 6.2 and Figure 6.1, we have chosen to remove the file system path from addon, hence the NoPath abbreviation, so we actually would have only one node $skype1$. We have also conducted Figure 6.2 experiment while not removing file system path from addon. The results are shown at Figure 6.4.

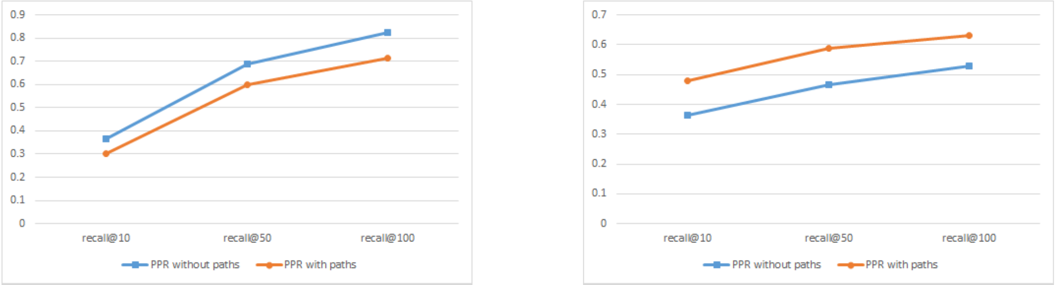


(a) Recall@K

(b) MRR

Figure 6.4: Graph with paths with popular nodes

If we compare the results of the same run, as shown in Figure 6.5a, we observe that adding addons paths hurts the performance, this can support our claim that high degree nodes, which obviously added when adding addons paths, disturb the PPR weight propagation from the real missing addons nodes. Interestingly, adding addons paths but removing high degree nodes improves the performance of our algorithm, see Figure 6.5b. These two experiments, shown in Figure 6.5, support our claim that high degree nodes should be removed to improve prediction quality. Since, once they are removed, addition information about the user, like it paths, contributes to understanding and predicting its ecosystem state.



(a) With/without addon paths with popular nodes (b) With/without addon paths without popular nodes

Figure 6.5: Comparing graph with/without addon paths

As we have mentioned earlier, in order for our prediction algorithm to run, we require that user has at least two add-ons. At Figure 6.6,we are showing the affect number of addons that user have pre-installed on the algorithm accuracy. The Figure 6.6 graph shows that, while the prediction algorithm still performs on any number of pre-installed, its accuracy decreases as the number addons that user has, growth. One possible explanation to this behavior could be - if user addon is predicted, i.e. receives its PageRank score, mainly from other users that have a similar ecosystem - that the lesser addons user has the more unique its ecosystem.

As described in chapter 4, we have added an additional layer of nodes to the graph - the term nodes. Now, we want check its efficiency on the prediction algorithm performance. As shown in Figure 6.7, adding term nodes has improved the prediction.

We have also verified that user nodes are crucial for the prediction algorithm, as shown in Figure 6.8.

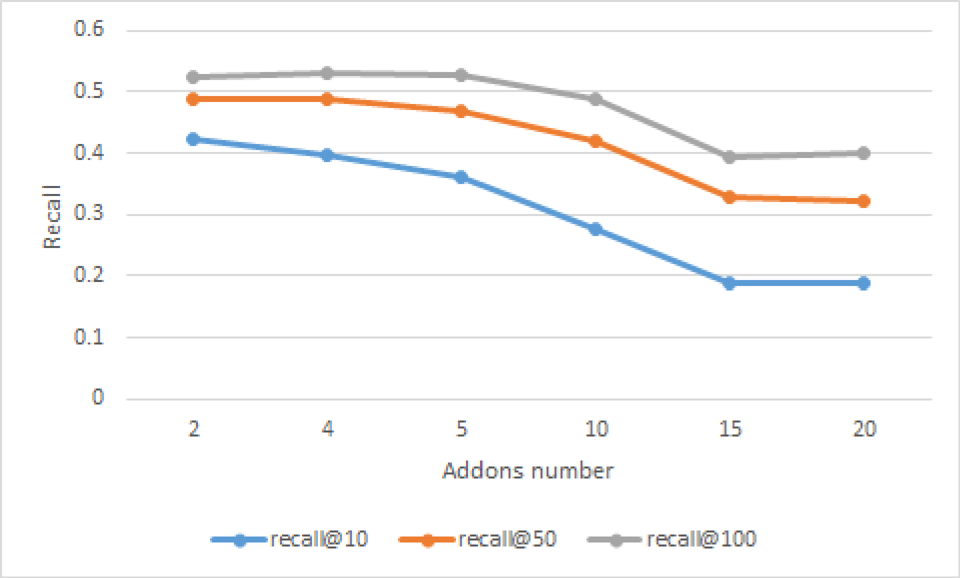
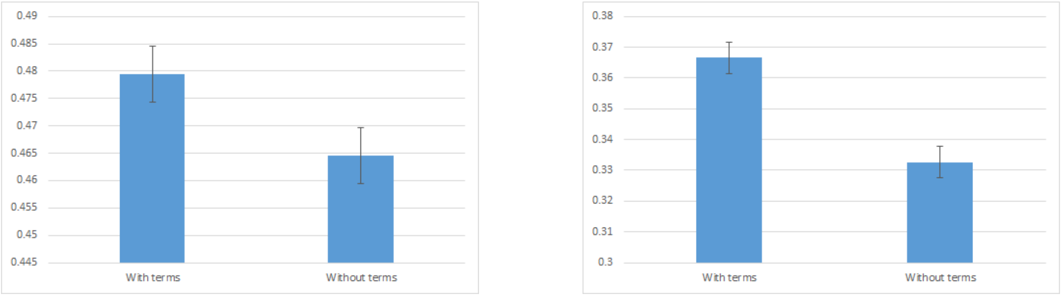


Figure 6.6: Effect of addon number on prediction accuracy



(a) With/without term nodes with paths without popular nodes

(b) With/without term nodes without paths with popular nodes

Figure 6.7: Comparing graph with/without addon paths

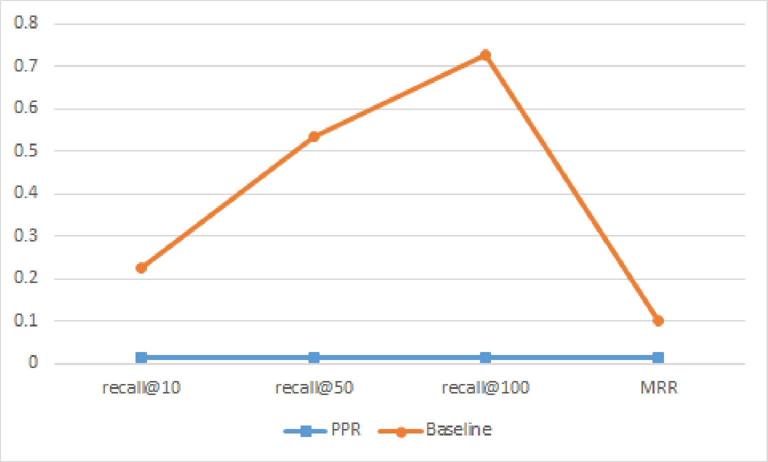


Figure 6.8: Effect of user nodes on prediction accuracy

6.4 Discussion

Our prediction algorithm performs best on full graph when highly connected nodes are removed, removing very highly connected nodes improves PPR convergence and reduces the noise in random walk. Moreover, users that are having only two addons eases the prediction, which intuitively makes sense since the PPR personalization vector contains only one node. In experiment shown in Figure 6.6, you can observe that picking users with higher number of pre-installed addons reduces the algorithm performance but still performs well. In experiment shown in Figure 6.7, you can observe the importance of term nodes, they improve connectivity between similar addons and thus improves our prediction rate. An obvious observation can be made from experiment shown in Figure 6.8., graph without user nodes are almost not possible for recommender to find a missing addon node.

Chapter 7

PPR-based analysis of addon coexistance

In this chapter, we investigate the addon coexistance phenomenon. We observe that some addons tend to coexist well with other addons, i.e. when addons of some company are installed on a user's PC, addons of another company have greater chances to get installed on the same machine. The opposive effect can be observed as well: addons of some company have less chances to exist on a machine if addons of another company are installed on this machine. This phenomenon occurs when addons of some well known companies are distributed via third parties. [RON: What's the typical process of an addon to appear in the browser? Who are those third parties?] For example, *Avira AntiVirus*, which develops addons for all browsers, treats *iMesh* addons as threats and removes them from the computer.

Companies that distribute browser addons are engaged in partnerships or in a competition with each other, such that the Web browser becomes a complex ecosystem similar in its characteristics to the biological environment of the nature. To illustrate this analogy, we say that addon distributors correspond to biological species. An instance of an addon developed by company A can be viewed as a subject of species A that coexists in the browser ecosystem with other subjects of the same or different species. Naturally, those subjects can live in symbiosys or in a conflict with each other. We observe situations when addons of some companies get installed on a machine together with addons of other companies, which is analogous to a natural migration process when subjects of different species travel together. We also observe clash situations when an addon gets installed on a machine and kicks out addons of other companies, similarly to a competition phenomenon between different species in the nature. The user (the computer owner) is also playing an important role in the addon ecosystem: some users "hunt down" and remove addons that occasionally appear in the computer's browser; other users are more tolerant – they let addons live in the browser for a long time and do not mind more addons to be added with time.

An ability to foresee such a clash or symbiosis between addon distributing companies could turn handy for addon owners. [RON: here we have room for a paragraph about how this information can be used by addon owners.] [SELA - Maybe the term symbiotic relations is not good since it definition is - Symbiotic relationships will always benefit at least one organism involved.]

7.1 Experimental setup

We have chosen companies, see Table 7.1 for the list of companies, that are known for distributing their add-ons. [Need a table with statistics on the data - RON]

In chapter 6 our algorithm's viewpoint was set of add-ons of a specific user set of add-ons - we were viewing user add-ons as its ecosystem and predicting a missing addon according to it. Now, we are aiming to investigate symbiotic relations between companies that are producing a variety of add-ons, the algorithm viewpoint is the symbiotic relationships between ecosystems created by add-ons manufactures.

Table 7.1: Companies list

Company name	**Example Product**	**Description**
ASK	Ask Toolbar	Advertisement/Search company
AVG	AVG Safe Search add-on	AntiVirus/Advertisement company
Avira	Avira Browser Safety	AntiVirus company
Babylon	Babylon Toolbar	Advertisement/Search company
Blekko	Blekko Toolbar	Advertisement/Search company
Conduit	Conduit Toolbar	Toolbar provider company
Google	Google Toolbar	Advertisement/Search company
iMesh	iMesh Search	Advertisement/Search company
Incredimail	MyStart by Incredimail	Advertisement company
Hotspot Shield	Hotspot Shield VPN	Security company
Kaspersky	Kaspersky Protection Plugin	AntiVirus company
Montiera	Montiera Toolbar	Toolbar provider company
Norton	Norton Toolbar	AntiVirus company
Zugo	Search Toolbar	Advertisement company
Softonic	Softonic Web Search	Advertisement company
SpeedBit	Video Accelerator	Software company
SweetIM	SweetIM Toolbar	Advertisement/Search company
Trend Micro	Trend Micro Toolbar	AntiVirus company
Zone Alarm	Zone Alarm Toolbar	AntiVirus company

7.2 Experimental design

Unlike in chapter 6, where all user addons were retrieved organically from the data collected from the users, now we need to identify each of the companies addons from Table 7.1. In general, we would to relate each addon to species, i.e. addons that belong to the same company said to be of the same species. For example, Kaspersky URL Advisor Firefox addon and Kaspersky Protection Chrome extension would relate to Kaspersky species. This turned out to be not a trivial task. Most of the companies from Table 7.1, does not publish anywhere list of their addons. Our first pass to identify the addons species, was to take all the data we have for each addon and look for strings that would contain a company name. That would work since many companies install their addons software under folder containing company name or write the company name in the description, see Table 7.2 for example.

Table 7.2: Company name contained in path or description example

Addon located in a folder containing company name	C:\Program Files (x86)\Kaspersky Lab \Kaspersky Internet Security 2012\avp.dll
Extension containing company name in description	Kaspersky Protection extension

After we have completed the above procedure, we've observed that there are add-ons related to the companies from Table 7.1, but do not contain company name neither in file system path nor in its description. For example, an addon $tbbaby.dll$ doesn't contain company name in any of its attributes, but is related to Babylon[1]. We propose the following procedure, see algorithm 1, for identifying this kind of addons, as the teleportation vector we provide the previously found addons that are related to the specific company.

An example for an addon that has drastically changed its score, is an addon named tbmyba.dll, the addon's $PR_g score$ was 1200, and the $PPR_g score$ turned to be 15 when using seed nodes related to Babylon, and indeed we found that this addon is related to Babylon[2].

[1] http://www.shouldiremoveit.com/Babylon-English-Toolbar-31094-program.aspx

[2] http://www.file.net/process/tbmyba.dll.html

Algorithm 1 Finding addon relation to Company

```
Inputs: Graph $G$, teleportation vector $v$, company name $n$
Run Personalized PageRank on graph $G$, with seed node from $v$.
Rank the addons according to $PPR_g score$, i.e. the Personalized PageRank scores after run
for each addon node do
    Compare $PPR_g score$ with $PR_g score$
    if $PPR_g score \gg PR_g score$ then
        This addon could be related to the company - check manually
    end if
end for
```

For every add-on that was detected as suspected by the algorithm 1, we've performed manual check to which company it belongs. This procedure has identified many *unknown* addons and correctly related to the manufacture company. This outcome presents an interesting phenomena, there is some kind of community between addon of the same species, and this community is discover by Personalized PageRank weight propagation from already known community members.

Now that we have ascribed each addon to its species, we are able to analyze the relations between the addon species in World Wide Web ecosystem.

We have executed the following algorithm 2 to collect all the necessary data of each addon and the companies.

Algorithm 2 Collecting data for each add-on

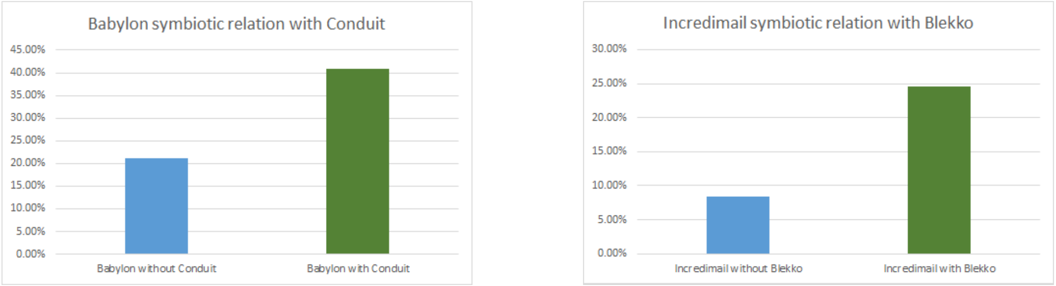
```
**Inputs:** Graph $G$, Companies $COMP$
Run PageRank on graph $G$
**for** company $c \in COMP$ **do**
    Define personalization vector $v$, with seed nodes as all addons $\in COMP$
    Execute Personalized PageRank
    Rank all addons according to the PPR score
    Store the results
**end for**
```

7.3 Analysis

7.3.1 Symbiotic Relationships

The regular PageRank score that each addon describes an ecosystem where non of the species are biased. This viewpoint allows us to assume what is the original species distribution when no other species is helping or disturbing the other. The total PageRank scores for each company is shown in Table 7.3.

While looking to relate add-ons to companies, we've observed that there are add-ons of certain companies that improve their ranking position not only when the PPR vector contains their company's nodes but also when it contains some other company nodes. This led us to hypothesize that there might be symbiotic relations between some companies. A personalized PageRank vector is the stationary distribution of a random walk that, with probability α follows a step of a random walk and with probability (1-α) jumps back to a seed node. When there are multiple seed nodes, then the choice is uniformly random. Thus, nodes close by the seed are more likely to be visited and get a higher rank. Recently, such techniques were shown to produce communities that best match communities found in real-world networks (Abrahao et al., 2012). Putting company nodes as the only seed nodes in personalized PageRank will grant a higher score to the company addons and the addons that are related in some way to this company. Our hypothesis is that nodes of the company that are having symbiotic relation with the seed nodes company will also get a higher score because of its relatedness to the symbiosis company. Eventually, we are aggregating the total PageRank score of all company addons and comparing between companies based on this score. In this way, we believe that we have found a real symbiotic relations between some companies,see Figure 7.1. In Figure 7.1b, we have indicated that there is a symbiotic relationship between Incredimail company and Blekko company, this observation has reference in reality[3]. As shown in Figure 7.2, this relationship is mutual, Blekko also benefits from this symbiosis, as well as Incredimail. The same relationship were indicated for Babylon and Conduit companies, Figure 7.1a, and again there are more than one we source indicating the same. For example, Babylon addon named tbmyba.dll is also indicated as Conduit Toolbar component[4]. Actually, Conduit is known as Toolbar manufacture for other companies.



(a) Babylon symbiotic relationship with Conduit

(b) Incredimail symbiotic relationship with Blekko

Figure 7.1: Symbiotic relationship between companies

It is known that some company addons are getting to a user machine when they are ”bundled” with add-ons of other companies. For example, the Ask Toolbars are integrated with the Java download[5]. During the installation of Java, users are presented with an option of downloading an Ask Toolbar, see Figure 7.3.

[3] http://www.im-infected.com/hijacker/blekko-search.html

[4] http://www.file.net/process/tbmyba.dll.html

[5] https://java.com/en/download/faq/ask_toolbar.xml

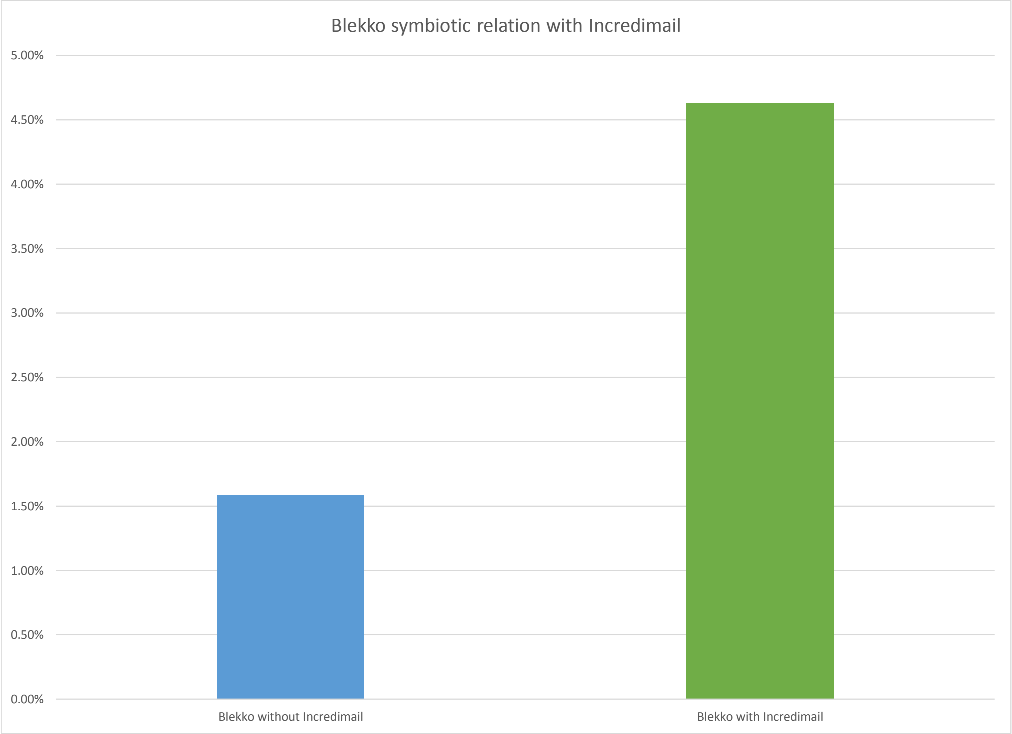
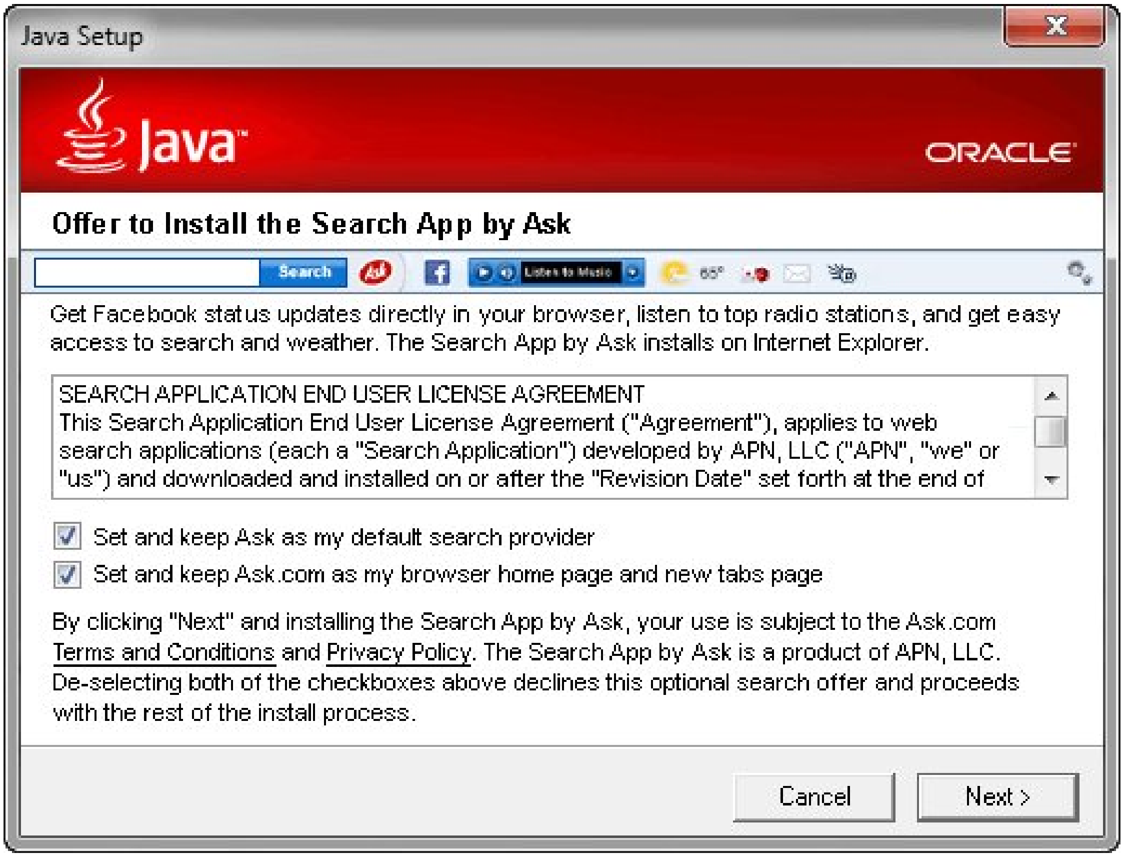


Figure 7.2: Blekko symbiotic relationship with Incredimail

Table 7.3: PageRank scores

Company name	#addons	Total PageRank score	Average PageRank score
ASK	229038	8.79139	0.97682
AVG	131337	8.50985	0.38681
Avira	10624	0.45046	0.15015
Babylon	191386	23.22184	1.10580
Blekko	14381	0.43715	0.06245
Conduit	21671	0.51959	0.07423
Google	187455	13.40115	1.11676
iMesh	23954	7.05849	0.64168
Incredimail	76198	5.79210	0.48268
Hotspot Shield	54675	2.00773	0.28682
Kaspersky	47793	2.65493	0.53099
Montiera	809	0.01752	0.01752
Norton	53826	3.22903	0.40363
Zugo	5169	0.23296	0.02912
Softonic	29410	0.82528	0.05158
SpeedBit	484676	59.02209	2.81058
SweetIM	84871	2.49227	0.49845
Trend Micro	10325	0.56126	0.05102
Zone Alarm	3597	0.10790	0.01541



Offer to Install the Search App by Ask

Get Facebook status updates directly in your browser, listen to top radio stations, and get easy access to search and weather. The Search App by Ask installs on Internet Explorer.

SEARCH APPLICATION END USER LICENSE AGREEMENT
This Search Application End User License Agreement ("Agreement"), applies to web search applications (each a "Search Application") developed by APN, LLC ("APN", "we" or "us") and downloaded and installed on or after the "Revision Date" set forth at the end of

☑ Set and keep Ask as my default search provider

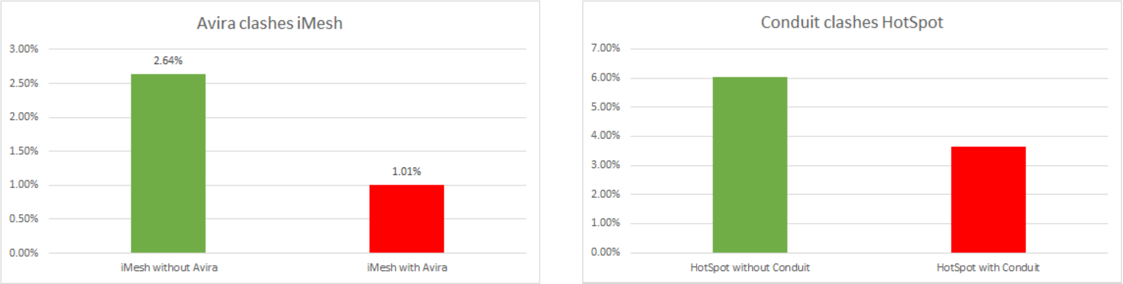
☑ Set and keep Ask.com as my browser home page and new tabs page

By clicking "Next" and installing the Search App by Ask, your use is subject to the Ask.com Terms and Conditions and Privacy Policy. The Search App by Ask is a product of APN, LLC. De-selecting both of the checkboxes above declines this optional search offer and proceeds with the rest of the install process.

Figure 7.3: Bundled ASK Toolbar installation

7.3.2 Clash Relationships

In subsection 7.3.1, we have investigated symbiotic relationships between addon species, i.e. when one species benefit from another. Here, we will look for the opposite kind of relationship, it can be referred Parasitism, i.e. where one species benefits and causes harm to the other species, but we are referring to this relation as a clash between two (or more) species. The idea that lays in the way of of identifying clash relation is very similar to the way described in subsection 7.3.1, again we are looking at the aggregated company scores, but this time we are looking for a company which score is getting lower when other company is present. In Figure 7.4, you can see two examples for this relationship. The Figure 7.4a shows that when AntiVirus company Avira is present in ecosystem it causes an iMesh company to perform worth than in regular ecosystem. The same is shown in Figure 7.4b, between Conduit and Hotspot Shield company.

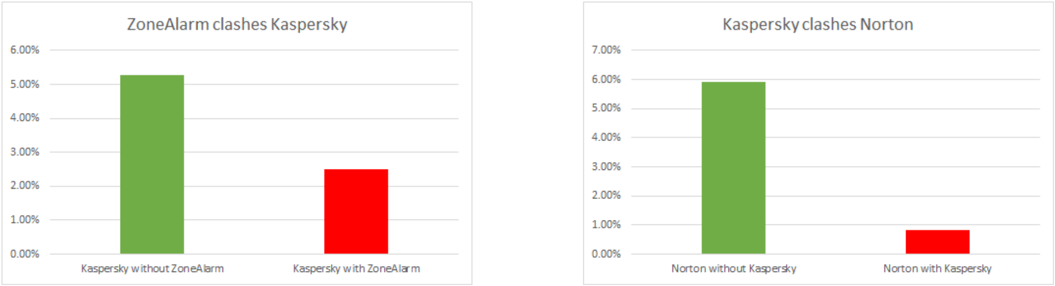


(a) Avira clash relationship with iMesh

(b) Conduit clash relationship with HotSpot

Figure 7.4: Clash relationship between companies

Interesting validation for these measures could be applied if we would know about actual clash between companies from the Table 7.3 list. Having AntiVirus companies at the list comes handy, it is known that in most case two AV software programs can not exist on the same computer. Thus, using our terminology, the AV companies should have a clash relationship. And indeed, as shown in Figure 7.5, our algorithm detects AV companies as clashing.



(a) ZoneAlarm clash relationship with Kaspersky

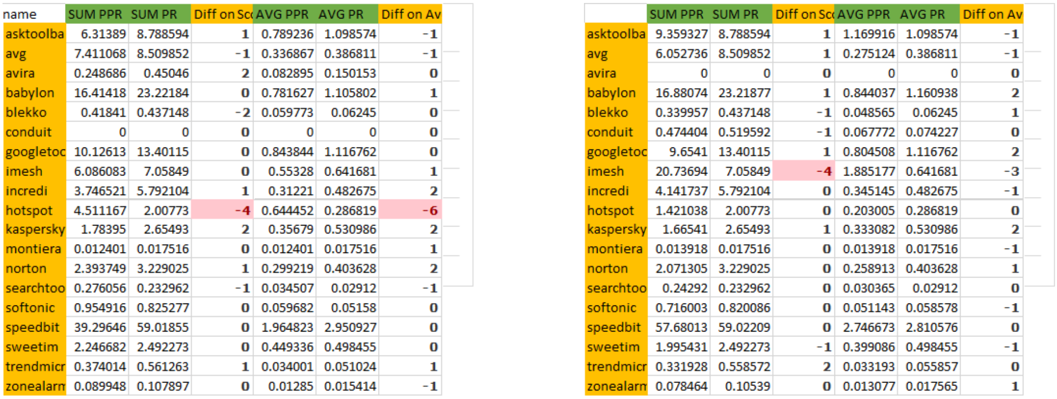
(b) Kaspersky clash relationship with Norton

Figure 7.5: Clash relationship between AntiVirus companies

7.4 Discussion

We are comparing for each company its total weighted rank in PageRank with its total weighted rank in Personalized PageRank when another company is the PPR vector. We have found that there are companies that co-exist well with other companies, there are even companies that improve greatly other companies add-ons rank. We have also observed companies that cause other companies add-ons decline.

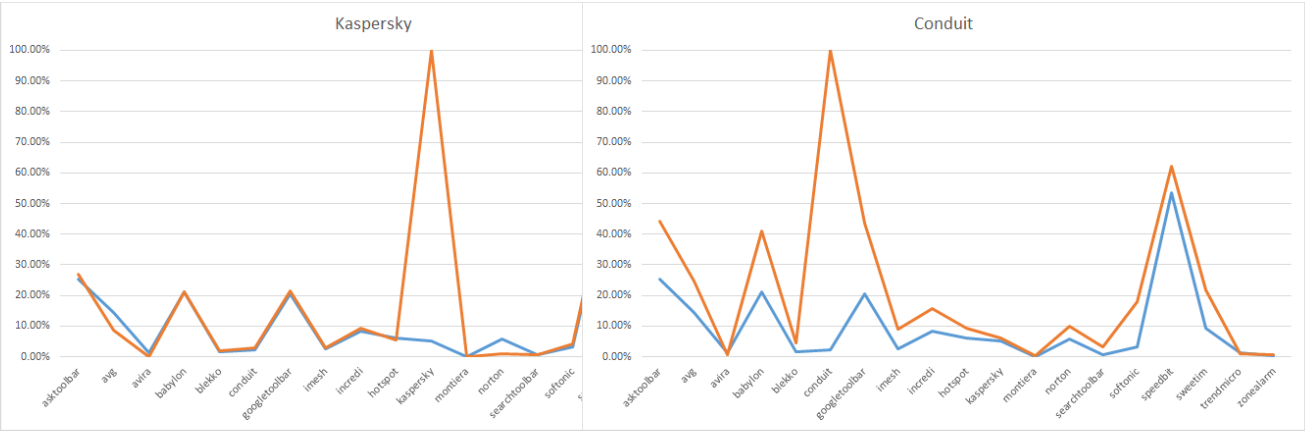
Appendix A



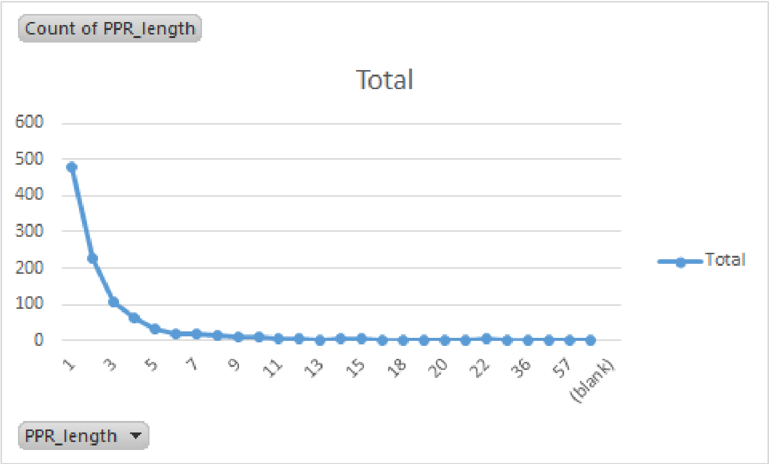
name	SUM PPR	SUM PR	Diff on Sc	AVG PPR	AVG PR	Diff on Av
asktoolba	6.31389	8.788594	1	0.789236	1.098574	-1
avg	7.411068	8.509852	-1	0.336867	0.386811	-1
avira	0.248686	0.45046	2	0.082895	0.150153	0
babylon	16.41418	23.22184	0	0.781627	1.105802	1
blekko	0.41841	0.437148	-2	0.059773	0.06245	0
conduit	0	0	0	0	0	0
googletoc	10.12613	13.40115	0	0.843844	1.116762	0
imesh	6.086083	7.05849	0	0.55328	0.641681	1
incredi	3.746521	5.792104	1	0.31221	0.482675	2
hotspot	4.511167	2.00773	-4	0.644452	0.286819	-6
kaspersky	1.78395	2.65493	2	0.35679	0.530986	2
montiera	0.012401	0.017516	0	0.012401	0.017516	1
norton	2.393749	3.229025	1	0.299219	0.403628	2
searchtoo	0.276056	0.232962	-1	0.034507	0.02912	-1
softonic	0.954916	0.825277	0	0.059682	0.05158	0
speedbit	39.29646	59.01855	0	1.964823	2.950927	0
sweetim	2.246682	2.492273	0	0.449336	0.498455	0
trendmicr	0.374014	0.561263	1	0.034001	0.051024	1
zonealarm	0.089948	0.107897	0	0.01285	0.015414	-1

	SUM PPR	SUM PR	Diff on Sc	AVG PPR	AVG PR	Diff on Av
asktoolba	9.359327	8.788594	1	1.169916	1.098574	-1
avg	6.052736	8.509852	1	0.275124	0.386811	-1
avira	0	0	0	0	0	0
babylon	16.88074	23.21877	1	0.844037	1.160938	2
blekko	0.339957	0.437148	-1	0.048565	0.06245	1
conduit	0.474404	0.519592	-1	0.067772	0.074227	0
googletoc	9.6541	13.40115	1	0.804508	1.116762	2
imesh	20.73694	7.05849	-4	1.885177	0.641681	-3
incredi	4.141737	5.792104	0	0.345145	0.482675	-1
hotspot	1.421038	2.00773	0	0.203005	0.286819	0
kaspersky	1.66541	2.65493	1	0.333082	0.530986	2
montiera	0.013918	0.017516	0	0.013918	0.017516	-1
norton	2.071305	3.229025	0	0.258913	0.403628	1
searchtoo	0.24292	0.232962	0	0.030365	0.02912	0
softonic	0.716003	0.820086	0	0.051143	0.058578	-1
speedbit	57.68013	59.02209	0	2.746673	2.810576	0
sweetim	1.995431	2.492273	-1	0.399086	0.498455	-1
trendmicr	0.331928	0.558572	2	0.033193	0.055857	0
zonealarm	0.078464	0.10539	0	0.013077	0.017565	1

Appendix A.1: PR scores



Appendix A.2: Graph scores



Appendix A.3: Personaliztion vector length

Bibliography

Abrahao, B., Soundarajan, S., Hopcroft, J., and Kleinberg, R. (2012). On the separability of structural classes of communities. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 624–632. ACM.

Bauer, L., Cai, S., Jia, L., Passaro, T., and Tian, Y. (2014). Analyzing the dangers posed by chrome extensions. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 184–192. IEEE.

Bekkerman, R., El-Yaniv, R., and McCallum, A. (2005). Multi-way distributional clustering via pairwise interactions. In *ICML*.

Bekkerman, R. and Jeon, J. (2007). Multi-modal clustering for multimedia collections. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Bekkerman, R. and Sahami, M. (2006). Semi-supervised clustering using combinatorial mrfs. In *ICML-06 Workshop on Learning in Structured Output Spaces*.

Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. (2009). Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 49–62. ACM.

Bilenko, M. and Richardson, M. (2011). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Brandes, U. and Erlebach, T. (2005). *Network analysis: methodological foundations*, volume 3418. Springer Science & Business Media.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI)*.

Brin, S., Motwani, R., Page, L., and Winograd, T. (1998). What can you do with a web in your pocket? *IEEE Data Eng. Bull.*, 21(2):37–47.

Chen, H. and Karger, D. R. (2006). Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 429–436. ACM.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Elzer, S., Schwartz, E., Carberry, S., Chester, D., Demir, S., and Wu, P. (2007). A browser extension for providing visually impaired users access to the content of bar charts on the web. In *WEBIST (2)*, pages 59–66.

Freschi, V. (2007). Protein function prediction from interaction networks using a random walk ranking algorithm. In *Bioinformatics and Bioengineering, 2007. BIBE 2007. Proceedings of the 7th IEEE International Conference on*, pages 42–48. IEEE.

Gauch, S., Speretta, M., Chandramouli, A., and Micarelli, A. (2007). User profiles for personalized information access. In *The adaptive web*, pages 54–89. Springer.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Golder, S. A. and Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of information science*, 32(2):198–208.

Gong, S., Ye, H., and Tan, H. (2009). Combining memory-based and model-based collaborative filtering in recommender system. In *Circuits, Communications and Systems, 2009. PACCS'09. Pacific-Asia Conference on*, pages 690–693. IEEE.

Han, J., Kamber, M., and Pei, J. (2006). *Data mining, southeast asia edition: Concepts and techniques*. Morgan kaufmann.

Haveliwala, T., Kamvar, S., and Jeh, G. (2003). An analytical comparison of approaches to personalizing pagerank.

Haveliwala, T. H. (2002). Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM.

Haveliwala, T. H. (2003). Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):784–796.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115.

Jeh, G. and Widom, J. (2003). Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM.

Joachims, T., Freitag, D., and Mitchell, T. M. (1997). Webwatcher: A tour guide for the world wide web. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*.

Koch, N. and Wirsing, M. (2001). Software engineering for adaptive hypermedia applications. In *8th International Conference on User Modeling, Sonthofen, Germany*. Citeseer.

Koschützki, D., Lehmann, K. A., Peeters, L., Richter, S., Tenfelde-Podehl, D., and Zlotowski, O. (2005). Centrality indices. In *Network analysis*, pages 16–61. Springer.

Leontiadis, I., Efstratiou, C., Picone, M., and Mascolo, C. (2012). Don't kill my ads!: balancing privacy in an ad-supported mobile application market. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 2. ACM.

Lieberman, H. et al. (1995). Letizia: An agent that assists web browsing. *IJCAI (1)*, 1995:924–929.

Lu, T. and Boutilier, C. (2011). Learning mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 145–152.

Marin, L., Isern, D., and Moreno, A. (2013). Dynamic adaptation of numerical attributes in a user profile. *Applied intelligence*, 39(2):421–437.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web.

Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331.

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58.

Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C. (2005). Stronger password authentication using browser extensions. In *Usenix security*, pages 17–32. Baltimore, MD, USA.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys*, 34(1):1–47.

Sebastiani, F. (2003). Machine learning in automated text categorization. *ACM computing surveys*, 34(1):1–47.

Si, L. and Jin, R. (2003). Flexible mixture model for collaborative filtering. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*.

Sugiyama, K., Hatano, K., and Yoshikawa, M. (2004). Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM.

Ungar, L. H. and Foster, D. P. (1998). Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1.

Vapnik, V. N. (1999). An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5):988–999.

Voorhees, E. M. et al. (1999). The trec-8 question answering track report. In *TREC*, volume 99, pages 77–82.

Zhu, R. and Gong, S. (2009). Analyzing of collaborative filtering using clustering technology. In *Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on*, volume 4, pages 57–59. IEEE.