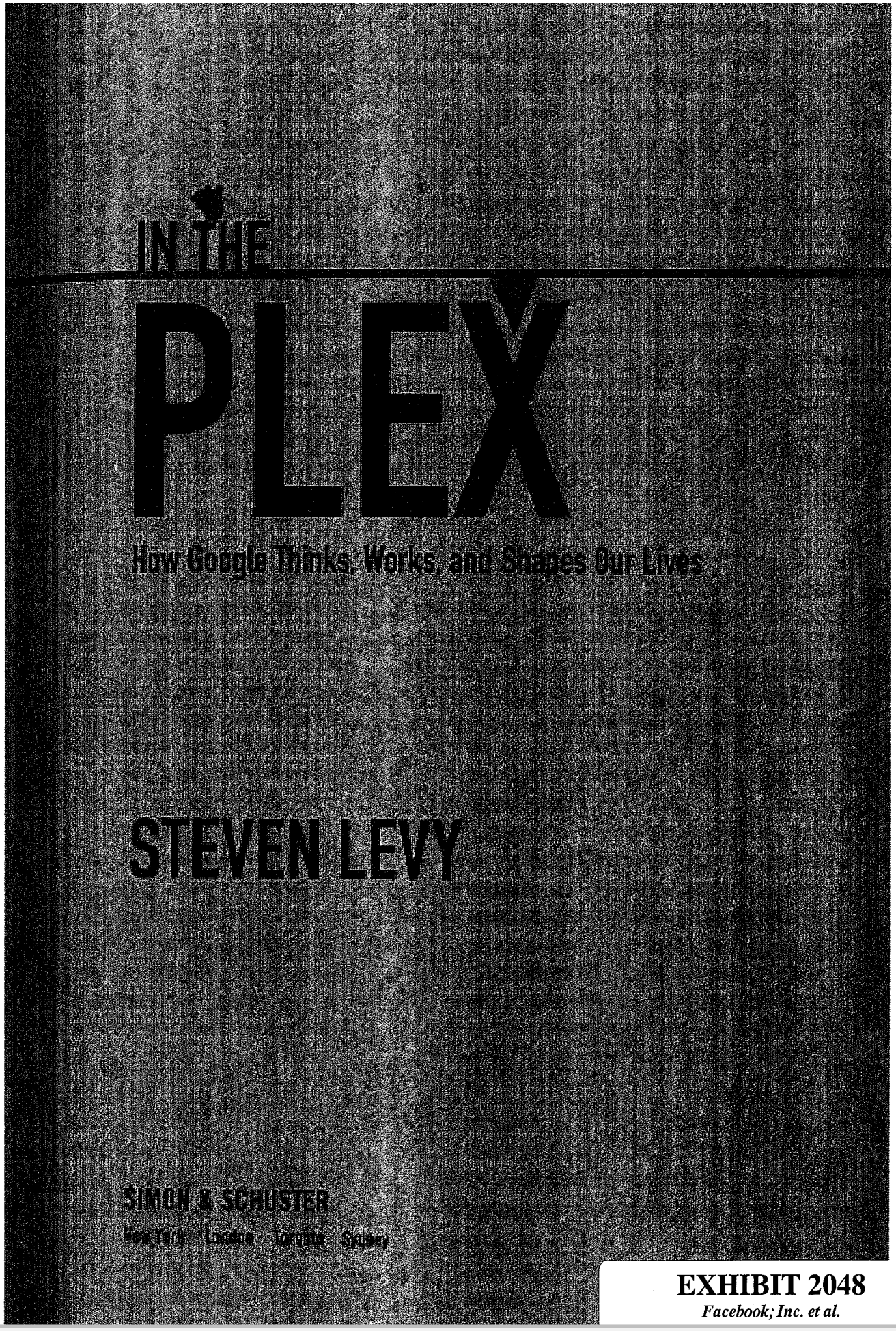

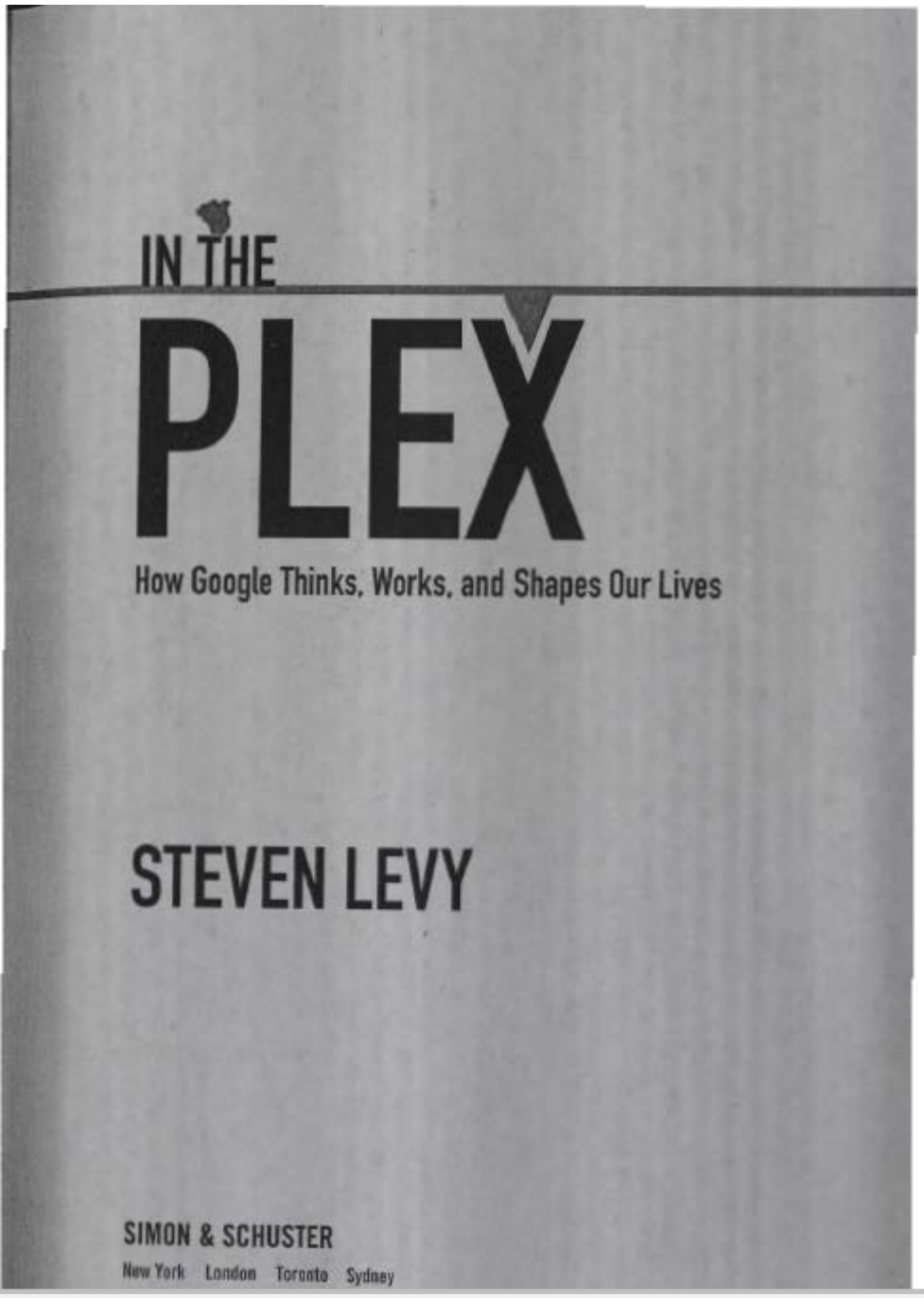

Simon & Schuster
1230 Avenue of the Americas
New York, NY 10020

First Simon & Schuster hardcover edition April 2011

SIMON & SCHUSTER and colophon are registered trademarks of Simon & Schuster, Inc.

For information about special discounts for bulk purchases, please contact Simon & Schuster Special Sales at 1-866-506-1949 or business@simonandschuster.com

The Simon & Schuster Speakers Bureau can bring authors to your live event. For more information or to book an event contact the Simon & Schuster Speakers Bureau at 1-866-248-3049 or visit our website at www.simonspeakers.com.

Designed by Ruth Lee Mui

Manufactured in the United States of America

10 9 8 7 6 5 4 3

Library of Congress Cataloging-in-Publication Data

Levy, Steven.
In the plex : how Google thinks, works, and shapes our lives / Steven Levy.
—1st Simon & Schuster hbk. ed.
p. cm.
Includes bibliographical references and index.
1. Google (Firm). 2. Google. 3. Internet industry—United States. I. Title.
HD9696.8.U64G6657 2011
338.7'6102504—dc22 2010049964

ISBN 978-1-4165-9658-5
ISBN 978-1-4165-9671-4 (ebook)

than others. Page realized that such data already existed and no one else was really using it. He asked Brin, "Why don't we use the links on the web to do that?"

Page, a child of academia, understood that web links were like citations in a scholarly article. It was widely recognized that you could identify which papers were really important without reading them—simply tally up how many other papers cited them in notes and bibliographies. Page believed that this principle could also work with web pages. But getting the right data would be difficult. Web pages made their outgoing links transparent: built into the code were easily identifiable markers for the destinations you could travel to with a mouse click from that page. But it wasn't obvious at all what linked *to* a page. To find that out, you'd have to somehow collect a database of links that connected to some other page. Then you'd go *backward*.

That's why Page called his system BackRub. "The early versions of hypertext had a tragic flaw: you couldn't follow links in the other direction," Page once told a reporter. "BackRub was about reversing that."

Winograd thought this was a great idea for a project, but not an easy one. To do it right, he told Page, you'd really have to capture a significant chunk of the World Wide Web's link structure. Page said, sure, he'd go and download the web and get the structure. He figured it would take a week or something. "And of course," he later recalled, "it took, like, years." But Page and Brin attacked it. Every other week Page would come to Garcia-Molina's office asking for disks and equipment. "That's fine," Garcia-Molina would say. "This is a great project, but you need to give me a budget." He asked Page to pick a number, to say how much of the web he needed to crawl, and to estimate how many disks that would take. "I want to crawl the *whole* web," Page said.

Page indulged in a little vanity in naming the part of the system that rated websites by the incoming links: he called it PageRank. But it was a sly vanity; many people assumed the name referred to web pages, not a surname.

Since Page wasn't a world-class programmer, he asked a friend to help out. Scott Hassan was a full-time research assistant at Stanford, working for the Digital Library Project program while doing part-time grad work. Hassan was also good friends with Brin, whom he'd met at an Ultimate Frisbee game during his first week at Stanford. Page's program "had so many bugs in it, it wasn't funny," says Hassan. Part of the problem was that Page was using the relatively new computer language Java for his ambitious project.

and Java kept crashing. "I went and tried to fix some of the bugs in J
itself, and after doing this ten times, I decided it was a waste of time," s
Hassan. "I decided to take his stuff and just rewrite it into the languag
knew much better that didn't have any bugs."

He wrote a program in Python—a more flexible language that
becoming popular for web-based programs—that would act as a "spider,"
called because it would crawl the web for data. The program would vis
web page, find all the links, and put them into a queue. Then it would ch
to see if it had visited those link pages previously. If it hadn't, it would
the link on a queue of future destinations to visit and repeat the proc
Since Page wasn't familiar with Python, Hassan became a member of
team. He and another student, Alan Steremberg, became paid assistants
the project.

Brin, the math prodigy, took on the huge task of crunching the ma
ematics that would make sense of the mess of links uncovered by th
monster survey of the growing web.

Even though the small team was going somewhere, they weren't qu
sure of their destination. "Larry didn't have a plan," says Hassan. "In
search you explore something and see what sticks."

By March 1996, they began a test, starting at a single page, the St
ford computer science department home page. The spider located
links on the page and fanned out to all the sites that linked to Stanfo
then to the sites that linked to *those* websites. "That first one just used
titles of documents because collecting the documents themselves requi
a lot of data and work," says Page. After they snared about 15 million
those titles, they tested the program to see which websites it deemed m
authoritative.

"Even the first set of results was very convincing," Hector Garc
Molina says. "It was pretty clear to everyone who saw this demo that t
was a very good, very powerful way to order things."

"We realized it worked really, really well," says Page. "And I sa
'Wow, the big problem here is not annotation. We should now use it
just for ranking annotations, but for ranking *searches*.'" It seemed the ob
ous application for an invention that gave a ranking to every page on
web. "It was pretty clear to me and the rest of the group," he says, "that
you have a way of ranking things based not just on the page itself but bas
on what the world thought of that page, that would be a really valua
thing for search."

* * *

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.