افیک F1

فصل‌نامه‌ی علمی انجمن علمی مهندسی کامپیوتر

شماره‌ی پانزدهم :: پاییز ۱۳۹۴ :: ۴۶ صفحه

مسیر پیش رو؛
ارشد، دکترا، اپلای یا کار!

موضوع فصل:

داده‌های بزرگ

همراه با پرونده‌ای در باب نرم‌افزارهای آزاد و داستان جادی درباره‌ی دانشمندان داده

اداره‌ی کل امور فرهنگی و اجتماعی
معاونت مشارکت‌های
فرهنگی و اجتماعی دانشگاهیان

پردیس دانشکده‌های فنی
دانشگاه تهران

به‌نام خدا

F1

فصل‌نامه‌ی اف‌یک

شماره‌ی پانزدهم - پاییز ۱۳۹۴

فهرست مطالب

F1

فصل‌نامه‌ی علمی
انجمن علمی مهندسی کامپیوتر (ACM)
دانشکده‌ی مهندسی برق و کامپیوتر دانشگاه تهران

با تشکر فراوان از
شمیم طاهری, امیرمسعود زارع‌بیدکی, کافه‌بازار
و
حامی مالی: شرکت نرم‌افزاری داتیس
آرین قشم (داتین)

acm@ece.ut.ac.ir
http://acm.ut.ac.ir/

سخن اول

سخن اول آخرین بخشی‌ست که در یک مجله نگاشته می‌شود، معمولاً هم ناخوانده باقی می‌ماند! از این رو تلاش کردم تا سخنان این بخش در راستای اهداف مجله و مقدمه‌ای مفید بر موضوعات آن باشد.

شاید این جمله‌ی پیتر نورینگ، سرپرست بخش پژوهش گوگل، که: «ما الگوریتم‌های بهتری نداریم، صرفاً داده‌ی بیشتری داریم» اهمیت داده‌های بزرگ را در روزگار ما بهتر بیان کند. از این رو تیم اف‌یک «موضوع فصل» این شماره را به «داده‌های بزرگ» اختصاص داد. این بخش نخست به مروری کلی بر مفهوم‌ها و اهمیت داده‌های بزرگ و همچنین کاربردهای آن می‌پردازد؛ در ادامه، معماری ابزارهای پردازش داده‌های بزرگ را معرفی می‌کند و تا بررسی برخی از جزئیات پیش می‌رود. در انتها روایتی جذاب از سه دانشمند داده آورده شده است که برای آزمون استخدام به یک پرسش با سه رویکرد گوناگون نگاه می‌کنند.

دغدغه‌ی آینده و چندراهی اپلای، کار، کارشناسی ارشد یا دکترا همیشه ذهن دانش‌جویان را درگیر کرده است، از این رو در این شماره کوشش شده است تا آگاهی بیشتری از هر رویکرد به خوانندگان داده شود. دو مصاحبه و یک نشست با حضور افرادی راه‌آزموده برگزار شد که برآیند آن‌ها در بخش «دانشکده» آورده شده است. توصیه می‌کنیم این بخش را از دست ندهید.

مقاله‌های گوناگونی در بخش پایانی آورده شده است. تلاش ما بر این بود تا از جای‌جای علم کامپیوتر مطالبی گرد آوریم که سلیقه‌های گوناگون را پوشش دهد. از موضوعات مختلفی که در این بخش به آن‌ها پرداخته شده است می‌توان به بازی‌سازی، شرحی ابتدایی بر برنامه‌نویسی وب، استارت‌آپ‌ها، دست‌آوردهای آلن تورینگ و فلسفه‌ی آزادی‌خواهی نرم‌افزار اشاره کرد. امید است که این مقالات برای خوانندگان سودمند باشند.

رسیدن هنر گام زمان است
احسان حاج‌یاسینی
سردبیر

در دانشکده‌ی مهندسی برق و کامپیوتر

در بخش دانشکده‌ی این شماره، علاوه بر مرور رویدادهای این فصل دانشکده و شاخه‌ی دانش‌جویی ACM، موضوع «آینده‌ی پس از مقطع کارشناسی» که از دغدغه‌های اصلی هر دانش‌جوی این مقطع به‌حساب می‌آید مورد بررسی قرار گرفته است. این مسئله‌ی مهم در قالب چند گفت‌وگو مطرح شده است که برای آگاهی از نظرات مختلف پیرامون این موضوع می‌توانید آن‌ها را مطالعه کنید.

تصویر مربوط به بورد همایش پنجره است که دانش‌جویان می‌توانستند سوالات‌شان را پیرامون آینده‌ی شغلی و تحصیلی خود روی آن مطرح نمایند

تازه‌های ACM

جشن ACM، بهانه‌ای برای گرد هم جمع‌شدن اعضا

امیررضا دادفرنیا
adadfarnia@acm.org

گردهمایی شاخه‌ی دانشجویی ACM فرصتی است که اعضای شاخه در کنار هم جمع شده و اعضای قدیمی و جدید با یکدیگر بیشتر آشنا شوند. برنامه‌ای که جای خالی آن در بین برنامه‌های ACM دیده می‌شد. سال‌های گذشته هم تصمیم بر برگزاری این برنامه وجود داشت اما هیچ‌گاه این امکان محقق نشد. امسال در روز سه‌شنبه ۲۹ اردی‌بهشت برای اولین بار این جشن برگزار شد.

از چند روز قبل از جشن هیجان خاصی بین دانشجویان قابل مشاهده بود و هر کس به کاری مشغول بود؛ یک نفر در حال مصاحبه با اعضای ACM در خارج از کشور بود، یکی در پی خرید جوایز و دیگری در فکر برنامه‌ریزی. تعدادی نیز در هیجان انتخاب بهترین‌های دو سال گذشته‌ی ACM و معرفی کاندیداها بودند. در روز جشن نیز برگزاری مسابقه‌ی ریمن[1] در اتاق ACM به شور و اشتیاق موجود افزوده بود.

با شروع برنامه ابتدا دکتر رامتین خسروی، سرپرست شاخه، سخنی دوستانه با دانشجویان داشتند و به معرفی شاخه و فعالیت‌های گذشته پرداختند. سپس حامد زمانی، رئیس ACM در سال ۱۳۹۰، به معرفی مسابقات ICPC پرداخت. پس از سخنرانی اسپانسر برنامه، نوبت به برنامه‌های اصلی رسید.

مرحله‌ی پایانی مسابقه‌ی ریمن و تغییر صدا با هلیوم با استقبال شرکت‌کنندگان همراه شد. پس از آن برای اولین بار در دانشکده، از افتخارآفرینان کنکور کارشناسی ارشد و المپیاد دانشجویی تجلیل شد و از طرف شاخه‌ی دانشجویی ACM یادگاری به آن‌ها اهدا شد.

با قاطعیت می‌توان گفت کلیپ معرفی شاخه‌ی ACM - که ویژه‌ی این جشن تهیه شده بود - بهترین بخش برنامه بود که مورد تشویق حضار قرار گرفت. در ادامه نیز بهترین همراهان ACM در دو سال گذشته در بخش‌های مختلف (افیک، طراحی پوستر و طرح جلد، ایده‌ی کاربردی، کلاس تابستانی، بهترین حلقه و...) معرفی شدند. امیدواریم این برنامه مقدمه‌ای برای برگزاری سالانه و بهتر جشن ACM باشد.

1 - Rayman Legends

جشن اوبونتو

پویا مرادی
pooya.moradi@acm.org

همایش گنو/ لینوکس و نرم‌افزارهای آزاد و متن‌باز، چند سال است که توسط انجمن کاربران ایرانی اوبونتو در دانشگاه‌های مختلف و با همکاری انجمن‌های علمی آن‌ها برگزار می‌شود. یکی از مناسبت‌های این همایش، جشن انتشار نسخه‌های مختلف توزیع لینوکس پرطرفدار اوبونتو است. در دو سال اخیر، این جشن همراه با کارگاه‌هایی در حوزه‌ی لینوکس و نرم‌افزارهای آزاد بوده است که تعدادی از آن‌ها از طرف انجمن کاربران ایرانی اوبونتو و تعدادی از طرف انجمن علمی دانشکده‌ی برگزارکننده، تشکیل می‌شود.

امسال، هفدهمین همایش گنو/ لینوکس و نرم‌افزارهای آزاد و متن‌باز با مسئولیت شاخه‌ی دانشجویی ACM دانشگاه تهران برگزار شد و اسپانسر این جشن شرکت رمیس بود. از طرف شاخه‌ی دانشجویی ACM، بشیر سجاد کارگاه نگاشت‌کاهش و مهدی دهقانی کارگاه node.js را برگزار کردند. همچنین از طرف انجمن کاربران ایرانی اوبونتو، کارگاه‌های Go lang، رزبری‌پای، فایروال، اوبونتو از صفر تا صد، لاراول، و روبی‌آن ریلز، و از طرف اسپانسر نیز کارگاه VyOS برگزار شد.[1]

از چندین هفته قبل از برگزاری مراسم در تاریخ ۷ خرداد ۱۳۹۴، جلسه‌ای تشکیل شد تا مسئولیت‌های مختلف این جشن بین دانشجویان تقسیم شود. دانشجویان زیادی برای کارها داوطلب شدند و نهایتاً با کمک‌های شایان بچه‌ها در طول دو هفته‌ی پایانی، این جشن برگزار شد.

ثبت‌نام همایش به این شکل بود که هر فردی باید دقیقاً یک کارگاه را برای شرکت‌کردن انتخاب می‌کرد؛ چرا که کارگاه‌ها به‌صورت هم‌زمان برگزار می‌شدند. پراستقبال‌ترین کارگاه در همایش امسال، کارگاه اوبونتو از صفر تا صد بود که توسط دانیال بهزادی ارائه شد.

برنامه به این صورت بود که شرکت‌کنندگان از ساعت ۸:۳۰، در محوطه‌ی جلوی دانشکده‌ی معدن پذیرش شده و در ساختمان کلاس‌های دانشکده برق و کامپیوتر مستقر شدند. امسال با موازی‌کردن روند پذیرش، سرعت پذیرش افزایش یافت که از نکات مثبت جشن امسال نسبت به جشن‌های پیشین بود.

به‌دلیل هم‌زمانی پذیرش شرکت‌کنندگان و مستقرشدن داخل کارگاه‌ها، نیاز به مدیریت هم‌زمان این دو فرآیند بود که به‌دلیل هم‌کاری بسیار خوب و منظم بچه‌ها در قسمت‌های مختلف دانشکده (ساختمان کلاس‌ها و ساختمان معدن) به‌خوبی صورت گرفت و سپس کارگاه‌ها شروع شد.

به‌دلیل نبود زیرساخت مناسب در دانشکده، ابتدا قرار بود شرکت مبین‌نت وظیفه‌ی تأمین اینترنت جشن را برعهده بگیرد، اما به‌دلایلی این امر در روزهای پایانی میسر نشد و ما مجبور شدیم که خود اینترنت کارگاه‌ها را تأمین کنیم. بعد از برگزاری کارگاه‌ها و پذیرایی مختصر، شرکت‌کنندگان به‌سمت آمفی‌تئاتر دانشکده‌ی معدن هدایت شدند. متأسفانه در آن روز، با وجود این‌که مجوز آمفی‌تئاتر را برای همایش گرفته بودیم، مسئول آمفی‌تئاتر حضور نداشت و ما خود آمفی‌تئاتر را راه‌اندازی کردیم.همچنین به دلیل عدم همکاری مناسب دانشگاه، چندین بار دچار مشکلات فنی شدیم.

برخلاف انتظار ما و تأکید انجمن کاربران ایرانی اوبونتو بر نیامدن بسیاری از ثبت‌نام کنندگان، استقبال از همایش زیاد بود و تعداد قابل توجهی از شرکت‌کنندگان روی زمین نشستند و تعدادی هم بیرون آمفی‌تئاتر ایستادند.

در ابتدا کلیپی برای معرفی شاخه‌ی دانشجویی ACM پخش شد. سپس محمد تشکری از وزارت اقتصاد و دارایی درباره‌ی نیاز شدید سازمان‌های دولتی به افراد مسلّط به لینوکس صحبت کرد. در بخش بعدی سخنرانی‌ها، بهراد غیاث‌الدین مدیرعامل شرکت رمیس، درباره‌ی شروع کار این شرکت و پروژه‌های بزرگ‌شان توضیحاتی ارائه کرد. در بخش بعدی، ایریکس اسماعیلی درباره‌ی فرازونشیب‌های اوبونتو و شرکت کنونیکال در ۱۰ سال اخیر صحبت کرد.

بعد از سخنرانی ایریکس، نوبت به سخنرانی جادی که از وبلاگ‌نویس‌های محبوب وب فارسی و از فعالان حوزه‌ی لینوکس و نرم‌افزارهای آزاد است رسید که با تشویق حضار به روی صحنه آمد. او درباره‌ی اخبار دنیای نرم‌افزارهای آزاد و متن‌باز از جمله نام عجیب‌وغریب کرنل ۴ لینوکس که Hurr durr I'm a sheep است صحبت کرد. در قسمت آخر، امیر صدیقی درباره‌ی داده‌های بزرگ سخنرانی کرد.

بعد از اتمام سخنرانی‌ها، جایزه‌ای از طرف انجمن کاربران ایرانی اوبونتو به ایریکس اسماعیلی به پاس زحماتی که برای پیشرفت انجمن در این چند سال کشیده است اهدا شد.

۱ - ویدیوهای این کارگاه‌ها توسط اسپانسر ضبط شده و از طریق وبسایت http://gotoclass.ir در دسترس عموم است.

ACM EXCELLENCE AWARD 2015

OUTSTANDING SCHOOL SERVICE

پژمان حسینی
pejman.invincible19@gmail.com

هر سال در اواخر اسفند و اوایل فروردین، همه در فکر و شور و اشتیاق عید و سال نواند، اما اعضای شاخه‌ی دانش‌جویی ACM در تکاپوی آماده‌کردن گزارش‌های برنامه‌های یک سال گذشته‌ی خود برای ارائه به انجمن ACM هستند.

انجمن ماشین‌های محاسباتی[1] بزرگ‌ترین جامعه‌ی علمی و آموزشی کامپیوتری است. شاید بیشتر ACM را با مسابقات برنامه‌نویسی بین‌المللی دانشگاهی[2] بشناسیم که معتبرترین مسابقه‌ی برنامه‌نویسی به‌شمار می‌رود. اما بهتر است بدانیم که مهم‌ترین جایزه در حوزه‌ی کامپیوتر توسط این انجمن اهدا می‌شود. جایزه‌ی تورینگ به‌عنوان جایزه‌ی نوبل محاسبات جهان شناخته می‌شود.

شاخه‌های دانش‌جویی ACM به‌عنوان نماینده‌های این انجمن در سراسر دنیا اقدام به گسترش و آموزش علوم محاسباتی و کامپیوتری می‌کنند. هم‌اکنون ۲۴۳ شاخه‌ی دانش‌جویی در ۲۶ کشور دنیا در حال فعالیت است. ACM هر سال در پنج زمینه، بهترین شاخه‌ی دانش‌جویی را انتخاب می‌کند. این پنج زمینه عبارت‌اند از:

۱- **فعالیت‌ها:** در این بخش کلیه‌ی فعالیت‌های یک شاخه بررسی می‌شود و شاید این جایزه به‌دلیل نگاه کلی به شاخه‌ها و بررسی تمامی فعالیت‌هایشان مهم‌ترین جایزه باشد.

۲- **وبسایت:** همان‌طور که از اسم آن پیدا است، وبسایت شاخه از نظر طراحی، واسط کاربری و اطلاع‌رسانی برنامه‌ها مورد ارزیابی قرار می‌گیرد.

۳- **عضویت:** در این بخش، برنامه‌هایی که یک شاخه برای معرفی ACM به دانش‌جویان و اضافه‌کردن آن‌ها به جامعه‌ی کامپیوتری جهان برگزار می‌کند بررسی می‌شود.

۴- **خدمات ارائه‌شده به دانشگاه:** پروژه‌ها و برنامه‌هایی که شاخه‌ها برای ارتقاء سطح آموزش دانشگاه خود انجام می‌دهند عاملی است تا برترین شاخه در این بخش مشخص شود.

۵- **خدمات ارائه‌شده به جامعه:** از شاخه‌ها انتظار می‌رود که فراتر از دانشگاه خود فعالیت کنند و به گسترش سطح اطلاعات کامپیوتری و محاسباتی جامعه بپردازند.

در ماه‌های فروردین و اردی‌بهشت، شاخه‌های دانش‌جویی فرصت دارند تا با ارسال گزارش فعالیت‌های یک سال گذشته‌ی خود، در بخش مورد نظر خود کاندید شوند.

شاخه‌ی دانش‌جویی ACM دانشگاه تهران در سال‌های ۲۰۱۲-۲۰۱۳ و ۲۰۱۱-۲۰۱۲ موفق شد که در بخش عضویت به‌عنوان بهترین شاخه‌ی دانش‌جویی انتخاب شود. در سال گذشته و با توجه به گسترش فعالیت‌های شاخه در دانشگاه، تصمیم بر آن شد که بیشترین سرمایه‌گذاری برای دریافت جایزه در بخش دانشگاه صورت بگیرد. خوشبختانه در اوایل اردی‌بهشت و با اعلام برترین‌های ACM، طبق پیش‌بینی‌ها، ACM دانشگاه تهران برای سومین بار و این بار در بخش دانشگاه، موفق به کسب عنوان بهترین شاخه شد.[3]

برنامه‌هایی از قبیل حلقه، مسابقات هفتگی برنامه‌نویسی، کارگاه شناخت توانایی‌ها، کلاس‌های تابستانی، مسابقه‌ی هوش مصنوعی و برنامه‌ی انتخاب واحد مجازی از مهم‌ترین فعالیت‌های شاخه‌ی دانش‌جویی ACM در سال گذشته بود که در انتخاب این شاخه به‌عنوان یکی از پنج شاخه‌ی برتر دنیا تأثیر زیادی داشت.

افراد زیادی در راه دریافت این جایزه تلاش کردند، فعالیت‌هایی که در طول یک سال دوره‌ی «آبی لاجوردی»[4] انجام شد و تلاش‌های زیادی که برای مدون‌کردن و نوشتن گزارش‌ها صورت گرفت. اعضای شاخه‌ی دانش‌جویی ACM از تمامی افرادی که در این مسیر شاخه را یاری کردند، به‌ویژه دکتر رامتین خسروی، سرپرست شاخه، و همچنین زنده‌یاد دکتر مهدی فخرایی که از پایه‌گذاران این شاخه بود تشکر می‌کند.

1 - Association for Computing Machinery (ACM)
2 - International Collegiate Programming Contest (ICPC)
3 - http://www.acm.org/chapters/students/essay-contest/essay_contest-toc
4 - Persian Blue

ACM's How to Apply 2015

محمد بابامحمودی
mohammadbabamahmoudi@gmail.com

آیا واقعاً قصد ادامه‌ی تحصیل دارید؟ چرا در داخل ادامه نمی‌دهید؟ مزایا و معایب تحصیل و زندگی در خارج از کشور را می‌دانید؟ همه‌ی این‌ها سؤالاتی هستند که برای گرفتن یک پذیرش خوب به پاسخی مناسب نیاز دارند. در گذشته دانش‌جویان و فارغ‌التحصیلان برای یافتن جواب با افراد مختلفی صحبت می‌کردند. مانند کسانی که ادامه‌ی تحصیل نداده‌اند و پس از دوره‌ی کارشناسی وارد بازار کار شده‌اند، آن‌ها که برای ادامه‌ی تحصیل در ایران مانده‌اند، آن‌هایی که به خارج از کشور رفته‌اند، آن‌ها که پس از اتمام تحصیل بازگشته‌اند و آن‌ها که بازنگشته‌اند. همه‌ی این افراد حرف‌ها و تجربه‌هایی دارند که در نوع خود حتماً شنیدنی و ارزشمند است، اما مشکل این روش این است که وقت و انرژی بسیار زیادی از افراد می‌گیرد و چه بسا ممکن است آن‌ها را دچار سرگردانی کند و روند پذیرش‌گرفتن را دشوارتر و پراسترس‌تر.

به همین دلیل است که در چند سال اخیر در دانشگاه تهران همایشی تحت عنوان How To Apply برگزار می‌شود که به دانش‌جویان کمک می‌کند تا راحت‌تر این مسیر دلهره‌آور را طی کنند. یکی از این همایش‌ها در روز هفتم مرداد ۱۳۹۴ در آمفی‌تئاتر دانشکده‌ی مهندسی برق و کامپیوتر دانشگاه تهران به میزبانی شاخه‌ی دانش‌جویی ACM برگزار گردید. در این همایش مطالب مربوط به روند پذیرش‌گرفتن طی چند مرحله به‌طور مفصل شرح داده شد و ارائه‌دهندگان هر مرحله خود از دانش‌جویانی بودند که این مسیر را گذرانده بودند و نکات مفید و تجارب شخصی خود را به سایر دانش‌جویان منتقل کردند. در ادامه توضیح مختصری در مورد هر یک از مراحل داده می‌شود.

یکی از مهم‌ترین مراحل این روند که در ابتدا به آن پرداخته شد این بود که دانش‌جویان در مورد زمینه‌های مختلف تحقیقاتی در رشته‌ی تحصیلی خود یا رشته‌ی تحصیلی مورد علاقه‌شان جست‌وجو کنند و شناخت کلی به‌دست آورند. این کار با پرس‌وجو از دانش‌جویان تحصیلات تکمیلی داخل و خارج از کشور، سرزدن به وبسایت گروه‌های تحقیقاتی دانشگاه‌های مختلف، بررسی موضوعات پیشنهادی مقالات روی سایت کنفرانس‌های معتبر علمی، صحبت با اساتید دانشگاه و... قابل انجام است.

در بخش دیگر همایش به یکی دیگر از مراحل لازم برای پذیرش‌گرفتن از دانشگاه‌های خارج از کشور، یعنی مدرک زبان، پرداخته شد. معمولاً در مرحله‌ی پرکردن فرم‌های تقاضا برای بررسی پرونده، دانش‌جویان باید نمره‌ی آزمون‌های زبان مثل تافل و آیلتس از یکی از مؤسسات معتبر را ارائه کنند. اطلاعات مورد نیاز این مرحله در اختیار شرکت‌کنندگان همایش قرار گرفت و همچنین برنامه‌ی مشخصی برای زمان شروع مطالعه‌ی زبان، نحوه‌ی مطالعه، معرفی کانون‌های زبان‌آموزی و... به افراد ارائه شد.

تهیه‌ی رزومه‌ی مناسب که به‌جرئت می‌توان گفت مهم‌ترین بخش روند پذیرش است در بخش دوم همایش مورد بحث قرار گرفت و سخنرانان این بخش با ارائه‌ی نمونه‌ی رزومه‌های مختلف و ذکر جزئیات و نکات کلیدی این مرحله را توضیح دادند.

در ادامه‌ی همایش به سایر نکات لازم برای تسهیل فرآیند پذیرش پرداخته شد. به‌عنوان مثال برای برآورد هزینه‌های روند پذیرش از قبیل هزینه‌های ارسال تقاضا[1] برای بررسی پرونده در هر دانشگاه، امتحان‌های زبان، ویزا، آزادکردن مدارک تحصیلی و نحوه‌ی پرداخت این هزینه‌ها اطلاعات سودمندی در اختیار دانش‌جویان قرار گرفت.

لازم به ذکر است که در انتهای هر یک از بخش‌های همایش، پنل پرسش‌وپاسخ برگزار می‌شد و شرکت‌کنندگان می‌توانستند از طریق سامانه‌ی پیامکی سؤالات خود را مطرح کرده و پاسخ آن‌ها را بیابند.

از مسائلی که این همایش را از سایر همایش‌های مشابه متمایز می‌کرد امکان مشاهده‌ی آن به‌صورت زنده از وبسایت ACM بود که با استقبال زیادی از سوی دانش‌جویان سایر دانشگاه‌ها نیز مواجه شد. از دیگر خصوصیات متمایز این همایش پرداختن به مسائلی بود که شاید به آن‌ها کمتر توجه می‌شود؛ مانند شرایط درس‌خواندن و زندگی‌کردن در کشورها و شهرهای مختلف، وضعیت آب‌وهوا، هزینه‌های زندگی و حمل‌ونقل که از طریق مصاحبه‌ی اینترنتی با کسانی که در خارج از کشور زندگی می‌کردند در قالب ویدیوکلیپ در اختیار حضار قرار گرفت.

۱- Application fee

علی شیراوند
alish.icu@gmail.com

راستش دقیق نمی‌دانم این ماجرا از کجا شروع شد. فکر می‌کنم اواسط تابستان ۹۳ بود. این سؤال همیشه در ذهن من مطرح بود که ما چرا درس می‌خوانیم؟ چه‌طور باید وارد بازار کار شویم؟ چه کارهایی را بعد از دانشگاه می‌توانیم انجام دهیم؟ آیا در دوره‌ی کارشناسی کار کنیم یا نکنیم؟ و تعداد زیادی سؤال دیگر. تابستان سال پیش جلسه‌ای از طرف ACM برگزار شد و ایده‌ی برگزاری همایشی مطرح شد که قرار بود در آن دانش‌جوهای رشته‌ی مهندسی کامپیوتر بیشتر در مورد آینده‌شان بدانند و فضای صنعتی این رشته را بیشتر بشناسند. من و دو نفر دیگر از هم‌دوره‌ای‌هایم مسئول برگزاری چنین همایشی شدیم. در آن زمان «فنعت»[1] یک دوره بود که برگزار شده بود، ولی این همایش بیشتر روی رشته‌ی مهندسی برق تمرکز داشت و برای رشته‌ی کامپیوتری که دانش‌جوهای آن به‌مراتب زودتر می‌توانند وارد بازار کار شوند مطالب زیادی بیان نشده بود. از طرفی هم از همان ابتدا نمی‌خواستیم پنجره (که البته آن موقع این اسم را نداشت) مانند فنعت باشد و با برگزاری یک همایش به پایان برسد. با چند نفر از استادهای دانشکده از جمله دکتر خسروی و مهندس شفیعی قرار گذاشتیم و راجع به این موضوع صحبت کردیم. فهمیدیم که مشکل دانشگاه‌های ما ریشه‌ای‌تر از این حرف‌هاست، خیلی ریشه‌ای‌تر. البته این مسئله هم مشخص بود که تغییر در ساختارهای مدیریتی و اجرایی کار سختی است و اگر کاری قرار است انجام شود باید خود دانش‌جوها دل‌شان برای خودشان بسوزد.

صحبت‌های اولیه انجام شد؛ وظایف روزبه‌روز بیشتر می‌شد و ما نیاز داشتیم که تیم‌مان را بزرگ‌تر کنیم. از افراد زیادی کمک گرفتیم تا در مورد نحوه‌ی برگزاری همایش و همچنین نحوه‌ی دعوت از شرکت‌ها سؤال کنیم. اوضاع ارتباط صنعت با دانشگاه واقعاً خوب نیست؛ شرکت‌ها حاضر نیستند نیازها و پروژه‌های خود را با دانشگاه در میان بگذارند و از طرف دیگر در دانشگاه هم تلاش کمی می‌شود که نیازهای صنعت شناخته شود و دانش‌جویان بر اساس نیازهای کشور پروژه انجام دهند. این مشکل بزرگی بود. ما نمی‌دانستیم چه‌طور باید با یک شرکت صحبت کنیم تا قانع شود بدون ارائه‌ی تبلیغات مستقیم در سخنرانی، در همایش شرکت کند. بهترین راه این بود که از اساتیدی که ارتباط بیشتری با شرکت‌ها دارند کمک بگیریم و از طریق آن‌ها همایش خودمان را معرفی کنیم. اساتید محترمی همچون دکتر نوابی، دکتر خسروی، دکتر شاه‌منصوری و مهندس ادهمی کمک‌های بسیاری در زمینه‌ی صحبت با مدیران شرکت‌ها به ما کردند.

نام «پنجره» اواخر پاییز بود که انتخاب شد. در ابتدا قرار بود که همایش در هفته‌ی آخر اسفند برگزار شود، ولی به‌دلیل شرایط نامناسب شرکت‌ها و همچنین عدم آمادگی تیم اجرایی، قرار شد که این زمان به ۲۰ الی ۲۲ی اردی‌بهشت تغییر کند. این تاریخ مناسبی بود زیرا امتحانات میان‌ترم تا آن زمان تمام می‌شد و تا امتحانات پایانی نیز فرصت خوبی باقی مانده بود. چند بخش اصلی برای کارهای همایش وجود داشت. مهم‌ترین بخش محتوای جلسه بود و ما باید به‌خوبی شرکت‌ها را توجیه می‌کردیم که این همایش فرصتی برای آشنایی دانش‌جویان با آینده‌شان و فضای صنعت است و نباید مطالب تبلیغاتی در سخنرانی‌ها بیان شود. تناسب پنل هر روزِ همایش با محتوای مطرح‌شده در آن روز نیز بسیار مهم بود. موضوع دیگر بحث جلب مخاطب و کارهای تبلیغاتی بود که در عمل بیشترین بار اجرایی را داشت. از تولید محتوای تبلیغاتی تا طراحی و چاپ آن. بخشی از تیم هم کارهای فنی مربوط به سایت و پخش زنده‌ی همایش را بر عهده گرفتند. عده‌ای نیز مسئولیت کارهای اجرایی روز همایش مانند پذیرایی و امور اجرایی سالن را پذیرفتند. در کل حدود ۲۵ نفر به‌طور مستقیم و غیرمستقیم درگیر کارهای پنجره شدند.

بالأخره پس از نه ماه از مطرح‌شدن ایده‌ی اولیه، ۲۰ اردی‌بهشت فرارسید. روز اول همایش بود و ما بسیار نگران بودیم که آیا بچه‌ها در همایش شرکت می‌کنند یا خیر. خوشبختانه سالن پر بود و همایش با سخنرانی دکتر خسروی شروع شد. همه چیز خوب پیش می‌رفت. بعد از آن نیز سخنرانی نمایندگان شرکت‌های کافه‌بازار و آدران اجرا شد. قصد این بود که در روز اول شرکت‌های استارت‌آپی سخنرانی کنند تا شرکت‌کنندگان جلسه بتوانند با فضای استارت‌آپ و کارآفرینی بیشتر آشنا شوند. بعد از آن هم نوبت به پنل رسید و دکتر محمدی نیز در کنار شرکت‌ها پاسخگوی سؤالات ما بود. موضوع پنل روز اول پیرامون ورود به فضای کار بود. برخی از مهمانان ازجمله دکتر محمدی عقیده داشتند ورود زودهنگام به بازار کار نادرستی است، ولی نظر نمایندگان شرکت‌ها برخلاف این عقیده بود.

ساعت ۱۲:۳۰ روز بعد، دومین روز همایش پنجره شروع شد. طبق برنامه قرار بود دکتر پاکروان در مورد شرکت مخابراتی پرمان سخنرانی داشته باشد و حوزه‌های فعالیت در صنعت شبکه‌ی ایران را معرفی نماید. در ادامه دکتر سجاد، که به‌تازگی به ایران آمده بود، در مورد اوضاع کارآفرینی در خارج از کشور و تفاوت‌های آن با کار در شرکت‌های بزرگ مباحثی را مطرح نمود. در انتها نیز مهندس فولادی، مدیر فروش شرکت همکاران سیستم، در مورد فضای کار صحبت کرد. در روز دوم همایش در زمان‌بندی به مشکل برخوردیم. بسیار از زمان عقب بودیم اما به‌دلیل این‌که با شرکت‌ها قرار گذاشته بودیم، نمی‌توانستیم زمان جلسه را کوتاه کنیم. ولی پنل روز دوم واقعاً پنل خوبی بود. با حضور دکتر خسروی و دکتر فرهنگی و نمایندگان شرکت‌ها بحث خوبی در مورد ارتباط صنعت با دانشگاه شکل گرفت و خیلی از مسائل به‌خوبی مطرح شد. در این پنل برخی مشکلات از جمله مهارت‌های ناکافی برخی دانش‌جویان برای ورود به بازار کار و همچنین نبود بستر مناسب برای فعالیت شرکت‌ها در دانشگاه بررسی شد.

روز دوم هم به پایان رسید و سرانجام به روز آخر همایش رسیدیم. سخنرانان روز سوم مهندس صادقیان‌پور از شرکت توسن، مهندس مسگری از شرکت داتین و مهندس شیخ‌عطار از شرکت ارگ‌تلکام بودند. در آخر همایش مهندس شیخ‌عطار سخنرانی خوبی را ارائه کرد و به بیان مشکلات ریشه‌ای عدم ارتباط صحیح صنعت با دانشگاه پرداخت. او با اشاره به روند ارتباط صنعت با دانشگاه در دیگر کشورها، مشکلات این چرخه را در ایران مورد بحث و بررسی قرار داد.

این اولین همایش پنجره بود که بهار امسال برگزار شد. ایرادات بسیاری داشت که امیدواریم سال آینده حل شود. ولی مهم‌تر از این همایش سه‌روزه، برنامه و تصمیم ما دانش‌جوها برای ادامه‌ی راه تحصیل‌مان است. ما باید تصمیم بگیریم و از دانشگاه بخواهیم مطالبی به ما تدریس شود که در آینده بتوانیم به‌وسیله‌ی آن مشکلی از مشکلات جامعه و کشورمان را حل کنیم. ما باید از دانشگاه بخواهیم که ارتباط بیشتری با صنعت داشته باشد و پروژه‌ها را بر اساس نیازهای صنعتی کشور تعریف کند. ما باید مشکلات‌مان را خودمان حل کنیم و منتظر نباشیم کسی دیگر چنین کاری برای ما انجام دهد. امیدواریم که همایش پنجره در سال آینده و با برنامه‌ها، کارگاه‌ها و حلقه‌های بیشتر برگزار شود.

۱ - همایش ارتباط فنی با صنعت بسیج دانشجویی دانشکده‌ی برق و کامپیوتر

این صفحه از نشریه‌ی اف‌یک قصد دارد فضای بیرون دانشگاه را به دانشجوهای دانشگاه تهران نشان دهد، در صورت امکان تبدیل به یکی از صفحه‌های همیشگی نشریه شود و خبرهای کسب‌وکار ایران، همایش‌های مرتبط و تجربه‌های کسانی را که در این راه بوده‌اند نشان دهد. نشریه‌ی اف‌یک از نظرها و پیشنهادهای شما استقبال می‌کند.

تازه‌های دانشکده

مراسم اختتامیه‌ی AppTalent 2015

هومن حبیب‌نیا
h.habibnia@gmail.com

مراسم اختتامیه‌ی مسابقات برنامه‌نویسی اندروید، AppTalent 2015، پنجشنبه ۴ تیر ۱۳۹۴ در دانشگاه تهران برگزار شد. در این مراسم، در کنار برگزارکنندگان مسابقه (شرکت موبیرو و همراه اول)، میهمانان ویژه‌ای از شرکت‌ها، اساتید دانشگاه تهران و همچنین جمعی از اصحاب رسانه حضور داشتند.

اولین سخنرانی توسط محمدرضا سوهانیان، مدیرکل خدمات ارزش افزوده‌ی همراه اول، با محوریت مدیریت اکوسیستم خدمات ارزش افزوده انجام شد. سوهانیان در صحبت‌های خود به این نکته اشاره کرد که فعالیت در این حوزه از سال ۹۱ آغاز شده و در حال حاضر ۱۳۰ شرکت با همراه اول در این زمینه همکاری می‌کنند که افتخاری در حوزه‌ی کارآفرینی به‌شمار می‌آید. وی در ادامه به سرمایه‌گذاری‌های عظیم همراه اول و سرویس‌های ارزش افزوده اشاره نمود و برگزاری مسابقاتی مانند AppTalent را گامی مؤثر در جهت پشتیبانی از تجاری‌سازی اپلیکیشن‌ها دانست.

پس از صحبت‌های سوهانیان، محمدرضا دولتیان، معاون تجاری همراه رسانه‌ی موج، پشت تریبون آمد و درباره‌ی مارکت اندرویدی «اول مارکت» توضیحاتی ارائه کرد و در ادامه، به دغدغه‌های اصلی توسعه‌دهندگان برنامه‌های موبایل و پیشرفت‌هایی در جهت حل این مسائل اشاره نمود.

پس از آن، نوبت به سخنرانی بهنام برنا، مدیر گروه خدمات همراه موبیرو، رسید. او درباره‌ی روندهای جدید در حوزه‌ی خدمات همراه توضیحاتی ارائه داد و درباره‌ی OTTها و لزوم تعامل اثربخش اپراتورها با آن‌ها در زمینه‌ی انواع سرویس‌های اینترنتی صحبت کرد.

در ادامه‌ی این مراسم، سیروس رزاقی در رابطه با صندوق‌های سرمایه‌گذاری خطرپذیر در حوزه‌ی موبایل و ICT سخنرانی نموده و آینده‌ی روشنی را با گسترش خدمات 3G و 4G برای ایران در این حوزه ترسیم کرد.

پس از اتمام سخنرانی‌ها، جلسه‌ی پرسش‌وپاسخ با محسن ملایری و محمدعلی طهرانچی توسط کتایون شکوری، مدیر پروژه‌ی مسابقات AppTalent در شرکت موبیرو، انجام شد. وی به توصیه‌های مفید و کاربردی در رابطه با تجاری‌سازی اپلیکیشن، نوآوری و تبلیغ و جذب مشتری پرداخت.

در بخش پایانی این اختتامیه، محمدعلی اخایی به‌عنوان یکی از داوران مسابقه به ارائه‌ی پیشنهاداتی در زمینه‌ی کارآفرینی و افزایش مهارت‌های اجتماعی و فراترفتن از آموزه‌های دانشگاهی و پرورش خلاقیت پرداخت.

این مسابقه در چهار بخش «بخش آزاد»، «ارتباطات»، «تجارت همراه» و «ایده‌ها» برگزار شد که در انتها به نفرات برگزیده‌ی هر چهار بخش لوح تقدیر و تندیس AppTalent اهدا شد.

تابستان با کد ۹۴

امیررضا دادفرنیا
adadfarnia@acm.org

همیشه در طول سال با خود فکر می‌کنیم که تابستان امسال، کارهای زیادی انجام می‌دهیم. چند کتاب را در ذهن خود آماده می‌کنیم و با خود می‌گوییم این کتاب‌ها را می‌خوانیم، زبان‌های برنامه‌نویسی x و y را یاد می‌گیریم و احتمالاً یک پروژه‌ی کاربردی را در ذهن خود تعریف می‌کنیم که در تابستان انجام دهیم. اما چیزی که معمولاً در پایان تابستان متوجه آن می‌شویم فقط گذر زمان است و فرصت‌هایی که از دست داده‌ایم.

برنامه‌ی «تابستان با کد»، که برگرفته از برنامه‌ای است که هر سال شرکت گوگل اجرا می‌کند، فرصتی است که به فکرهای خود امکان تبدیل‌شدن به واقعیت را بدهیم. همچنین این برنامه باعث می‌شود حتی در گرمای فصل تابستان، سایت دانشکده‌ی مهندسی برق و کامپیوتر خالی از جمعیت نباشد و شور و هیجان از دانشکده دور نشود.

از سال ۱۳۹۱، هرساله این برنامه در دانشگاه اجرا می‌شود. در ابتدا این برنامه به‌همت امیر صبوری و حامد زمانی شروع شد و یکی از بزرگ‌ترین دستاوردهای آن سایت شیرکد[1] است که بعید است دانش‌جوی مهندسی کامپیوتر دانشگاه تهران باشید و حداقل یک‌بار از آن استفاده نکرده باشید.

امسال این برنامه از مدت‌ها قبل و با هم‌فکری افراد زیادی آغاز شد. پس از پیشنهاد و بررسی پروژه‌های زیادی، در پایان هشت پروژه از بین آن‌ها انتخاب شد. لیست کامل پروژه‌ها در سایت تابستان با کد[2] قابل مشاهده است.

این برنامه فرصتی است که دانش‌جویان سال‌های اول و دوم با پروژه‌هایی در ابعاد بزرگ‌تر و واقعی‌تر از آن‌چه در درس‌ها می‌خوانند روبه‌رو شوند و کار گروهی را بیش از پیش تمرین کنند. همچنین برای دانش‌جویان سال‌های بالاتر این امکان فراهم می‌شود که توانایی خود را در مدیریت یک گروه بسنجند و بتوانند دانش خود را در اختیار دیگران قرار دهند.

همچنین از امسال با همکاری دانش‌جویان رشته‌ی مهندسی برق و شاخه‌ی دانش‌جویی ACM، برنامه‌ی مشابهی با نام «قلعه‌ی شنی»[3] برای آشنایی بیشتر دانش‌جویان این رشته با پروژه‌های صنعتی اجرا شد. پیش‌بینی می‌شود که در پایان تابستان، نمایشگاهی از پروژه‌های انجام‌شده توسط دانش‌جویان در طول تابستان برگزار شود.

۱ - sharecode.io
۲ - summerofcode.ir
۳ - sandcastle.ir

همایش آشنایی با رشته‌های مهندسی و دانشکده‌ی فنی

امین ادراکی
a.edraki.g@gmail.com

دانشگاه تهران امسال نیز همانند سال‌های گذشته با هدف آشنایی دانش‌آموزان و شرکت‌کنندگان آزمون سراسری با رشته‌های مهندسی و امکانات و فضاهای آموزشی دانشگاه تهران، توسط دفتر ارتباط با مدارس دانشکده‌ی فنی برگزار شد. این همایش در تاریخ چهارشنبه ۷ مرداد ۱۳۹۴ از ساعت ۸:۳۰ تا ۱۹ در دو بخش برگزار گردید که نزدیک به ۲۰۰ نفر از شرکت‌کنندگان آزمون سراسری و برخی از اولیای آن‌ها در آن شرکت کردند.

در بخش نخست - که در سالن آمفی‌تئاتر دانشکده‌ی مهندسی معدن برگزار شد - اساتید حاضر به معرفی رشته‌های مختلف مهندسی و امکانات آموزشی و پژوهشی دانشکده‌ی فنی پرداختند. در بخش دوم برنامه و پس از صرف ناهار، بازدید از تعدادی از آزمایشگاه‌های پژوهشی دانشکده‌های مهندسی مکانیک، مهندسی مواد و مهندسی معدن انجام شد. دانشکده‌ی مهندسی برق و کامپیوتر نیز برنامه‌ی ویژه‌ای را شامل بازدید از برخی از آزمایشگاه‌های پژوهشی و مشاوره‌ی انتخاب رشته با حضور دانشجویان و اساتید برجسته در گرایش‌های مختلف در همین بخش برگزار کرد.

لازم به ذکر است که اطلاع‌رسانی ضعیف و دیرهنگام و همچنین کیفیت نامناسب برگزاری برنامه در سال‌های گذشته از عمده‌ترین دلایل استقبال نسبتاً ضعیف از این برنامه بود و همان‌گونه که در بخش قبل اشاره شد، تنها کمتر از ۲۰۰ نفر از شرکت‌کنندگان کنکور سراسری در این برنامه حضور یافتند. همچنین بروز بی‌نظمی‌های متعدد در طول برگزاری برنامه و سخنرانی‌های پیاپی و بعضاً کسل‌کننده در طول بخش اول برنامه موجب نارضایتی برخی از شرکت‌کنندگان شد. امید است با اتخاذ تصمیمات مناسب در سطوح مختلف مدیریتی دانشکده‌ی فنی و کمیته‌ی ارتباط با مدارس این دانشکده، شاهد بهبود شرایط در سال‌های آتی باشیم.

نقش‌تان را انتخاب کرده‌اید؟

نمایشنامه‌ی «انتخابی برای مسیر پیش رو»

چه زمانی و چگونه وارد بازار کار شوم؟ چگونه حوزه‌ی مورد علاقه‌ام را پیدا کنم؟ تفاوت دوره‌های کارشناسی ارشد و دکترا چیست؟ چرا کارشناسی ارشد بخوانم؟ و بسیاری سؤالات دیگر که دیر یا زود با آن‌ها مواجه خواهیم شد و برای تعیین مسیر زندگی‌مان باید به آن‌ها توجه کنیم، مسائلی که هر فرد با شناخت خود و محیط اطراف به آن‌ها پاسخ می‌دهد. در این بخش از فصلنامه‌ی اف‌یک تلاش شده است با ارائه‌ی گزارش‌هایی از نشست‌ها و مصاحبه‌هایی که پیرامون این مسائل انجام شده است، به تعدادی از این سؤالات پاسخ داده شود و ابعاد مختلف موضوعاتی همچون مسائل ورود به بازار کار، تحصیل در خارج از کشور و دوره‌های کارشناسی ارشد و دکترا مورد بحث و بررسی قرار گیرند.

پرده‌ی اول:
انتخاب یک دانش‌جو
مصاحبه با
مهسا شهشهانی

شبنم بهنام
sh_behnam20@yahoo.com

چرا تصمیم به خواندن کارشناسی ارشد در داخل ایران گرفتید و چرا به‌سمت بازار کار و یا اپلای کشیده نشدید؟

در خصوص بازار کار، یک تابستان برای کارآموزی در یک شرکت مشغول به کار شدم و در تابستان قبل از آن نیز در یک شرکت دیگر کار می‌کردم و کار واقعی را در شرکت ها تجربه کردم که با علایق من سازگار نبود و در نتیجه تصمیم گرفتم کار آکادمیک را نیز تجربه کنم.

با توجه به این‌که در دوره‌ی کارشناسی معمولاً کار آکادمیک چندانی انجام نمی‌شود، تصمیم گرفتم ارشد بخوانم که متوجه شوم به کار آکادمیک علاقه‌مندم یا نه و در نتیجه الان نمی‌دانم که می‌خواهم در مقطع دکترا ادامه دهم یا نه و فعلاً قصد تجربه‌ی کار آکادمیک را دارم.

در خصوص دلیل اپلای‌نکردن که تقریباً برای تمام دانش‌جویان دانشکده، چه آن‌هایی که در نهایت اپلای می‌کنند و چه آن‌هایی که اپلای نمی‌کنند، باید بگویم که این مسئله‌ی بسیار بزرگی است که برای هر شخص چندین بعد دارد که هرکس آن‌ها را کنار هم می‌چیند و با توجه به معیارهای زندگی‌اش به آن‌ها امتیاز می‌دهد و تصمیم‌گیری می‌کند. من نیز همین کار را کردم و تصمیمم در نهایت این شد که فعلاً اپلای نکنم. شاید یک زمانی تصمیم بگیرم برای مقاطع بعد اپلای کنم، اما فعلاً چنین قصدی ندارم.

در مورد این انتخاب‌ها یک چیز را اضافه کنم! این‌که هر کس به من می‌رسید می‌پرسید: «استریت[1] شدی یا اپلای می‌کنی؟» یعنی اصلاً گزینه‌ی شرکت در کنکور ارشد و یا کارکردن مطرح نبود! که این جو انتخاب‌های شخصی را تحت تأثیر قرار می‌دهد.

تصمیم‌تان در خصوص ادامه‌ی تحصیل در مقطع کارشناسی ارشد به‌دلیل این بوده که همه‌ی دانش‌جویان کارشناسی ارشد می‌خوانند یا تصمیم شخصی خودتان بوده است؟

نه، کاملاً تصمیم خودم بوده است و خیلی به این مسئله فکر کردم که اصلاً قصد ادامه‌ی تحصیل دارم یا نه؛ و چون فکر می‌کردم که احتمالاً از کار آکادمیک و تحقیقاتی بیشتر خوشم می‌آید، تصمیم به ادامه‌ی تحصیل گرفتم.

در خصوص رتبه‌تان در رشته‌ای که امتحان داده‌اید و شرایط لازم برای قبولی در دانشگاه‌های خوب در مقطع کارشناسی ارشد توضیح دهید.

من برای گرایش هوش مصنوعی درس می‌خواندم و قصد ادامه‌ی تحصیل در این گرایش را داشتم. اما در نهایت تصمیم گرفتم گرایش نرم‌افزار را انتخاب کنم. البته تفاوت چندانی بین این دو وجود ندارد و تنها در چند درس اختصاصی متفاوتند. در واقع دست دانش‌جویان گرایش نرم‌افزار برای انتخاب زمینه‌ی کاری و استاد بازتر از دانش‌جویان هوش مصنوعی است.

رتبه‌ی کنکور نرم‌افزار من ۱۹، هوش مصنوعی ۳۹ و شبکه ۲۹ بود. فکر می‌کنم در سال‌های قبل تا حدود رتبه‌ی ۴۰ در دانشگاه تهران پذیرفته می‌شد.

گرایش کارشناسی ارشد خود را چگونه انتخاب کردید؟ راه شناخت گرایش‌های مختلف چیست؟ در رشته‌ی کامپیوتر چه گرایش‌هایی وجود دارد؟

کامپیوتر تا سال قبل شامل گرایش‌های نرم افزار، هوش مصنوعی و سخت افزار (معماری کامپیوتر) بود که البته امسال قرار است شبکه و امنیت شبکه نیز از کنکور فناوری اطلاعات به کامپیوتر منتقل شود. در خصوص انتخاب من، در ابتدا قصد داشتم کنکور هوش مصنوعی بدهم اما بعد متوجه شدم که آزمایشگاه سیستم‌های هوشمند اطلاعات دانشگاه ما (به‌سرپرستی دکتر شاکری) که در زمینه‌های بازیابی هوشمند اطلاعات و داده‌کاوی فعالیت می‌کند فقط از نرم‌افزار و فناوری اطلاعات دانشجو می‌پذیرد و نه از هوش مصنوعی. به‌همین دلیل تصمیم گرفتم نرم‌افزار بخوانم.

در خصوص شناخت گرایش‌ها، دانش‌جویان می‌توانند به وبسایت دانشگاه مراجعه کنند و اطلاعات آزمایشگاه‌های مختلف را ببینند و دانش‌جویان و فعالیت‌های آن گرایش و پایان‌نامه‌ها و موضوعات آن‌ها را مشاهده کنند و حتی از عنوان آن‌ها نیز می‌توانند اطلاعاتی را کسب کنند. به‌علاوه می‌توان با دانش‌جویان و اساتید آن رشته نیز صحبت کرد.

از چه زمانی دانش‌جویان باید شروع به خواندن دروس برای کنکور کارشناسی ارشد کنند؟

برای دانش‌جویانی که در دانشگاه‌های خوب درس خوانده‌اند و دروس آن‌ها با کیفیت بسیار بالایی ارائه شده است، کنکور ارشد سخت نیست و بسیاری از دانش‌جویان به‌دلیل آن‌که دروس را در طول ترم خوب خوانده‌اند، بدون این که مشخصاً درسی را برای کنکور بخوانند در کنکور شرکت کرده و رتبه‌های خوبی را نیز کسب می‌کنند.

اما برای افرادی که دروس را معمولی‌تر خوانده‌اند یا در طول چند ترم کمتر به دروس اهمیت داده‌اند، به‌نظر من کنکور بیش از دو ماه فشرده وقت نمی‌گیرد.

۱ - Straight

پرده‌ی دوم:
نظرات یک استاد
گفت‌وگویی با دکتر رامتین خسروی

دانش‌جویانی که هنوز فارغ‌التحصیل نشده‌اند معمولاً این دو ماه فشرده را فرصت ندارند. به‌طور مثال، من از مهر شروع کردم و با وجود این‌که درس‌هایم سبک بود، چندین بار خواندن را رها کردم. اما اصل درس‌خواندن من در فرصت قبل و بعد از امتحانات بود. یعنی به‌طور فشرده چیزی حدود شش هفته. (البته این صحبت‌ها فقط در مورد کنکور کامپیوتر است و کنکور رشته‌ی برق داستان کاملاً متفاوتی دارد.)

به‌نظر شما برای کنکور لازم است در کلاسی نیز شرکت کرد یا دروس دانشگاه کافی است؟

نه، به نظر من اصلاً کلاسی نیاز نیست و دروس دانشگاه و مطالعه‌ی چند کتاب کاملاً کفایت می‌کند.

در مورد تأثیر معدل در کنکور کارشناسی ارشد توضیح دهید.

تأثیر معدل به این صورت است که ۸۰٪ نمره‌ی کل، نمره‌ی علمی آزمون ارشد و ۲۰٪ مابقی معدل دوره‌ی کارشناسی است. البته این تأثیر به این صورت است که اگر معدل خیلی بالا باشد، تأثیر مثبتی دارد و اگر معدل پایین باشد، تأثیر منفی زیادی نمی‌گذارد و این تأثیر منفی نیز تقریباً زمانی است که معدل کمتر از ۱۳ باشد و معدل ۱۴ به بالا معدل بدی محسوب نمی‌شود.

همچنین لازم به توضیح است که ارزش معدل در دانشگاه‌های مختلف یکسان نیست و از نتایج به‌نظر می‌آید که در این خصوص دانشگاه تهران بیشترین ضریب را بین دانشگاه‌ها دارد.

درس‌های تأثیرگذار در کنکور که نیازمند توجه در طول ترم است کدام‌اند؟

سیستم عامل، ساختمان داده، طراحی الگوریتم و معماری کامپیوتر (البته معماری کامپیوتر در کنکور چندان به معماری کامپیوتری که در دانشگاه ما ارائه می‌شود شبیه نیست).

برای چند رشته می‌توان در کارشناسی ارشد کنکور داد؟

می‌توان در یک رشته‌ی اصلی و یک رشته‌ی شناور امتحان داد. به‌طور مثال، نمی‌توان در رشته‌ی مهندسی کامپیوتر و برق امتحان داد، اما می‌توان در رشته‌ی مهندسی کامپیوتر و فناوری اطلاعات امتحان داد. زیرا فناوری اطلاعات یک رشته‌ی شناور است.

چرا تعداد زیادی از افراد، دوره‌های مدیریت اجرایی و بازاریابی را انتخاب می‌کنند؟

به‌نظر من دانش‌جویان پس از خواندن رشته‌ی مهندسی سراغ رشته‌های مدیریتی می‌روند تا بتوانند در رشته‌ی خودشان یک تیم مهندسی را مدیریت کنند.

علی شیراوند
alish.icu@gmail.com

گرایش‌های کارشناسی ارشد را در رشته‌ی کامپیوتر چه‌طور می‌توان شناخت؟

در رشته‌ی کامپیوتر، سبک دانشگاه ما این‌گونه است که در زمان انتخاب رشته، گرایش تعیین نمی‌شود و دانش‌جو می‌تواند پس از پذیرش و انتخاب استاد موضوع را انتخاب نماید. دانش‌جویان می‌توانند گرایش‌هایی مثل هوش ماشین و رباتیک را در درس‌هایی که در دوره‌ی کارشناسی ارائه می‌شود یاد بگیرند. اساتید فعال در هر زمینه نیز به‌خوبی استقبال می‌کنند که با دانش‌جویان کارشناسی در آزمایشگاه‌ها همکاری کنند و دانش‌جویان نیز می‌توانند در این فرصت با حوزه‌ی مورد نظرشان آشنا شوند. مصوبات وزارت علوم برای برخی از دانشگاه‌ها - مانند دانشگاه تهران - که هیئت‌امنا دارند لازم‌الإجرا نیست. برنامه‌ی وزارت علوم بیشتر برای دانشگاه‌های کوچک‌تر و جدیدتر مورد استفاده قرار می‌گیرد. در دانشگاه ما دانش‌جویان با توجه به زمینه‌ای که به آن علاقه دارند اساتید مورد نظرشان را انتخاب می‌کنند و درس‌هایشان را طبق علاقه‌ی خود می‌گذرانند.

چه‌طور می‌توان از راه‌هایی به‌جز کنکور وارد دوره‌ی کارشناسی ارشد شد؟

به‌جز کنکور، سهمیه‌ی استعدادهای درخشان وجود دارد که دانش‌جویان اصطلاحاً به آن استریت[1] می‌گویند و قاعده‌ی آن هر سال عوض شده است. به‌صورت کلی ۱۰٪ تا ۲۰٪ برتر هر رشته در دانشگاه پذیرفته می‌شوند ولی شرایطی نیز در وزارت علوم وجود دارد که طبق آن دانش‌جویانی که مقاله داشته باشند یا برخی شرایط دیگر را داشته باشند می‌توانند به‌عنوان استعداد درخشان درخواست کنند، منتها نه به‌عنوان آن ۱۰٪ دانش‌جویانی که در المپیاد دانش‌جویی رتبه کسب می‌کنند نیز می‌توانند وارد دوره‌ی ارشد شوند.

کسانی که دوره‌ی دکترا را می‌گذرانند چه فعالیت‌های دیگری به‌جز فعالیت در دانشگاه و آزمایشگاه‌ها می‌توانند انجام دهند؟

این افراد می‌توانند علاوه بر این مکان‌ها، در مراکز تحقیقاتی نیز کار کنند. متأسفانه در کشور ما صنایع آن‌قدر پیشرفته نیستند که ما مراکز تحقیقاتی همانند سایر شرکت‌های بزرگ دنیا مثل IBM یا گوگل داشته باشیم. بنابراین اگر کسی بخواهد خارج از این محیط‌ها کار کند مکان‌های محدودی مانند مرکز تحقیقات مخابرات و مرکز تحقیقات فیزیک نظری و ریاضیات و امثال این‌ها وجود دارد؛ اما برخی بعد از این‌که دوره‌ی دکترای خود را به پایان می‌رسانند صرفاً از عنوان دکترای خود برای ورود به محیط‌هایی که به عنوان مدرک بها می‌دهند و گرفتن پروژه‌های صنعتی استفاده می‌کنند. حتی برخی از افراد شرکت تأسیس می‌کنند، کارهای تحقیقاتی خود را ادامه نمی‌دهند و در واقع از آن مسیر اصلی خارج می‌شوند. به‌نظر من این افراد عمر خود را تلف کرده‌اند اما متأسفانه در کشور ما برای برخی فعالیت‌ها به این عنوان نیاز است. اگر کسی می‌خواهد این مسیر را انتخاب کند، من توصیه می‌کنم از جایی که راحت‌تر بتواند مدرک خود را دریافت کند این راه را ادامه دهد.

۱ - Straight

پرده‌ی سوم: روایت یک نشست

تیم اف‌یک در تابستان امسال نشستی را با هدف بررسی راه‌های پیش روی دانش‌جویان پس از اتمام دوره‌ی کارشناسی برگزار کرد. این نشست از افرادی با تجربه‌های متفاوت تحصیلی و کاری تشکیل شده بود که هر کدام با توضیحات خود سعی در ارائه‌ی تصویر واضحی از نیازمندی‌ها و فواید هر یک از این راه‌ها داشتند. از این رو این گزارش می‌تواند پاسخگوی سؤالات طیف گسترده‌ای از دانش‌جویان باشد.

ارشد

آیا حتما باید مقطع کارشناسی ارشد را بگذرانیم؟ چه اطلاعاتی در این دوره به ما اضافه می‌شود؟ چه کسانی نباید کارشناسی ارشد بخوانند؟ آیا این درست است که افراد با این مدرک حقوق بالاتری خواهند داشت؟

احسان خامس‌پناه

دانش‌آموخته‌ی کارشناسی مهندسی کامپیوتر(نرم‌افزار) در دانشگاه تهران،
کارشناسی ارشد مهندسی کامپیوتر (نرم‌افزار) در دانشگاه امیرکبیر؛
دانش‌جوی دکترای مهندسی کامپیوتر (نرم‌افزار) در دانشگاه تهران؛
استاد دانشگاه تهران.

امیر صدیقی

دانش آموخته ی کارشناسی ارشد مهندسی کامپیوتر (نرم افزار) در دانشگاه شهید بهشتی؛
متخصص در حوزه‌های داده‌های بزرگ، یادگیری ماشین و پردازش جریان داده؛
مدیر اجرایی و مدیر فنی در شرکت رایانش دادگان اکباتان.

سیدشمیم طاهری

دانش‌آموخته‌ی کارشناسی مهندسی کامپیوتر (سخت‌افزار) در دانشگاه تهران،
کارشناسی ارشد اقتصاد در دانشگاه صنعتی شریف،
کارشناسی ارشد اقتصاد در مدرسه‌ی اقتصاد لندن؛
دانش‌جوی مقطع دکترا اقتصاد در London Business School.

سمانه کریمی

دانش‌آموخته‌ی کارشناسی مهندسی کامپیوتر (نرم‌افزار) در دانشگاه تهران،
کارشناسی ارشد مهندسی کامپیوتر (نرم‌افزار) در دانشگاه تهران؛
دانش‌جوی دکترای مهندسی کامپیوتر (نرم‌افزار) در دانشگاه تهران.

محمدجواد ایزدی

دانش‌آموخته‌ی کارشناسی مهندسی کامپیوتر (نرم‌افزار) در دانشگاه تهران،
کارشناسی ارشد مهندسی کامپیوتر (نرم‌افزار) در دانشگاه تهران؛
مشغول به کار در شرکت داتین (مربوط به نرم‌افزارهای بانکی).

امیر صدیقی:

من بعد از ۱۵ سال دوره‌ی ارشد را با رهاکردن کاری با حقوق بسیار خوب شروع کردم، چون دوره‌ی کارشناسی ارشد با دوره‌ی کارشناسی بسیار متفاوت است؛ دوره‌ای خیلی کوتاه با دانش و اطلاعات خیلی زیاد. تکنیک‌ها و دانشی که در دوره‌ی ارشد به‌دست می‌آورید بسیار مفید و کاربردی است. این دوره از نظر رزومه هم بسیار باارزش است. به‌نظر من بهتر است بین دوره‌ی کارشناسی و کارشناسی ارشد فاصله باشد. علاوه بر آن، افرادی که در دانشگاه با آن‌ها روبه‌رو می‌شوید انسان‌های بسیار سالمی هستند که در فضای کار کمتر چنین افرادی را می‌بینید. لذا شبکه‌ای از دوره‌ی ارشد به‌دست می‌آید که بسیار ارزشمند است.

باید بگویم افرادی هستند که علاقه‌ای ندارند و صرفاً برای شخصیت اجتماعی مثلاً برای این‌که در پروفایل لینکداین خود، کلمه‌ی Master بنویسند به دانشگاه می‌آیند. اگر قرار است چیزهایی را که به شما یاد می‌دهند در دنیای واقعی به‌کار نگیرید، می‌توانید از طریق آموزش‌های آنلاین (مثل کورسرا) آن‌ها را یاد بگیرید. خود دانشگاه ارزش دارد، انسانیت داخل دانشگاه بسیار ارزشمند است. زمان خود را صرف کاری که دوست ندارید نکنید، ارزش زمان زندگی بیشتر از این‌ها است.

احسان خامس‌پناه:

در دو جهت آکادمیک و کار بررسی می‌کنم. از نظر آکادمیک مشخص است که ارشد می‌خوانید که ادامه‌ی تحصیل بدهید. دوره‌ی کوتاهی که می‌توانید پژوهش کنید و ببینید که به‌درد این‌کارها می‌خورید یا نه! استاد دانشگاه یا دانشمند شوید یا خیر. اگر این ایده را دارید در یک جای امن و در یک دوره‌ی دوساله می‌توانید آن را تست کنید. و اگر هم متوجه شوید که دوست ندارید خیلی زمان زیادی را از دست نداده‌اید.

از نظر کارکردن باید بگویم حتی اگر پشت سر هم

این دو دوره را بخوانید، این کار ارزش بسیار زیادی دارد، زیرا درس‌هایی در کارشناسی ارشد هستند که به شما دید بهتری می‌دهند که کاملاً در کار به شما کمک می‌کند. شاید تنها ۱۰٪ از اطلاعات مورد نیاز کاری که از آن درآمد کسب می‌شود از دوره‌ی دکترا باشد. اگر زمانی در موقعیت مصاحبه‌کننده باشید، می‌بینید که نوع ادبیات آدم‌ها با توجه به سطح علمی افراد متفاوت است. از طرف دیگر، با وجود این‌که که دوره‌ی ارشد برای ایجاد شبکه بسیار مهم است، اما نباید در دوره‌ی ارشد به‌دنبال ساختن شبکه باشید، بلکه باید طوری درست و خوب کار کنید که شبکه ساخته‌شود.

ارشد را همه می‌توانند بخوانند. به‌نظر من هم بخوانند ولی تکلیف خود را قبل از شروع دوره روشن کنید. اگر صرفاً برای گرفتن مدرک دوره را نمی‌خواهید و می‌خواهید واقعاً ادامه‌ی تحصیل دهید، خوب درس بخوانید، رتبه‌ی خوب بیاورید و ادامه‌ی تحصیل دهید. در غیر این صورت از «یک دانشگاه همین‌جوری» مدرک خود را بگیرید و در یک حد معقول درس بخوانید.

شمیم طاهری:

کارشناسی ارشد حد فاصل دوره‌های کارشناسی و دکترا است؛ تعدادی درس با جزئیات بیشتر و عمیق‌تر و پژوهش در دو ترم آخر. فارغ‌التحصیل‌های دوره‌ی کارشناسی ارشد موقعیت‌های شغلی خیلی بهتری از کارشناسی و حتی دکترا دارند. حتی تعداد زیادی از فارغ‌التحصیلان دکترا نیز در کار خود بیشتر از دانشی استفاده می‌کنند که در دوره‌ی کارشناسی ارشد به‌دست آورده‌اند و نه دوره‌ی دکترا.

تقریباً هرکسی بعد از ترم شش یا هفت کارشناسی می‌تواند به این نتیجه برسد که به رشته‌ی خود علاقه دارد یا خیر. بعضی از افراد می‌توانند خیلی واضح و شفاف به این نتیجه برسند که به درد رشته‌شان نمی‌خورند و بهتر است برای ادامه‌ی تحصیل رشته‌ی فعلی خود را تغییر دهند. من اگر برای کارشناسی ارشد تغییر رشته نمی‌دادم هیچ‌گاه نمی‌توانستم به این فکر کنم که در یکی از گرایش‌های کامپیوتر ادامه‌ی تحصیل بدهم.

سمانه کریمی:

اگر افرادی را که از قبل هدف خود را می‌دانند و مطمئن هستند که می‌خواهند چه کاری انجام دهند از بحث جدا کنیم، شاید یک ایده این باشد که از اتلاف وقت جلوگیری کنیم و کارشناسی را چهارساله و ارشد را دوساله تمام کنیم. ولی تجربه‌ی شخصی من نشان می‌دهد که اشکالی ندارد اگر بگذارید فاصله‌ای بین دوره‌های مختلف وجود داشته باشد و اگر این فاصله برایتان پیش آمد ناراحت نباشید. در این فاصله می‌توانید کارهایی را که تجربه نکرده‌اید بسنجید و در نهایت بهتر تصمیم‌گیری کنید. در واقع دوره‌ی ارشد آزمونی است که با کمک آن تصمیم بگیریم به درد دوره‌ی دکترا می‌خوریم یا خیر. من شخصاً از مسیری که طی کرده‌ام راضی هستم، ولی بدون فاصله پیش‌رفتن می‌تواند برای بعضی از افراد باعث تحلیل رفتن انرژی آنها شود. معمولاً افرادی که پس از یک دوره‌ی کار وارد دوره‌ی ارشد می‌شوند اشتیاق و علاقه‌ی بیشتری به تحقیق و یادگرفتن دانش‌های متفاوت دارند.

محمدجواد ایزدی:

ارشد پایه‌ی حقوق بیشتری دارد! عبارت درستی است اما پس از گذشت مدت زمان یک یا دو سال، اثر مستقیم آن میرا می‌شود. البته از تأثیر و دیدی که دوره‌ی ارشد به شما می‌دهد نمی‌توان گذشت، اما این موضوع که صرفاً به‌خاطر مدرک میزان حقوق شما تعیین شود - مخصوصاً در رشته‌ی کامپیوتر - خیلی زود از بین می‌رود.

این‌که ممکن است ارشد باعث شود از فضای کاری دور شویم مقداری درست است، اما کیفیت آن به‌شدت به افراد، دانشگاه و استاد راهنما بستگی دارد. در دوره‌ی کارشناسی ارتباط بین میزان تلاش و موفقیت بسیار نزدیک است، اما بوده‌اند کسانی که در دوره‌ی ارشد با تلاش خیلی کم به موفقیت‌های زیادی رسیده‌اند و برعکس. نمی‌توان گفت کسی که در ارشد کار می‌کند حتماً به درسش لطمه می‌خورد. من قبل از خواندن ارشد تصمیم داشتم که دکترا هم بخوانم، در حالی که در دوره‌ی ارشد متوجه شدم من به‌درد این کار نمی‌خورم.

افرادی هستند که وقتی وارد کاری می‌شوند از همان ابتدا تا انتها سود ثابتی دارند. اما عده‌ای دیگر به مرور زمان تأثیر حضور و سودشان بیشتر احساس می‌شود. این همان تأثیر دانش است، مانند الگوریتم، طراحی و تأثیر همروندی و امثال این‌ها که جاافتادنِ آن فقط در محیط دانشگاهی امکان‌پذیر است. من اخیراً به این نتیجه رسیده‌ام که واقعاً هر چیزی یادگرفتنی است و دانش‌بنیان‌بودن بسیار مهم‌تر از داشتن تجربه‌ی بی‌دانش است.

دکترا

چه کسانی باید دکترا بخوانند؟ خواندن دکترا چه خوبی‌ها و بدی‌هایی می‌تواند داشته باشد؟

شمیم طاهری :

من صلاحیت ندارم که در مورد دوره‌ی دکترا در رشته‌ی مهندسی کامپیوتر صحبت کنم، اما به‌طور کلی اگر کسی قصد گذراندن این دوره را در رشته‌های زیرمجموعه‌ی علوم اجتماعی دارد لازم است شخصیتی مناسب برای تحقیقات بلند مدت - گاهی شب تا صبح - و انجام‌دادن کارهای بعضاً یکنواخت به‌مدت طولانی داشته باشد؛ ضمن این‌که خواندن دکترا، مستقل از رشته، ممکن است چند سال ورود شما به بازار کار را به تعویق بیندازد.کسانی که برای بازار کار نیرو جذب می‌کنند ترجیح می‌دهند از فارغ‌التحصیلان دوره‌ی کارشناسی ارشد استفاده کنند تا حقوق نسبتاً پایین‌تری بدهند. شخصاً این مورد را در خارج از کشور زیاد دیده‌ام. و البته چیزی که اهمیت دارد این است که اگر کسی با آگاهی وارد دوره‌ی دکترا شده باشد، قاعدتاً نوع کاری که در آینده خواهد کرد - مستقل از حقوقش - وی را راضی می‌کند. همان‌طور که کسی که پس از دوره‌ی ارشد وارد بازار کار میشود از کارش رضایت دارد، فردی که دکترا می‌خواند نیز از کارش لذت خواهد برد؛ هر کس را باید در جایگاه خودش سنجید.

برخلاف دانش‌جوهایی که در ایران دکترا می‌خوانند، تعداد زیادی از کسانی که برای دکترا از کشور خارج می‌شوند برای کاری که انجام می‌دهند از دانشگاه پول می‌گیرند و این موضوع به‌جهت این‌که فرد دیگر دغدغه‌ی کارکردن و کسب درآمد نخواهد داشت که به درسش لطمه بزند بسیار اهمیت دارد.

به‌عنوان جمع‌بندی نکته‌ای هست که دو-سه سال است به آن رسیده‌ام؛ چیزی که شاید اگر قبلاً کسی به من می‌گفت فکر می‌کردم شعار می‌دهد. آن نکته این است که در زندگی خوشحال‌بودن از همه‌چیز مهم‌تر است. شما باید کاری را انجام دهید که در لحظه خوشحال‌تان کند؛ زندگی مسابقه نیست که مدام خودتان را با بقیه مقایسه و فکر کنید که موقعیت‌تان در مقایسه با دیگران شایسته‌ی شما نیست.

احسان خامس‌پناه:

در این مورد دکتر خسروی یک توصیه‌ی خوب به من کردند که سعی می‌کنم عین عبارتش را بگویم: «خامس... عاقل باش... دکترا نخون!» و در ادامه: «ببین، تو که شروع می‌کنی به دکتراخوندن، پول که نمی‌دن ، می‌گی که درسه دیگه، یه سال درس می‌خونم بعدش اگر نخواستم می‌رم توی بازار کار. بعد که اون تموم می‌شه می‌گی یه آزمون جامع رو هم

بدیم حالا که تا این‌جا اومدیم. بعد از آزمون جامع می‌گی که یه پروپوزال دفاع کردن هم که دیگه این حرفا رو نداره. پروپوزال رو دفاع می‌کنی یعنی تا نصف راه اومدی. باقیش رو هم می‌ری و پنج سال از زندگی‌ت رو فنا می‌کنی!»

هرکس در این مورد نظر خودش را دارد که شاید لزوماً درست نباشد. من از کسی شنیدم که می‌گفت: «افرادی که دکترا می‌خوانند بازنده‌های زندگی هستند چون هیچ کار دیگری از آن‌ها برنمی‌آید.» از یکی دیگر از دوستان هم پرسیدم: «تو که رفته بودی سر کار، چی شد؟» جواب داد: «من مدتی رفته بودم در یک شرکت و دیدم که راضی نیستم، تصمیم گرفتم دکترا بخوانم و بعدش شروع کنم به درس‌دادن.» این یکی از بهترین دلیل‌هایی بود که من برای دکتراخواندن شنیدم. جدا از این مسائل، آمار نشان می‌دهد که نه فقط در کشور ما (که هیچ پولی به دانش‌جوهای دکترا نمی‌دهند) بلکه در آمریکا نیز ۹۴ درصد از افراد در سال چهارم دوره‌ی دکترا از تحصیلات خود راضی نیستند. این نرخ بالا به این دلیل است که دانش‌جو چهار سال از وقتش را صرف دکتراگرفتن می‌کند و هنگامی که وارد بازار کار می‌شود از خیلی از دانشی که در این دوره کسب کرده است استفاده نمی‌کند. در این مرحله درآمدش به‌جای این‌که ۱۰۰۰ واحد باشد ۱۲۰۰ واحد می‌شود. در صورتی که اگر ۴ سال پیش کار را شروع می‌کرد حقوقش ۲۰۰۰ واحد بود و جایگاهش هم مشخص بود.

در واقع اهدافی که دوره‌ی دکترا دنبال می‌کند پژوهش و ایجاد آمادگی برای تربیت مهندس است. اگر این دو کار را دوست دارید دکترا بخوانید. از نظر من این مرحله از تحصیلات چندان سود دیگری برایتان ندارد. اما در کنار این‌ها تعدادی فرصت ایجاد می‌کند، شبکه‌ای جدید برایتان ایجاد می‌شود و الگوهای شغلی‌تان تغییر می‌کند. اما در نهایت نسبت سود به هزینه بسیار مهم است.

۵ یا ۴ سال دوره‌ای فوق‌العاده طولانی است و بیشتر چیزی که به شما یاد می‌دهد این است که چه‌طور مقاله بنویسید، چگونه خوب آن را ارائه کنید و چه‌طور خوب پژوهش کنید. اما من به‌شخصه دوست دارم مهندس تربیت کنم، دوست دارم تحقیقات هم در کنار کارم انجام بدهم و مسائل شخصی‌تر. به این دلیل دکترا خواندن برای من ارجحیت دارد و من حتی اگر از همه‌ی سختی‌های راه باخبر بودم باز هم به همین مسیر ادامه می‌دادم.

محمدجواد ایزدی:

من از وقتی وارد دانشگاه شدم متوجه شدم تدریس و تربیت‌کردن مهندس را دوست دارم. همیشه فکر می‌کردم بهترین راه این است که دکترا بخوانم و استاد دانشگاه بشوم. قبل از آمدن به دانشگاه هم چیزی درباره‌ی پژوهش نمی‌دانستم.

وارد دوره‌ی کارشناسی ارشد شدم و کم‌کم با پژوهش آشنا شدم. پژوهش در ابتدا مسئله‌ی جذابی است. پژوهشگر مسئله‌ای را که کسی تا به حال حل نکرده است حل می‌کند و انگار دردی از کسی دوا کرده است، اما من سه سال و نیم که کارشناسی ارشد خواندم تقریباً هیچ‌کاری نکردم و هر چه‌قدر که به دوره‌ی کارشناسی‌ام افتخار می‌کنم از دوره‌ی کارشناسی ارشدم ناراضی هستم. یکی از دلایل این موضوع این بود که من به درد پژوهش نمی‌خوردم. با این حال همیشه علاقه به تدریس‌کردن برایم وجود داشت. اما حالا، چیزهایی که دنبالش بودم در کاری که می‌کنم وجود دارد. من تقریبا یک-سوم از وقت کاری‌ام در شرکت را روی تدریس، رهبری و مربیگری نیروهای شرکت می‌گذرانم. یعنی به آدم‌هایی که مثل خودم هستند و فقط سه-چهار سال بعد از من وارد بازار کار یا دانشگاه شده‌اند چیزهایی که می‌دانم را منتقل می‌کنم و در کنار آن لذت درس‌دادن را هم تجربه می‌کنم. ضمن این‌که درس‌دادن در محیط کار به این دلیل که می‌شود نتیجه‌ی کار را دید و افراد از آن در حل‌کردن ایرادی از پروژه استفاده می‌کنند بسیار جذاب و شیرین است.

> «خامس... عاقل باش... دکترا نخون!»

احسان خامس‌پناه:

توصیه‌ای که در نهایت می‌توانم بکنم این است که سعی کنید در هر کاری که تصمیم می‌گیرید انجام دهید جدی باشید. یک زمانی هرکس به ما میگفت جدی باشیم و روی کار وقت بگذاریم ما می‌خندیدیم! اما این موضوع واقعاً تجربه شده است. سعی کنید از وقت خود در دوره‌ای که هستید درست استفاده کنید. امروزه راه موفقیت این شده که ارشد و دکترا بخوانیم، در حالی که وقت‌گذاشتن برای بالابردن سواد و یادگرفتن این‌که چگونه کار خوب انجام دهیم کمرنگ شده است و می‌تواند به‌مرور دانشگاه را از بین ببرد.

کار

آیا کار کردن در دوره‌ی تحصیل خوب است؟ چه نوع کاری می‌تواند مفید باشد؟ تجربه‌ی کسانی که استارت‌آپ راه‌اندازی کرده‌اند چگونه بوده است؟

امیر صدیقی:

من از ۱۵سالگی شروع به کار کردم؛ هم به درآمد آن احتیاج داشتم و هم بسیار به کارم علاقه‌مند بودم. درآمد خوبی داشتم و می‌توانستم مجلات مورد علاقه‌ی خود را بخرم. از همان سال‌ها تا به امروز، پروژه‌های مختلفی را در زمینه‌های متفاوت انجام داده‌ام. بعد از دوران تحصیل با بهترین شرکت‌های کامپیوتری کار کردم و خیلی حس خوبی داشتم که با آدم‌های معروف کار می‌کنم. اما این‌که ناخدای یک کشتی (یا حتی قایق) باشید و برای خودتان کار کنید، خیلی جذاب‌تر است. این ناخدابودن را ما مهندسین نرم‌افزار و در استارت‌آپ می‌توانیم تجربه کنیم. بعد از تجربه‌های کاری زیاد و دو استارت‌آپ موفق، اکنون به این نتیجه رسیده‌ام که باید بیش از این بیاموزم و نیاز به درس‌خواندن دارم. برای مثال باید در حوزه‌ی deep learning اطلاعات خود را افزایش دهم.

احسان خامس‌پناه:

اولین تجربه‌ی کاری من در ترم یک بود، باید برنامه‌ای می‌نوشتم که مقداری داده جمع‌آوری کند و آن را با کمک اطلاعاتی که از دبیرستان داشتم به‌وسیله‌ی زبان delphi نوشتم! مدتی هم در دبیرستان تدریس می‌کردم. در دوره‌ی دکترا در سه استارت‌آپ همکاری کرده‌ام که هیچ‌یک باقی نمانده‌است. کار کردن در دوره‌های ارشد و دکترا بسیار مشکل است.

بعضی از دانش‌جویان از سال‌های دوم و سوم به‌صورت نیمه‌وقت کار می‌کردند ولی من هیچ‌وقت این کار را نکردم و هرچه گذشت بیشتر به درستی کارم پی بردم. با این حال این حرف خیلی به دانشگاه بستگی دارد. شاید در دانشگاه‌های دیگر نتوان این حرف را زد چون فردی که آموزش می‌دهد ممکن است بیشتر وقت افراد را تلف کند. اما در این دانشگاه چنین اتفاقی نمی‌افتد. البته من با زاویه‌ی فکری شما که تقریباً فقط ۴-۵ درس را مفید می‌دانید هم مخالف هستم. به جز درس فیزیک-مکانیک تقریبا تمام درس‌هایی که در دانشگاه خوانده‌ام در کار تأثیر مثبتی داشته‌اند.

اگر کاری باشد که پاره‌وقت نباشد و نخواهد ۲-۳ روز در هفته وقت شما را بگیرد و دغدغه‌ی ذهنی شما شود مفید است. (کار کردن با پول درآوردن کاملاً متفاوت است.) توصیه‌ی من به شما این است که در طول دوره‌ی کارشناسی به‌هیچ‌وجه درگیر کار جدی نشوید حتی اگر درآمد زیادی داشته‌باشد. در دوره‌ی دکترا زیاد کار کردم، از جمله سه استارت‌آپ که هیچ‌یک باقی نمانده است. کارکردن در دوره‌های ارشد و دکترا هم بسیار سخت است.

شمیم طاهری:

به‌هیچ‌عنوان توصیه نمی‌کنم که در دوره‌ی کارشناسی کار کنید. مطمئن باشید که خوب درس‌خواندن در دوره‌ی کارشناسی زمان زیادی از شما می‌گیرد. وقت خود را روی درس بگذارید و حتی سال اول کارشناسی ارشد هم کار نکنید، چون درس‌های سال اول تمرکز زیادی می‌خواهد. اگر به هر دلیلی تصمیم به کارکردن گرفتید، توصیه می‌کنم از ترم سوم کارشناسی ارشد شروع به کار کنید؛ تجربه‌ی بسیار خوبی خواهد بود و به تصمیم‌گیری شما برای ادامه‌ی تحصیل یا انتخاب نوع کار آینده هم کمک می‌کند. دوره‌ی کارآموزی در دوره‌های کارشناسی و کارشناسی ارشد بسیار فرصت خوبی است. می‌توانید برای کارآموزی به کشورهای دیگر بروید و محیط‌های دیگر را مشاهده کنید. به‌خصوص در دوره‌ی کارشناسی ارشد، یک دوره‌ی کارآموزی خوب می‌تواند به شناخت خودتان خیلی کمک کند.

سمانه کریمی:

مهم این است که وقت خود را صرف کاری کنید که دوست دارید، و در این میان سعی کنید تخصص خود را بالا ببرید. اگر کارکردن به‌معنای تجارتی است که محصولی تولید کنید یا پولی کسب کنید می‌تواند تجربه‌ی خوبی باشد، اما من نیز معتقدم که بازه‌ی کارشناسی مناسب نیست. هرچند جنبه‌ی منفی و مثبت دارد ولی به‌نظرم جنبه‌ی منفی‌آن بیشتر است. اگر فکر می‌کنید کار تحقیقاتی را بیشتر دوست دارید، به این کار بپردازید. در هر حوزه‌ای کار می‌کنید سعی کنید به‌صورت هم‌زمان

ابزارهای آن حوزه را یاد بگیرید، نرم‌افزار توسعه بدهید، مقاله بخوانید و به‌معنای واقعی از آن زمان استفاده کنید.

محمدجواد ایزدی:

شما باید یک سری دانش و یک سری مهارت کسب کنید. بخش دانش را دانشگاه تا حد خوبی برای شما فراهم می‌کند. من به‌شدت از دانش دوره‌ی لیسانسی که اینجا گذارنده‌ام استفاده می‌کنم. در مقابل به شدت به شما توصیه می‌کنم در فعالیت‌های اجرایی فوق برنامه و خارج از برنامه‌ی درسی و همچنین دستیار آموزشی شدن مشارکت کنید. من خودم در همین شاخه‌ی دانش‌جویی ACM فعالیت کرده‌ام. همچنین در محیط کار باید تعدادی قابلیت فنی داشته باشید و تعدادی قابلیت غیرفنی. بخش توانایی‌های غیرفنی را با همین فعالیت‌های فوق برنامه در دانشگاه می‌توانید به دست‌آورید. در محیط کاری اشتباه‌کردن خیلی هزینه دارد، اما در دانشگاه این هزینه کم است.

زیاد شنیده می‌شود که چیزهایی که در دانشگاه یاد گرفته‌ایم هیچ ربطی به چیزی که در کار به آن نیاز داریم ندارد. این مشکل در نظام آموزشی کشور ما وجود دارد، اما این فاصله در نرم‌افزار و در دانشگاه ما بسیار کم است. با تجربه‌ای که با مشاهده‌ی افراد زیادی در طول سالیان دارم هیچ ارتباط مشخصی بین کارکردن یا نکردن در دوره‌ی کارشناسی و کسب موفقیت پیدا نکرده‌ام. البته توجه کنید که کار را می‌شود تغییر داد، اما تکرار کردن دانشگاه به شدت سخت‌تر است.

با توجه به موجی که در مورد استارت‌آپ وجود دارد، باز هم توصیه می‌کنم مطالعه کنید. فضای استارت‌آپ به شدت جذاب‌تر از فضای سنتی است. اما بدانید شکستْ جزئی از استارت‌آپ است.

اپلای

آیا کسی که شرایط اپلای کردن را دارد باید حتماً این کار را انجام دهد؟ زندگی و تحصیل در خارج از ایران چگونه است؟ اپلای کردن چه مزایا و معایبی می‌تواند داشته باشد؟

محمدجواد ایزدی:

به‌نظر من اپلای‌کردن نباید هیچ‌گاه هدف باشد.

من به این نتیجه رسیدم که موفق‌بودن من نتیجه‌ی ایران‌بودن یا نبودن من نیست. هم آدم‌هایی را دیده‌ام که در ایران موفق بودند و هم خارج از ایران، اگر بخواهم واقع‌بینانه نگاه کنم، به‌نظر من اپلای‌کردن یا نکردن نمی‌تواند تنها عامل موفقیت یا عدم موفقیت باشد.

در مسیری که من طی کردم، احساس نمی‌کنم برای من فرصتی خارج از کشور وجود داشته است که در ایران وجود نداشته‌باشد. اپلای نکردم چون در زمان تصمیم‌گیری‌ها و انتخاب‌هایم، این مسیر از اپلای کردن رد نمی‌شد.

سمانه کریمی:

جواب من مختص کسانی است که می‌خواهند اپلای کنند تا در دوره‌ی دکترا و کارهای پژوهشی موفق باشند. بسته به شخصیت افراد، اگر آدمی تشنه‌ی پژوهش هستید و این شرایط برایتان فراهم نیست، نمی‌توانم رد کنم که خارج از کشور از نظر علمی و اقتصادی شرایط بهتری موجود است. البته با این فرض که به دانشگاهی برویم که حداقل از نظر شرایط علمی بهترباشد. باید در نظر گرفت که حداقل در دانشگاه تهران مراحل دکتراگرفتن خیلی نزدیک به مراحلی است که در دانشگاه های آمریکا وجود دارد.

مهم‌تر از اپلای این است که خود فرد به‌دنبال چه چیزی است، چراکه خود اپلای‌کردن هدف نیست. ولی اپلای‌کردن، در یک دانشگاه خوب و نه هر دانشگاهی، فرصت خوبی است برای این‌که آدم‌ها، مخصوصاً افرادی که می‌خواهند بعداً فعالیت دانشگاهی داشته باشند از لحاظ علمی پیشرفت کرده و تجربه کسب کنند.

> من توصیه می‌کنم کارآموزی در خارج از کشور را تجربه کنید چون می‌توانید در این فرصت، فضای دانشگاهی و علمی و همچنین وضعیت زندگی را بسنجید.

شمیم طاهری:

تعدادی از دانش‌جویان صرفاً قصد خروج از ایران و زندگی در خارج از کشور را دارند، اما از آن‌جا که در این بازه‌ی سنی خارج‌شدن از کشور به بهانه‌های دیگر سخت است، به‌بهانه‌ی درس‌خواندن این کار را انجام می‌دهند.

از طرفی، تعدادی از دانش‌جویانی که می‌شناختم، در زمان اپلای‌کردن به‌دنبال درس‌خواندن و یادگرفتن بودند. آدم‌هایی بودند که به‌نظر می‌آمد خیلی نزدیک به محیط آکادمیک باشند، اما نوع بازار کار و فرصت‌هایی که در خارج از کشور وجود دارد باعث شد همه‌ی چیزهایی را که به‌برداشت ما - و حتی خودشان - برای‌شان ارزش است رها کنند و مسیر جدیدی برای خود انتخاب کنند. البته در مورد درستی یا نادرستی آن قضاوت نمی‌کنم.

گذشته از این دو دسته، تعدادی نیز می‌مانند که به‌هدف درس خواندن رفتند و واقعاً همان هدف را دنبال می‌کنند. هر دانشگاهی از نظر آموزشی ارزش رفتن ندارد؛ به‌نظرم محیطی که در دانشگاه‌های خوب خارج از کشور وجود دارد از جنبه‌های آموزشی، پژوهشی، تعدد افراد قوی از دید علمی و متوسط سواد این افراد بسیار قوی‌تر و پویاتر از فضای داخل کشور است.

به این نکته توجه کنید، وقتی شما از کشور خارج می‌شوید، فقط در محوطه‌ی دانشگاه نیستید و در جامعه و محیط آن کشور زندگی می‌کنید. در این‌که الان شما خوب در دانشگاه درس می‌خوانید ممکن است عوامل دیگری نیز دخیل باشد و چون خیلی به محیط نزدیک هستید، شاید نتوانید اهمیت هر کدام را درک کنید. این مسئله وقتی مهم می‌شود که شما از اینجا بروید و ببینید محیط داخل دانشگاه خیلی شبیه این‌جاست و چه بسا همه چیز از این‌جا بهتر باشد، ولی خارج از دانشگاه، شاید آدم‌ها و محیطی را نداشته باشید که بخشی از موفقیت‌تان در ایران را مدیون آن‌ها بوده‌اید.

من توصیه می‌کنم کارآموزی در خارج از کشور را تجربه کنید چون می‌توانید در این فرصت، فضای دانشگاهی و علمی و همچنین وضعیت زندگی را بسنجید. منتها سعی کنید از قبل و با دانشی هرچند مختصر وارد آن فضا شوید. از دست دادن این موقعیت شاید حیف باشد.

احسان خامس‌پناه:

بودن با آدم‌های مختلف و گشتن در جاهای مختلف خودش تجربه است و این تجربه‌ها قطعاً مفید است. فقط باید نسبت هزینه‌ای که می‌کنید را به سودی که می‌برید حساب کنید. داریم عمر را از دست می‌دهیم و در مقابل تجربه کسب می‌کنیم. من شخصاً با توجه به شرایطی که بعد از دکترا برایم پیش آمد تصمیم گرفتم در ایران بمانم. من می‌خواستم در سطح جهانی پژوهش کنم؛ پژوهشی که خوب انجام شود و جامعه‌ی آکادمیک از آن تقدیر کند؛ می‌خواستم در آن فضا نفس بکشم. تا حدی دشوار بود ولی باعث شد بتوانم به همه‌ی چیزهایی که می‌خواستم برسم. الان در سطح جهانی پژوهش می‌کنم، مقالاتم را در مجلات معتبر مربوط چاپ می‌کنم، مقالات کنفرانس‌ها را بازبینی می‌کنم. این‌ها چیزهایی است که من توانستم این‌جا به‌دست آورم و این مهم است؛ فرصت کار پژوهشی در سطح جهانی در ایران هم وجود دارد. نمی‌گویم چنین چیزی در هر شاخه و هر شرایطی ممکن است، اما از طرف دیگر در خارج‌ازکشور نیز در هر شرایطی امکان‌پذیر نیست.

من در دوره‌ی فرصت مطالعاتی دکترا که در طول آن متناوباً ۳ ماه این‌جا بودم و ۳ ماه خارج از کشور، تجربه‌های زیادی کسب کردم، اما خانواده را از دست داده‌بودم؛ چیزی که برای من مهم بود. من حاضر نیستم دور هم جمع‌شدن خانواده را با چیزی عوض کنم. درنتیجه اگر با دیدی که الان دارم به ۵ سال قبل برمی‌گشتم، بین رفتن و نرفتن قطعاً نرفتن را انتخاب می‌کردم. من دوست دارم که با تعاملات خودم زندگی کنم و قصد عوض‌شدن هم ندارم. یعنی به بلوغی رسیده‌ام که خودم را پیدا کرده‌ام. اگر به جای دیگری می‌رفتم باید خودم را تغییر می‌دادم یا در محیطی منزوی از افراد آن مکان قرار می‌گرفتم. کاملاً به شخصیت بستگی دارد و ممکن است که شما چنین نباشید. فقط بدانید که با هر انتخابی حتماً چیزی را از دست می‌دهید و در مقابل چیزی به‌دست می‌آورید. همه‌ی این‌ها را کامل بسنجید.

امیر صدیقی:

من برای هیچ‌جایی اپلای نکرده‌ام چون تازه دو سال است که درس‌خوان شده‌ام. ولی هم با خارجی‌ها کار کرده‌ام و هم اقامت کانادا را دارم. دوسال پیش به کانادا رفتم ودر آن‌جا اولین تجربه‌ی کارم را خیلی سریع و با حقوقی خوب، شروع کردم. با این همه باور کنید آن‌جا نه «فان» ایران را دارد و نه لذتی را که با هم داریم.

در مقابل در ایران می‌توانید در بازار کار شبکه‌ی دوستی بسازید و همچنین در ایران توسط خانواده و دوستان حمایت می‌شوید، اما در آن‌جا تنها یاور شما، حساب بانکی‌تان است و هیچ حمایت دیگری حتی از جانب خانواده هم ندارید. من آن‌جا درس نخوانده‌ام اما شنیده‌ام که تحصیل در خارج از ایران، تجربه‌ای دل‌نشین است.

موضوع فصل
داده‌های بزرگ

مرئی‌کردن نامرئی :: مقدمه‌ای بر داده‌های بزرگ :: آینده‌ی زیست‌فناوری :: پردازش جریان داده :: بررسی شباهت افراد بر اساس تاریخچه‌ی مکانی :: داستان یک استخدام

میزان اطلاعاتی که هر انسانی روزانه به‌شکل داده‌های متفاوت به جهان اطرافش تزریق می‌کند، از هر کلیک گرفته تا عکس‌هایی که در سایت‌های مختلف توسط او منتشر می‌شود، مسیرهایی که روزانه از آن‌ها عبور می‌کند و... شگفت‌آور است. تحلیل هر کدام از این داده‌ها به هوشمندترشدن تصمیمات و تخمین دقیق‌تر آن‌ها در زمینه‌های مختلف، کاهش هزینه‌ها، کاهش ریسک در امور و بسیاری از مسائل دیگر می‌انجامد؛ این موضوعات زندگی انسان‌ها را بسیار تغییر می‌دهد. اما نکته‌ی قابل تأمل در این‌جا بحث نگهداری و تحلیل این حجم عظیم داده است. فقط به وسعت کار فکر کنید. در این بخش با بحث داده‌های بزرگ، مسائل و چالش‌های مرتبط با آن و کاربردهایش در زمینه‌های مختلف آشنا می‌شویم.

مرئی‌کردن نامرئی

ترجمه‌ی کتاب داده‌گرایی، استیو لور

مترجم: آذین زهرایی
azin.zahrayi@gmail.com

کمی خارج از ممفیس، سمفونی‌ای صنعتی متشکل از ماشین‌ها و انسان‌ها کالاهایی را از سویی به سوی دیگر می‌برند، حرکاتِ به‌دقت هماهنگ‌شده‌ی آن‌ها با برچسب‌های شناسایی و توسط اسکنرِ بارکد و تراشه‌های رادیویی دنبال می‌شود. دست‌های مکانیکی بسته‌های سلفون‌کشیده‌شده را از روی نوارهای نقاله می‌قاپند، در حالی که جرثقیل‌های چنگک‌دار بسته‌ها را به داخل کامیون‌های حمل بار انتقال می‌دهند. این انسان‌ها هستند که جریان کالاها را هدایت و نظارت می‌کنند و جرثقیل‌ها و کامیون‌ها را می‌رانند.

این نمایش عظیم از کارایی را مک‌کسون[1]، که تقریباً یک-سوم کل محصولات دارویی آمریکا را توزیع می‌کند، به‌اجرا درمی‌آورد. ساختمان‌های آن، با مساحتی بیش از هشت زمین فوتبال، مرکز فعالیت شبکه‌ی توزیع ملی مک‌کسون را تشکیل می‌دهند. یک سازمان برجسته که به ۲۶هزار مشتری، از جمله داروخانه‌های محلی و فروشگاه‌های والمارت، کالا می‌فرستد. محموله‌ی اصلی دارو است، تقریباً ۲۴۰میلیون قرص در روز. تجارت توزیع محصولات دارویی تجارتی است با حجم بالا و حاشیه‌ی سود بسیار پایین. پس قابل درک است که کارایی و بهره‌وری دهه‌هاست که برای مک‌کسون فوق‌العاده مهم بوده است.

با این وجود در چند سال گذشته، مک‌کسون با کم‌کردن یک‌میلیارد دلار از پولی که هر لحظه در شبکه‌اش در گردش است قدمی قابل توجه به جلو برداشته است. نتیجه‌ی نهایی حاصل بینشی است که از طریق جمع‌آوری اطلاعات مکان و حمل‌ونقل محصولات به‌وسیله‌ی اسکنرها و حسگرها، و سپس داده‌کاوی این اطلاعات به‌دست آمده است. این داده‌کاوی توسط نرم‌افزارهای هوشمند برای شناسایی فرصت‌های بالقوه به‌منظور صرفه‌جویی در زمان و هزینه‌ها انجام می‌شود. این نمای بهتر از تجارت که با استفاده از فناوری محقق شده یک موفقیت بزرگ است که دونالد واکر[2]، یک مدیر اجرایی ارشد مک‌کسون، آن را «مرئی‌کردن نامرئی» می‌خواند.

در آتلانتا، من در طبقه‌ی پنجم بیمارستان دانشگاه اِموری، بیرون یکی از اتاق‌های شیشه‌ای بخش مراقبت‌های ویژه ایستاده‌ام. انبوهی از دستگاه‌های الکترونیکی و محاسبات پزشکی اتاق را شلوغ کرده است: یک دستگاه تنفس مصنوعی، یک دستگاه دیالیز، دستگاه‌هایی که آنتی‌بیوتیک و مواد مخدر ضددرد را تزریق کرده و ابزارهایی که ضربان قلب، تنفس، فشار خون، اشباع اکسیژن و دیگر علائم حیاتی را نظارت می‌کنند. تقریباً هر ماشین یک کامپیوتر مخصوص به خود دارد که هر کدام مخلوطی از صداهای ناهنجار بوق و هشدار ایجاد می‌کند. من یک دوجین صفحه‌ی نمایشگر می‌شمارم، نمایشگرهای بزرگ و کوچک.

یک بخش مراقبت‌های ویژه‌ی معمولی بیست‌تخته حدوداً ۱۶۰،۰۰۰ داده در ثانیه تولید می‌کند. طبق تحقیقات اموری، در میان همه‌ی این داده‌ها، پزشکان و پرستاران به‌سرعت تصمیم‌گیری می‌کنند؛ تقریباً ۱۰۰ تصمیم در روز برای هر بیمار، یا بیش از ۹٫۳میلیون تصمیم درباره‌ی مراقبت در طول یک سال. پس امکان اشتباه زیادی وجود دارد. این افراد پرمشغله نیاز به کمک دارند و اموری یکی از مراکز تحقیقات پزشکی انگشت‌شماری است که برای متحول‌کردن مراقبت‌های ویژه بر اساس داده‌ها فعالیت می‌کند. جریان داده‌های پزشکی که توسط دستگاه‌های مراقبت تولید می‌شود با استفاده از نرم‌افزار هوشمندی پردازش می‌شود تا بر اساس نشانه‌های اولیه، قبل از این‌که وضع بیمار رو به وخامت بگذارد، این تغییرات را پیش‌بینی کند.

تفسیر حجم انبوه داده و تشخیص الگوهای ظریف جایی است که رایانه‌ها و الگوریتم‌های نرم‌افزاری نسبت به انسان‌ها برتری دارند. دکتر تیموتی بوکمان[3] سرپرست این تحقیقات در اموری است. بوکمان که یک جراح، دانشمند و خلبان باتجربه است از یک قیاس استفاده می‌کند تا هدفش را توضیح دهد. اطلاعات مکانی هواپیماها روی صفحه‌های نمایش مرکز کنترل ترافیک هوایی نمایش داده می‌شود و آن‌ها هنگامی که پرواز از مسیر منحرف می‌شود و خیلی پیش از آن‌که هواپیما دچار سانحه شود متوجه این انحراف می‌شوند. بوکمان به‌دنبال همین سیستم هشدار زودهنگام برای بیمارانی است که الگوی علائم حیاتی آن‌ها از مسیر عادی منحرف شده است. به‌گفته‌ی او: «این هدفی است که داده‌های بزرگ رسیدن به آن را برای ما ممکن می‌کند.»

دوره‌ی داده‌های بزرگ در حال فرارسیدن است و وسعت آن از شرکت‌های اینترنتی نوپا در دره‌ی سیلیکون مثل گوگل و فیس‌بوک فراتر می‌رود. داده‌های بزرگ از دنیای دیجیتال بیت‌ها شروع شد و به‌سرعت در دنیای فیزیکی اتم‌ها قدم گذاشت. مرکز توزیع مک‌کسون و بخش مراقبت‌های ویژه‌ی اموری مسیر آینده را نشان می‌دهند؛ این‌که داده‌های بزرگ چه‌طور در هزینه‌ها صرفه‌جویی می‌کند و زندگی افراد را نجات می‌دهد. مطمئناً دورنمای این فناوریْ هوش مصنوعی است که به‌صورت یک لایه‌ی اطلاعات‌محور روی هر دو دنیای فیزیکی و دیجیتال قرار می‌گیرد. امروزه قدم‌های اولیه به‌سمت این رؤیا را می‌بینیم. فناوری داده‌های بزرگ آغازگر یک انقلاب در اندازه‌گیری است که به‌نظر می‌رسد پایه‌های موج بعدی کارایی و نوآوری در اقتصاد را شکل دهد. ولی این‌جا حرف از چیزی بیشتر از روی کار آمدنِ یک فناوری است. داده‌های بزرگ ناقل یک نقطه‌نظر یا یک فلسفه راجع به این است که تصمیمات در آینده چگونه گرفته می‌شوند یا حتی بهتر است چه‌طور گرفته شوند. دیوید بروکس[4]، همکار من در نیویورک تایمز، این طرز تفکر روبه‌رشد را «داده‌گرایی»[5] خوانده است، لفظی که من هم از آن استفاده می‌کنم چرا که وسعت این پدیده را نشان می‌دهد. ابزارهای نوآوری، همان‌طور که غالباً در گذشته دیده‌ایم، فقط برای رشد اقتصادی اهمیت ندارند، بلکه می‌توانند دید ما را نسبت به جهان و تصمیم‌هایی که در مورد آن می‌گیریم نیز تغییر دهند.

مجموعه‌ای از فناوری‌ها تکیه‌گاه داده‌های بزرگ است. اولین آن‌ها تمامی منابع قدیمی و جدید داده است: وبسایت‌ها، عادت‌های مرور وب، سیگنال‌های حسگرها، رسانه‌های اجتماعی، اطلاعات مکانی گوشی‌های هوشمند، داده‌های ژنومیک و تصاویر دوربین‌های مداربسته. موج عظیم داده همین‌طور بزرگ و بزرگ‌تر می‌شود و هر دو سال، تقریباً حجم آن دو برابر می‌شود. ولی از نظر من جنبه‌ی «بزرگی» داده‌های بزرگ مبالغه‌شده‌ترین و شاید در بیشتر مواقع کم‌اهمیت‌ترین جنبه‌ی آن است. شمار داده‌های دنیا به بازی تخمین و گمانه‌زنی برای گیک‌ها تبدیل شده است؛ گشت‌هایی در دنیای زبانیِ زتابایت، یتابایت و برنتوبایت. اعداد و معادل‌های آن‌ها بسیار توجه‌برانگیز هستند. طبق تخمینی، ۹۰٪ تمام داده‌های ذخیره‌شده‌ی تاریخ در دو سال گذشته تولید شده است. در سال ۲۰۱۴، شرکت بین‌المللی داده‌ها[6] حجم جهان داده را ۴٫۴ زتابایت تخمین زد، که معادل ۴٫۴میلیارد ترابایت است. نگهداری این حجم اطلاعات به‌گفته‌ی این شرکت تحقیقاتی نیازمند تعداد کافی آی‌پد برای ساختن پشته‌ای به‌ارتفاع ۲۵۰هزار کیلومتر یا دو-سوم فاصله تا کره‌ی ماه است.

۱ - McKesson
۲ - Donald Walker
۳ - Timothy Buchman
۴ - David Brooks
۵ - Data-ism
۶ - IDC

ولی همه‌ی داده‌ها یکسان نیستند. حجم خیره‌کننده‌ی داده‌ها، به‌خاطر افزایش تولید عکس و ویدیو، رشد بسیار زیادی پیدا کرده است. فقط به تمامی عکس‌ها و ویدیوهایی که توسط گوشی‌های هوشمند گرفته می‌شود فکر کنید. گفته می‌شود که یک عکس به‌اندازه‌ی هزاران کلمه ارزش دارد. ولی در دنیای دیجیتال، این صرفاً دست‌کم‌گرفتن حقیقت است. هر حرف از یک متن یک بایت مصرف می‌کند، در حالی که یک عکس با کیفیت بالا با مگابایت - میلیون‌ها بایت - اندازه‌گیری می‌شود، و از نظر میزان مصرف بیت‌ها، ویدیو در مقابل عکس مانند فیل و فنجان است. همین‌طور که من این را می‌نویسم، هر دقیقه ۴۸ ساعت ویدیو در یوتیوب آپلود می‌شود؛ با سرعتی که احتمالاً فقط در حال افزایش است.

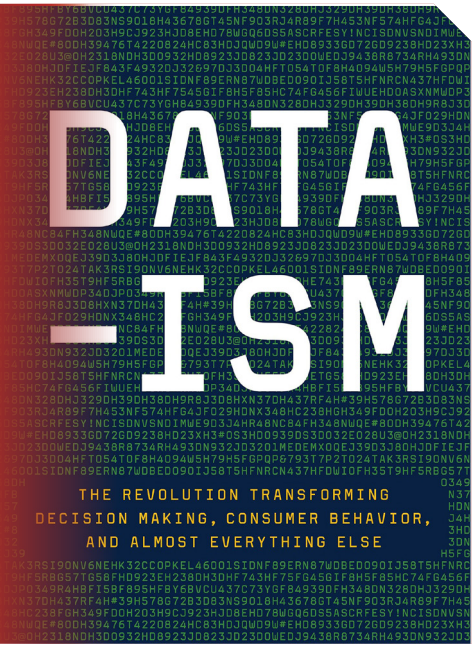
بزرگی در داده‌های بزرگ اهمیت دارد، ولی خیلی کمتر از آن که بسیاری از مردم فکر می‌کنند. آب اقیانوس‌ها هم بسیار زیاد است، ولی نمی‌توان همه‌ی آن را نوشید. معضل مهم‌تر دیگر آن است که چگونه از داده‌ها استفاده کنیم. در حالی که پیشرفت‌ها در پردازش‌های کامپیوتری، ذخیره‌سازی و حافظه به چالش داده‌ها کمک می‌کند، بزرگ‌ترین قدم پیش رو در عرصه‌ی نرم‌افزار است. کدهای حیاتی از ابزارهای هوش مصنوعی مثل نرم‌افزارهای یادگیری ماشین حاصل می‌شوند.

داده و فناوری‌های هوشمند درهایی را به افق‌های جدید اندازه‌گیری باز می‌کنند. فناوری داده‌های بزرگ معادل تلسکوپ و میکروسکوپ در عصر دیجیتال است. هر دوی این‌ها این امکان را فراهم کردند تا چیزها را به‌گونه‌ای بی‌سابقه ببینیم و اندازه بگیریم؛ با تلسکوپ آسمان‌ها و کهکشان‌های جدید، و با میکروسکوپ معماهای زندگی در سطح سلولی.

همان‌طور که تلسکوپ‌های مدرن ستاره‌شناسی را متحول کردند و میکروسکوپ‌های مدرن همین کار را در زیست‌شناسی کردند، داده‌های بزرگ چیزی مشابه را وعده می‌دهد، فقط در مقیاسی گسترده‌تر در هر رشته و موضوعی. پیشرفت‌های گسترده در فناوری موتور تغییرات اقتصادی است. اینترنت اقتصاد ارتباطات را متحول کرد. سپس فناوری‌های دیگر مثل وب در اینترنت بنا نهاده شد، که به بستر نرم‌افزاری برای نوآوری و تجارت‌های تازه تبدیل شده است. به‌طور مشابه، داده‌های بزرگ با این‌که هنوز یک فناوری جوان است، اقتصاد اکتشاف را متحول می‌کند و شاید بتوان گفت به بستر نرم‌افزاری برای تصمیم‌گیری انسان تبدیل می‌شود.

تصمیم‌ها - از همه نوع - بیشتر و بیشتر بر اساس داده و تحلیل به‌جای تجربه و شهود گرفته می‌شوند؛ دانش بیشتر و تصمیمات حسی کمتر.

بخوانید



استیو لُور، نویسنده‌ی کتاب «داده‌گرایی»، در این کتاب در مورد داده‌های بزرگ و انقلابی که داده‌های بزرگ و تکنولوژی‌های پردازش آن به‌وجود آورده است صحبت می‌کند. در مورد حجم عظیم داده حرف می‌زند و می‌گوید در آینده تصمیم‌گیری‌های مختلف بر اساس تحلیل این داده‌ها انجام خواهد شد. وی همچنین در مورد تازه‌کارها و دانشمندان داده در برابر غول‌های تکنولوژی سخن می‌آورد. اگر به این مباحث علاقه‌مندید خواندن این کتاب را به شما توصیه می‌کنیم.

ببینید

یادگیری عمیق شاخه‌ای از یادگیری ماشینی است و شامل مجموعه‌ای از الگوریتم‌هایی است که تلاش می‌کند تا مفاهیم سطح بالا را با استفاده از یادگیری در سطح‌ها و لایه‌های مختلف مدل کرده و فعالیت‌های لایه‌های مختلف عصب‌ها در مغز را شبیه‌سازی کند. سیستم‌های مبتنی بر یادگیری عمیق به‌دلیل وجود حجم زیادی از داده‌ها و پردازش‌های مرتبط با آن‌ها وابسته به مسائل و چالش‌های داده‌های بزرگ خواهد بود. ویدیوی Deep Learning: Intelligence from Big Data تلاش کرده است تا به این مسائل بپردازد.

بشنوید

اگر به‌دنبال این هستید که هر ماه بسته‌ای کامل از اخبار فناوری را کامل و یک‌جا در اختیار داشته باشید، حتماً به وبسایت jadi.net سر بزنید و این رادیو را دنبال کنید. برای آشنایی بیشتر با داده‌های بزرگ و چالش‌های مرتبط با آن، یادگیری ماشینی، چگونگی ذخیره‌ی داده، هدوپ، NoSQL، Hive علم داده‌ها و بسیاری از دیگر مسائل مرتبط با این بحث، می‌توانید به «شماره‌ی ۵۳ - بیگ دیتا، دانشمند داده و برگشت دایناسورها» از رادیو گیک مراجعه کنید.

مقدمه‌ای بر داده‌های بزرگ

مجتبی بنایی متولد سال ۱۳۵۹ است. وی مدرک کارشناسی خود را در رشته‌ی کامپیوتر (نرم‌افزار) از دانشگاه تهران و مدرک کارشناسی ارشدش را در همین رشته از دانشگاه تربیت مدرس اخذ کرده است. وبسایت bigdata.ir متعلق به او است. وی همچنین سابقه‌ی همکاری با شرکت‌هایی مثل داده‌پردازی ایران و فرافرهنگ در دوران دانشجویی و شرکت‌هایی مانند ریپابلیشن آمریکا و هلوبل سوییس را داشته است. او هم‌اکنون یکی از اعضای هیئت‌علمی دانشگاه بزرگمهر قائنات است.

مجتبی بنائی
mojtaba.banaie@gmail.com

داده‌های بزرگ چند سالی است که در ادبیات فناوری اطلاعات به یک اصطلاح فراگیر تبدیل شده است و در این مقاله سعی شده است تا این حوزه‌ی نوین به‌صورت اجمالی معرفی گردد.

اگر بخواهیم تعریفی از داده‌های بزرگ ارائه کنیم، می‌توانیم آن را مجموعه داده‌هایی بدانیم که اندازه‌شان فراتر از حدی است که با نرم‌افزارها و روش‌های معمول بتوان آن‌ها را در یک زمان قابل قبول، دریافت، ذخیره، مدیریت و پردازش کرد. در این تعریف، حجم داده مشخص نشده است چون میزان کلان‌بودن داده، هم‌زمان با پیشرفت فناوری‌های ذخیره‌سازی و پردازش اطلاعات، روزبه‌روز و عموماً به‌دلیل تولید داده توسط تجهیزات و ابزارهای مختلف دیجیتال در حال افزایش است. گوشی‌های موبایل، حسگرهای محیطی، گزارش‌های[1] نرم‌افزارهای مختلف، دوربین‌ها، میکروفون‌ها، دستگاه‌های تشخیص RFID[2]، شبکه‌های حسگر بی‌سیم، ایستگاه‌های هواشناسی، سامانه‌های امواج رادیویی، مبادلات شبکه‌های اجتماعی آنلاین، متون و اسناد اینترنتی، داده‌های نجوم، اطلاعات پزشکی و سلامت بیماران، اطلاعات سامانه‌های خرید از فروشگاه‌ها، پژوهش‌های زمین‌شناسی و غیره نمونه‌هایی از داده‌ها در مقیاس کلان هستند. مقیاسی که امروزه از گیگابایت و ترابایت به پتابایت و اگزابایت و زتابایت در حال حرکت است.

برای ایجاد یک دید مناسب در مورد داده‌های بزرگ و اهمیت آن، جامعه‌ای را تصور کنید که در آن جمعیت به‌طور نمایی در حال افزایش است، اما خدمات و زیرساخت‌های عمومی آن نمی‌تواند پاسخگوی رشد جمعیت باشد و از عهده‌ی مدیریت آن برآید. چنین شرایطی در حوزه‌ی داده در حال وقوع است. بنابراین نیازمند توسعه‌ی زیرساخت‌های فنی برای مدیریت داده و رشد آن در بخش‌هایی نظیر جمع‌آوری، ذخیره‌سازی، جستجو، به‌اشتراک‌گذاری و تحلیل هستیم. دست‌یابی به این توانمندی معادل است با شرایطی که مثلاً بتوانیم:

«هنگامی‌که با اطلاعات بیشتری در حوزه‌ی سلامت مواجه هستیم، با بازدهی بیشتری سلامت را ارتقا دهیم»،

«در شرایطی که خطرات امنیتی افزایش پیدا می‌کند، سطح امنیت بیشتری را فراهم کنیم»،

«وقتی که با رویدادهای بیشتری از نظر آب‌وهوایی مواجه باشیم، توان پیش‌بینی دقیق‌تر و بهتری به‌دست آوریم»،

«در دنیایی با خودروهای بیشتر، آمار تصادفات و حوادث را کاهش دهیم»،

«تعداد تراکنش‌های بانکی، بیمه و مالی افزایش پیدا کند، ولی تقلب کمتری را شاهد باشیم»،

«با منابع طبیعی کمتر، به انرژی بیشتر و ارزان‌تری دسترسی داشته باشیم»،

و بسیاری موارد دیگر از این قبیل که اهمیت پنهان داده‌های بزرگ را نشان می‌دهد.

چالش‌ها و خصوصیات داده‌های بزرگ

تاکنون چالش‌های زیادی در حوزه‌ی داده‌های بزرگ مطرح شده است که ویژگی‌های گوناگون این حوزه را از جنبه‌ی تئوری بیان می‌کند. در ابتدا سه بعد اصلی حجم، نرخ تولید و تنوع داده‌ها به‌عنوان سه چالش اصلی و مشخصه‌ی داده‌های بزرگ (3V's) عنوان شدند، ولی در ادامه چالش‌های بیشتری توسط محققان مطرح شده است:

- **حجم داده (Volume)**: حجم داده‌ها به‌کمک پدیده‌ی اینترنت، دستگاه‌های الکترونیکی و موبایل‌ها، زیرساخت‌های شبکه و سایر منابع هرساله رشد نمایی دارد و پیش‌بینی شده است که تا سال ۲۰۲۰ حدود ده زتابایت داده در جهان وجود خواهد داشت.
- **نرخ تولید (Velocity)**: داده‌ها از طریق برنامه‌های کاربردی و حسگرهای بسیار زیادی که در محیط وجود دارند با سرعت بسیار زیاد و به‌صورت بلادرنگ تولید می‌شوند که اغلب باید در لحظه پردازش و ذخیره شوند.
- **تنوع (Variety)**: تنوع در منابع و نوع داده بسیار زیاد است که در نتیجه ساختارهای داده‌ای بسیار زیادی وجود دارد. بیشتر حجم داده‌های دنیا نیز بی‌ساختار و بسیار متنوع است. بخشی از داده‌ها امروزه در بانک‌های اطلاعاتی، بخشی در صفحات وب، بخشی به صورت XML و JSON و بقیه نیز در فایل‌ها با قالب‌های متفاوت ذخیره شده‌اند که عمل پردازش آن‌ها را پیچیده می‌کند.
- **صحت (Veracity)**: با توجه به این‌که داده‌ها از منابع مختلف دریافت می‌شوند، ممکن است نتوان به همه‌ی آن‌ها اعتماد کرد.

۱- logs
۲- Radio-frequency identification

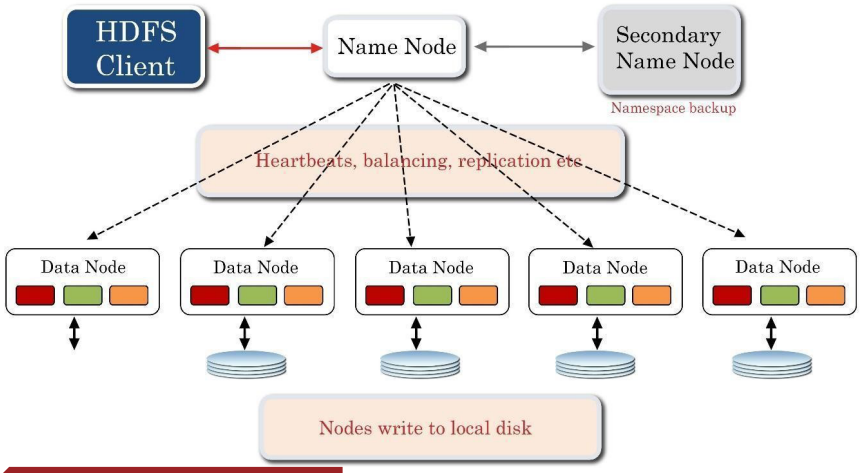
جدول ۱: مقایسه‌ی داده‌های کلاسیک با داده‌های بزرگ

معیار	داده‌های سنتی	داده‌های بزرگ
اندازه	گیگابایت تا ترابایت	پتابایت تا اگزابایت
معماری	متمرکز	توزیع‌شده
ساختار	دارای ساختار	بی‌ساختار یا نیم‌ساختار
مدل داده	مدل داده‌ی ثابت	بدون شِمای مشخص
ارتباط داخلی	ارتباطات پیچیده بین رکوردها	فاقد ارتباطات داخلی پیچیده

مثلاً در یک شبکه‌ی اجتماعی، ممکن است نظرهای زیادی درباره‌ی یک موضوع خاص ارائه شود. اما این‌که آیا همه‌ی آن‌ها صحیح و قابل اطمینان هستند، موضوعی است که نمی‌توان به‌سادگی از کنار آن در حجم بسیار زیادی از اطلاعات گذشت.

- **اعتبار (Validity)**: با فرض این‌که داده صحیح باشد، ممکن است برای برخی کاربردها مناسب نباشد یا به‌عبارت دیگر، دارای اعتبار کافی برای استفاده در برخی زمینه‌ها نباشد.
- **نوسان (Volatility)**: سرعت تغییر ارزش داده‌های مختلف در طول زمان می‌تواند متغیر باشد. در کاربردهایی نظیر تحلیل ارز و بورس، داده با نوسان زیادی مواجه است و داده‌ها به‌سرعت ارزش خود را از دست می‌دهند و مقادیر جدیدی به خود می‌گیرند. اگرچه نگهداری اطلاعات در زمان طولانی به‌منظور تحلیل تغییرات و نوسان داده‌ها حائز اهمیت است، افزایش دوره‌ی نگهداری اطلاعات مسلماً هزینه‌های پیاده‌سازی زیادی را در بر خواهد داشت که باید در نظر گرفته شود.
- **نمایش (Visualization)**: یکی از کارهای مشکل در حوزه‌ی داده‌های بزرگ نمایش اطلاعات است. این‌که بخواهیم کاری کنیم که حجم عظیم اطلاعات با ارتباطات پیچیده به‌خوبی قابل فهم و قابل مطالعه باشد از طریق روش‌های تحلیلی و بصری‌سازی مناسب اطلاعات امکان‌پذیر است.
- **ارزش (Value)**: آیا هزینه‌ای که برای نگهداری و پردازش داده‌ها پرداخت می‌شود، ارزش آن را از نظر تصمیم‌گیری دارد یا نه؟ و ارزش و فایده‌ی مورد نظر را برای یک سازمان خواهد داشت؟

به‌طور کلی، تفاوت‌های اصلی داده‌های بزرگ و داده‌های سنتی در جدول ۱ بیان شده است.



شکل ۲: ساختار فایل‌سیستم HDFS

ابزارهای ذخیره و پردازش در حوزه‌ی داده‌های بزرگ

رهیافت‌هایی که امروزه در بخش پردازش داده‌های بزرگ مطرح هستند چندین خاصیت مشترک دارند:

- قابلیت اجرا بر روی سخت‌افزار موجود که باعث می‌شود بتوان با هزینه‌ی کم، امکان پردازش موازی و ارتقای سخت‌افزاری را فراهم کرد.
- استفاده از ابزارهای تحلیل و مصورسازی پیشرفته برای سهولت کاربر نهایی.
- استفاده‌ی هم‌زمان از ابزارها و کتاب‌خانه‌های مختلف که معماری داده‌ی یک سازمان را شکل می‌دهند.
- استفاده از بانک‌های اطلاعاتی غیررابطه‌ای (NoSQL) به‌عنوان جزئی از معماری و بستر داده‌ی سازمان.

دو رهیافت اصلی که امروزه در پردازش و تحلیل داده‌های بزرگ بیشترین رواج را دارند عبارتند از هدوپ و بانک‌های اطلاعاتی NoSQL.

هدوپ

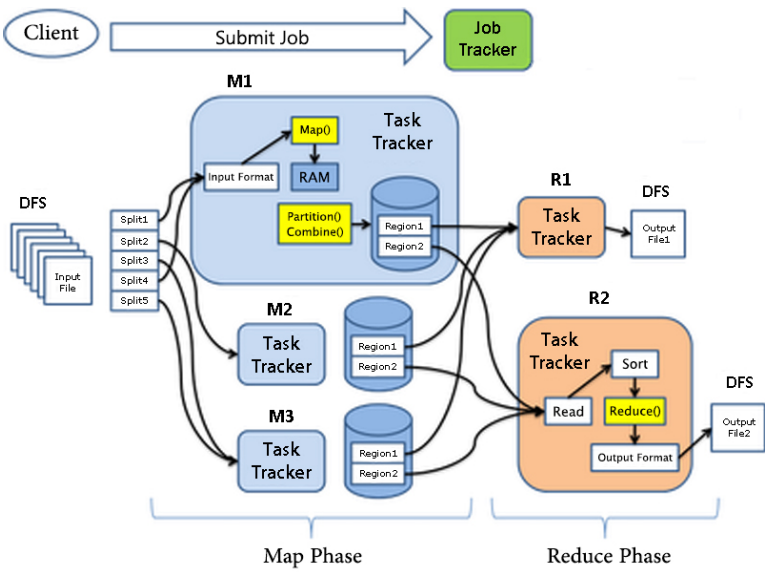
هدوپ یک چهارچوب متن‌باز برای پردازش، ذخیره و تحلیل حجم عظیم داده‌های توزیع‌شده و بدون ساختار است. منشاء اصلی پیدایش این چهارچوب پردازشی به شرکت‌های جستجوی اینترنتی یاهو و گوگل بازمی‌گردد که برای شاخص‌زدن[۳] صفحات وب و جستجوی آن‌ها به ابزار و مدل‌های جدید پردازشی نیاز داشتند. این چهارچوب برای پردازش موازی داده‌ها در سطح پتابایت و اگزابایت که بر روی رایانه‌های معمولی (همچون رایانه‌های شخصی) توزیع شده‌اند طراحی شده است که خوشه‌ی تشکیل‌دهنده‌ی آن به‌راحتی و بسته به نیاز، قابل گسترش است. هدوپ در حال حاضر توسط بنیاد آپاچی توسعه و گسترش می‌یابد.

هدوپ چگونه کار می‌کند

در این سامانه فایل‌های داده‌ای با حجم بالا مانند فایل‌های ثبت تراکنش، خوراک[۴] شبکه‌های اجتماعی و سایر منابع داده‌ای ابتدا بخش‌بندی شده و سپس در شبکه توزیع می‌شوند.

وظیفه‌ی تقسیم، ذخیره و بازیابی فایل‌های حجیم بر روی یک کلاستر هدوپ را فایل‌سیستم توزیع‌شده‌ی آن با نام HDFS[۵] بر عهده دارد. برای بالابردن ضریب

۳- Indexing
۴- Feed
۵- Hadoop Distributed File System



شکل ۳: ساختار عملیاتی هدوپ و فرآیند نگاشت کاهش

اطمینان سیستم، هر بخش از فایل در چندین گره‌ی محاسباتی ذخیره می‌شود تا در صورت ازکارافتادن یک گره، آن فایل باز هم قابل بازیابی باشد.

در HDFS سه نوع گره‌ی محاسباتی داریم. گره‌ی مدیریت‌نام وظیفه‌ی تقسیم فایل‌ها و ذخیره‌ی آدرس هر بخش از آن را بر عهده دارد. بررسی دوره‌ای گره‌ها و تعیین از رده خارج‌شدن آن‌ها هم جزء وظایف این مؤلفه از سیستم مدیریت فایل هدوپ است.

گره‌ی داده، که تک‌تک رایانه‌های عضو هدوپ را دربرمی گیرد، بلوک‌های فایل را شامل می‌شود. به‌ازای هر مجموعه از گره‌های داده، یک گره‌ی مدیریت‌نام در سامانه‌ی هدوپ وجود دارد. نوع سوم گره‌ی ثانویه است که یک رونوشت از اطلاعات گره‌ی مدیریت‌نام بر روی آن قرار دارد تا در صورت ازکارافتادن آن گره، اطلاعات آن از بین نرود. شکل ۲ شمایی کلی از مؤلفه‌ی مدیریت فایل هدوپ را نشان می‌دهد.

بعد از توزیع داده‌ها در سامانه‌ی هدوپ، تحلیل و پردازش آن‌ها بر عهده‌ی بخش نگاشت‌کاهش[۶] آن است. شکل ۳ این فرآیند را نمایش می‌دهد. در مرحله‌ی اول، کاربر درخواست خود را - که معمولاً یک پرس‌وجو به زبان جاوا است - به گره‌ای که وظیفه‌ی اجرای درخواست‌ها را بر عهده دارد (مدیر درخواست) ارسال می‌کند. در این مرحله، مدیر درخواست بررسی می‌کند که به چه فایل‌هایی برای پاسخ به پرس‌وجوی کاربر نیاز دارد و به‌کمک گره‌ی مدیریت‌نام ، گره‌های داده‌ی حاوی آن بخش‌ها را در کلاستر می‌یابد و این درخواست به تک‌تک آن گره‌ها ارسال می‌شود. این گره‌ها - که هنگام پردازش به آن‌ها مدیر وظیفه می‌گوییم - مستقلاً و به‌صورت موازی کار پردازش داده‌های خود را (اجرای تابع نگاشت) انجام می دهند.

پس از اتمام کار هر مدیر وظیفه، نتایج در همان گره ذخیره می‌شود. پس از آماده‌شدن نتایج میانی، که طبیعتاً چون وابسته به داده‌های موجود در روی یک گره است محلی و ناقص خواهد بود، مدیر درخواست «درخواست کاهش» را به این گره‌ها ارسال می‌کند تا پردازش نهایی را بر روی نتایج انجام دهد و نتیجه‌ی درخواست کاربر نیز در یک گره‌ی محاسباتی نهایی ذخیره شود. در این مرحله، نگاشت‌کاهش به اتمام رسیده است و پردازش بعدی بر روی نتایج حاصل بر عهده‌ی تحلیلگران حوزه‌ی داده‌های بزرگ است. این پردازش می‌تواند به‌صورت مستقیم بر روی نتایج انجام شود یا با انتقال داده‌های حاصل به پایگاه‌داده‌های رابطه‌ای یا انبارهای داده، از روش‌های کلاسیک تحلیل داده استفاده شود.

مزایا و معایب هدوپ

مهم‌ترین مزیت هدوپ توانایی پردازش و تحلیل حجم عظیم داده‌های بدون ساختار یا نیمه‌ساختمندی است که تاکنون پردازش آن‌ها به‌صورت بهینه (از نظر هزینه و زمان) امکان‌پذیر نبوده است.

مزیت بعدی هدوپ به امکان گسترش ساده و مقیاس‌پذیری افقی آن برمی‌گردد که به‌راحتی می‌توان تا سطح اگزابایت داده‌ها را تحلیل کرد و دیگر لازم نیست شرکت‌ها بر روی داده‌های نمونه و زیرمجموعه‌ای از داده‌های اصلی کار کنند و به‌کمک هدوپ، امکان بررسی تمام داده‌ها فراهم شده است.

مزیت دیگر آن هزینه‌ی راه‌اندازی اندک (که دلیل اصلی آن رایگان بودن آن است) و نیز عدم نیاز به سخت‌افزار حرفه‌ای و گران است. به‌خصوص با رواج رایانش ابری و قیمت‌های مناسب آن برای پردازش‌های موردی داده و نیز ابرهای خصوصی، راه‌اندازی یک سامانه‌ی هدوپ به فرآیندی چندساعته تبدیل شده است.

از طرف دیگر، هدوپ و زیرمجموعه‌های آن همگی در مراحل اولیه‌ی توسعه و در نتیجه غیربالغ و نوپا هستند. این امر خود باعث تغییر و اصلاح مداوم این چهارچوب می‌شود که هزینه‌ی آموزش مداوم را به سازمان‌ها تحمیل می‌کند.

از سوی دیگر، نوپابودن این مدل نرم‌افزاری باعث می‌شود افراد کمی مهارت لازم را برای ایجاد و کار با سامانه‌های مبتنی بر هدوپ داشته باشند و برای بسیاری از شرکت‌ها، کمبود نیروی انسانی متخصص مهم‌ترین چالش آن‌ها در استفاده از این سامانه است.

مشکل دیگر هدوپ که ماهیت ذاتی دارد عدم توانایی پردازش بلادرنگ داده‌ها است؛ چون مدیر درخواست باید منتظر تکمیل کار تک‌تک گره‌های محاسباتی سامانه بماند تا بتواند جواب نهایی را به کاربر تحویل دهد. هر چند با رشد سریع فناوری‌های بانک‌های اطلاعاتی NoSQL و تلفیق آن با هدوپ، این مشکل نیز تا حدی رفع خواهد شد.

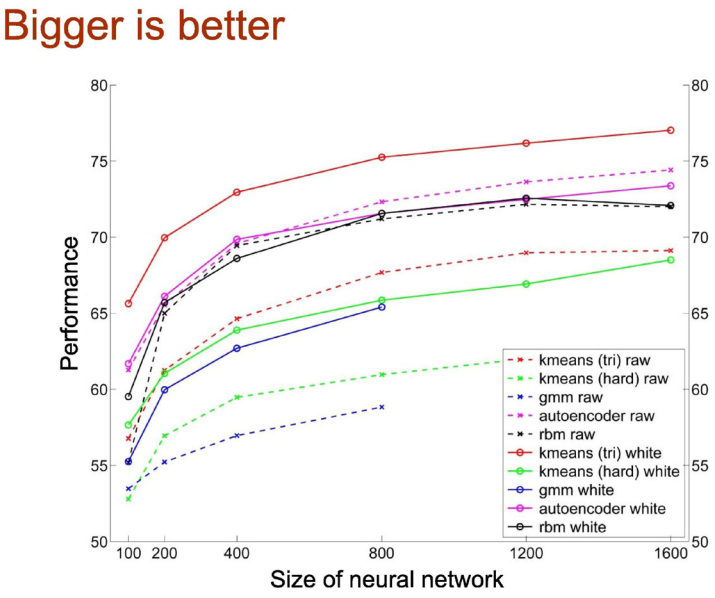
بانک‌های اطلاعاتی NoSQL

هدوپ به‌طور خاص برای پردازش داده‌های بزرگ شکل گرفته است و نیازهای ذخیره و بازیابی داده‌های بزرگ در آن پیش‌بینی نشده است. شرکت‌هایی مانند گوگل، فیس‌بوک، آمازون، توییتر و مانند آن که روزانه نیاز به ذخیره‌ی چندین گیگابایت تا چندین ترابایت داده را دارند و نیز بازیابی سریع و مؤثر اطلاعات برای‌شان امری حیاتی است، دست به ابداع نوع جدیدی از بانک‌های اطلاعاتی زده‌اند که به‌طور خاص برای ذخیره و بازیابی خودکار و حرفه‌ای داده‌های بزرگ طراحی شده‌اند.

این نوع از بانک‌های اطلاعاتی که دیگر مفاهیم کلاسیک پایگاه داده مانند جدول و رکورد در آن‌ها معنای خود را از دست داده است، به بانک‌های اطلاعاتی NoSQL یا Not Only SQL معروف شده‌اند و به‌عنوان یکی از روش‌های اصلی ذخیره‌ی داده‌های بزرگ در دنیای فناوری اطلاعات مطرح هستند. امروزه بیش از ۱۲۰ بانک اطلاعاتی در زمره‌ی این گروه قرار گرفته‌اند.

وجود این بانک‌های اطلاعاتی به تحلیلگران این امکان را می‌دهد تا بدون درگیرشدن در جزئیاتی مانند مدل نگاشت کاهش، داده‌ها را ذخیره کرده و با امکاناتی که خود بانک‌های اطلاعاتی در اختیار آن‌ها می‌گذارند، به تحلیل داده‌ها بپردازند. بعضی از این بانک‌های نوین مانند کاساندرا و HBase می‌توانند همراه با هدوپ به‌کار گرفته شوند.

مشکل اصلی در استفاده از این بانک‌های اطلاعاتی علاوه بر نوپابودن و توسعه‌ی سریع آن‌ها، عدم پشتیبانی‌شان از مفاهیم تراکنش، جامعیت و سازگاری داده و استقلال عملیات است که به‌دلیل افزایش بهره‌وری و سرعت انجام گرفته است.

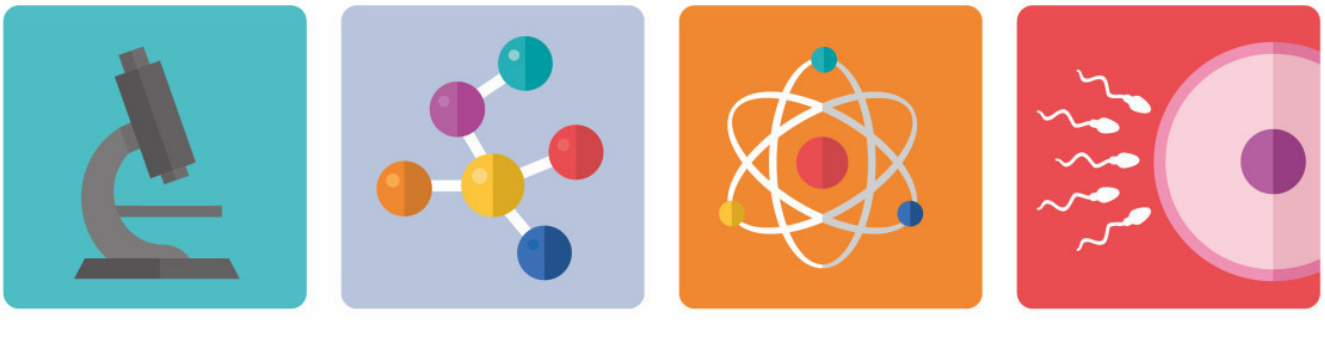


یادگیری ماشین

با همه‌ی این توصیفات داده‌های بزرگ با تمامی چالش‌های آن‌ها چیزی به‌جز دشواری محسوب نمی‌شوند و این همه داده سودی در پی نخواهد داشت. برای استفاده از این داده‌ها تنها راه استفاده از یادگیری ماشین است. یادگیری ماشین به ما این فرصت را می‌دهد تا بتوانیم تمامی این اطلاعات را بررسی کنیم و الگوهای مهم را از دل آن‌ها بیرون بکشیم. این یادگیری ماشین است که به مجموعه‌ی این داده‌ها معنا می‌بخشد و محصولاتی «جادویی» ایجاد می‌کند که سال‌ها به‌دنبال آن بوده‌ایم.

البته این دو فناوری به‌یکدیگر وابسته هستند و الگوریتم‌های یادگیری به کمک داده‌های بیشتر، به‌مراتب بهتر عمل می‌کنند. به‌عنوان مثال الگوریتم‌های یادگیری عمیق که با حجم عظیمی از داده‌ها آموزش یافته‌اند در کارهایی همچون درک بصری بسیار بهتر از روش‌های پیشین عمل می‌کنند، تا حدی که پس از سال‌ها تلاش محققان نتایج آن‌ها قابل مقایسه با انسان شده است.

۶- MapReduce



آینده‌ی زیست‌فناوری

حسین اخلاق‌پور صاحب بیش از ۱۸ سال تجربه در رهبری و هدایت اپلیکیشن‌های نرم‌افزاری در حیطه‌های تجارت الکترونیکی، امور مالی و صنعت مراقبت‌های بهداشتی در مقیاس سازمانی است. وی در پنج سال اخیر در زمینه‌های معماری محاسبات ابری با شرکت‌های ebay، ولتی و جنپسیس برای تولید پایپ‌لاین‌هایی برای پردازش داده‌های بزرگ همکاری داشته است. او همچنین مدیریت توسعه‌دهندگان مختلفی را از دور یا نزدیک بر عهده داشته و دو پروژه‌ی چندین میلیون دلاری را هدایت کرده است.

حسین اخلاق‌پور
hosseinakhlaghpour@gmail.com

بیست سال آینده برای زیست‌فناوری معادل سال‌های ۱۹۲۰-۱۹۴۰ برای فیزیک خواهد بود. همان‌طور که نسبیت و کوانتوم فیزیک را کاملاً متحول کرد، ژنتیک و نوروساینس هم علوم پزشکی، داروسازی، روان‌پزشکی، صنایع غذایی و حتی علم مواد را چنان متحول خواهد نمود که آیندگان روش‌های درمانی امروز را همچون روش‌هایی قرون وسطایی خواهند نگریست. دورانی که درمان‌ها به روش‌های سعی و خطا کشف می‌شد. این تحول امکان‌پذیر نخواهد بود مگر با کمک‌گرفتن از تکنولوژی داده‌های بزرگ.

قبل از هر چیز، کمی با اعداد و ارقام در این حوزه آشنا شویم.

- حجم اطلاعات ژنتیکی هر انسان در حدود ۱۰۰ گیگابایت است[۱] و اگرچه انسان‌ها از نظر ژنتیکی ۹۹٪ شبیه هم هستند، اطلاعات ژنتیکی همه انسان‌ها، گیاهان و باکتری‌ها در حدود چند ده زتابایت خواهد بود.
- کل حجم داده‌ی دیجیتال دنیا که تاکنون جمع‌آوری شده نزدیک به سه زتابایت تخمین زده می‌شود.[۲]
- تعداد سیناپس‌های مغز انسان حدود ۱۰۰ تریلیون و ظرفیت حافظه‌ی مغز حدود ۲٫۵ پتابایت تخمین زده می‌شود[۳]. به‌عبارتی ظرفیت مغزِ یک‌میلیون انسان معادل حجم تمام اطلاعاتی است که امروزه در دنیا نگهداری می‌شود.

این ارقام نشان می‌دهد که ژنتیک و به‌خصوص نوروساینس در چند دهه‌ی آینده در صدر جدول کاربردهای داده‌های بزرگ خواهد بود. بیماری‌ها یا منشأ ژنتیکی دارند، یا عفونی هستند یا ویروسی. لذا با شناخت ژنوتیپ انسان‌ها، باکتری‌ها و ویروس‌ها از یک طرف و فنوتیپ (خصوصیات و عوارض) انسان‌ها (سالم و بیمار) از طرف دیگر، می‌توان بروز یک بیماری را با دقت خوبی ریشه‌یابی کرد. به‌طوری که سال‌ها قبل ازظاهرشدن عوارض بیماری‌ای چون سرطان بتوان آن را تشخیص داد و درمان کرد. البته تحقیقات ده سال گذشته در پیداکردن منشأ ژنتیکی سرطان‌ها نشان داده است که این ارتباط می‌تواند بسیار پیچیده باشد؛ به‌طوری‌که بدون داشتن اطلاعات زیاد نمی‌توان به نتایج قابل اعتمادی رسید. در نتیجه شناخت و حل مسائل داده‌های بزرگ یکی از مهم‌ترین نقش‌ها را در آینده‌ی علم پزشکی دارد. پیش‌بینی می‌شود که بزرگ‌ترین مشکلات نه در نگهداری و حتی سرعت انتقال این اطلاعات، بلکه در پردازش و معنادادن به این حجم عظیم داده‌ها خواهد بود. از این رو تمرکز علومی چون داده‌کاوی و هوش مصنوعی در چند سال آینده به‌سرعت از تبلیغات آنلاین به زیست‌فناوری تغییر خواهد کرد.

قبل از آن‌که به جزئیات فنی نگهداری و پردازش این داده‌های بزرگ بپردازیم، بد نیست ترسیمی از کاربرد این علوم در چند دهه‌ی آینده داشته باشیم و چند مورد علمی-تخیلی را بررسی کنیم:

هر چند روز یک‌بار، شما قرصی را در آب حل می‌کنید و می‌نوشید. این قرص حاوی میلیاردها نانوبات است که از طریق مجرای گوارشی وارد خون می‌شود و هر یک مسئول نمونه‌برداری از قسمت خاصی از اعضای بدن است. به‌طور مثال یک نانوبات طوری طراحی شده که فقط به پروتئین خاصی که در کبد یافت می‌شود حساس است، و در نتیجه در مجاورت کبد سلول‌هایی از آن و محیط اطراف را به خود جذب کرده و در نهایت از طریق ادرار دفع می‌گردد. به این طریق هر چند روز یک‌بار تغیرات ژنتیکی کبد شما اندازه‌گیری شده و اگر اثری از رشد سلول سرطانی یا باکتری خاصی دیده شود، فرآیند درمان قبل از این‌که نشانه‌های بیماری در بدن بروز کند صورت می‌گیرد.[۴]

پزشک با شما تماس می‌گیرد و می‌گوید که باکتری خطرناک جدیدی در بدن شما مشاهده شده است. این باکتری که حاصل جهش ژنتیکی باکتری‌های ای-کُلای بوده، به آنتی‌بیوتیک‌های موجود مقاوم شده است (امروزه به این باکتری‌ها سوپرباگ می‌گویند). پزشک ژن جهش‌یافته را از باکتری جدا کرده و وارد یک تراشه‌ی زیستی[۵] می‌کند. این تراشه حاوی سلول‌هایی از سیستم لنفاوی شماست، به‌طوری که در آزمایشگاه می‌تواند مشابه سیستم دفاعی بدن شما عمل کند. محصول این فرآیند در انتها پادتن جدیدی است که اگر به بدن شما تزریق شود شما را نسبت به این باکتری مقاوم می‌کند و فرآیند درمان پیش از آن‌که عوارض بیماری ظهور کند انجام می‌شود.[۶]

همچنین پزشکی که این باکتری را تشخیص داد و آن را درمان کرد چیزی جز

۱ - http://www.pbs.org/newshour/rundown/storing-human-genome-cloud/
۲ - http://www.bigdatanews.com/profiles/blogs/a-comprehensive-list-of-big-data-statistics
۳ - http://www.scientificamerican.com/article/what-is-the-memory-capacity/

۴ - Based on Shawn M. Douglas,* Ido Bachelet and * George M. Church work at MIT "A Logic-Gated Nanorobot for Targeted Transport of Molecular Payloads" http://goo.gl/EnBHH4
۵ - Organ-on-Chip http://wyss.harvard.edu/viewpage/461/
۶ - Based on https://en.wikipedia.org/wiki/Immunotherapy

یک ماشین هوشمند نبود. امروزه پزشکان طبق آمار ۱۴٪ خطای تشخیص دارند[۷] و در مواردی ماشین‌های هوشمند بهتر از یک پزشک بیماری‌ها را تشخیص می‌دهند.[۸]

به‌وسیله‌ی مهندسی ژنتیک نوعی گیاه تولید شده است. در معرض نور خورشید بین ریشه و برگ‌های این گیاه اختلاف ولتاژ ایجاد می‌شود، به‌طوری که می‌توان از آن به‌عنوان باتری خورشیدی استفاده کرد.[۹] تصور کنید که باغچه‌ی شما انرژی الکتریکی خانه‌تان را تأمین می‌کند[۱۰]. در ضمن این گیاهان در شب نوری به رنگ دلخواه شما تولید می‌کنند.[۱۱]

قرصی وجود دارد که ویروس درون آن می‌تواند به‌مدت یک هفته به نورون‌های مخصوص حافظه‌ی کوتاه‌مدت شما متصل شود و در اثر یک میدان الکترومغناطیسی یون آزاد کند؛ به‌طوری که در این یک هفته شما با پوشیدن یک کلاه حاوی تراشه‌ی تحریک‌کننده‌ی ویروس‌ها می‌توانید حافظه‌ی خود را تقویت کنید. این ویروس کم‌کم شکسته شده و از طریق خون دفع می‌شود، اما اثر تقویت حافظه‌ی آن باقی خواهد ماند.[۱۲]

امروز مغز یک مورچه در یک تراشه‌ی زیستی جاسازی شد. این تراشه‌ی جدید معادل ابررایانه‌ای با ۲۰،۰۰۰ لایه‌ی یادگیری است[۱۳] و می‌تواند به‌سرعت الگوهای بینایی-شنوایی را تشخیص دهد.[۱۴] به‌دلیل حجم کم این تراشه‌ی زیستی و قدرت پردازش بالای آن، از این تراشه در ساعت‌های هوشمند استفاده می‌شود.

موارد ذکرشده اگرچه در نگاه اول غیرواقعی و تخیلی به‌نظر می‌رسند، تکنولوژی پایه‌شان در حال شکل‌گیری‌ست و در بازه‌ی ۱۰ تا ۳۰ سال آینده کاملاً امکان‌پذیر خواهند بود.

تمرکز ادامه‌ی این مقاله بر فناوری‌های مورد استفاده در اکتساب، پردازش و تحلیل داده‌های بزرگ در زمینه‌ی زیست‌فناوری است.

جمع‌آوری داده[۱۵]:

تقریباً تمامی داده‌ها در صنعت زیست‌فناوری از حسگرها تولید می‌شود. این مورد تفاوت اساسی با رسانه‌های اجتماعی دارد که داده‌ی اصلی از کاربر گرفته می‌شود. به‌عبارتی، زیست‌فناوری درجمع‌آوری داده بسیار شبیه به اینترنت اشیا است. این حسگرها، که اکثراً نوری هستند، حجم بسیار زیادی داده تولید می‌کنند که نگهداری و انتقال این داده‌های خام مقرون به صرفه و گاهی امکان‌پذیر نیست. در نتیجه داده‌ی خام غالباً به‌صورت فایل در محل خودِ حسگر برای مدت محدودی نگهداری می‌شود و لازم است به‌سرعت و حتی به‌صورت هم‌زمان[۱۶] نویززدایی شود. به همین دلیل الگوریتم‌های پردازش سیگنال در این مرحله کاربرد زیادی دارد. فرمت‌های معروف برای نگهداری داده در این مرحله، CSV، JSON و HDF5 هستند. از آن‌جا که حجم داده زیاد است، برای حذف داده‌های قدیمی به دستورات دوره‌ای[۱۷] نیاز داریم. در ضمن در اغلب دستگاه‌ها ما با بیش از یک حسگر سروکار داریم. تمامی این حسگرها باید بسته به محیط تنظیم شوند و تطابق زمانی اطلاعات آن‌ها بسیار مهم است. حسگرهای دما، صدا، نور، فشار، لرزش، سونوگرافی، مادون قرمز، شتاب‌سنجی و OCT[۱۸] از جمله حسگرهایی هستند که در دستگاه‌های زیست‌فناوری استفاده می‌شوند.

تقلیل داده‌ها[۱۹] یا نویززدایی:

در این مرحله بر اساس مدل پیشنهادی سیگنال و حسگرها، نویز شناسایی و از داده حذف می‌شود. به این طریق حجم داده به‌میزان قابل ملاحظه‌ای کاهش می‌یابد. این نویزها یا تصادفی هستند که معمولاً از طریق روش‌های آماری حذف می‌شوند یا نظام‌مند که از طریق روش‌های کنترلی و پردازش سیگنال جدا می‌شوند. البته گرایش پژوهشگران از روش‌های سنتی پردازش سیگنال مثل فیلترهای فرکانسی به‌سمت استفاده از روش‌های داده‌کاوی و یادگیری ماشین همچون طبقه‌بندی از طریق PCA، k-means و حتی یادگیری عمیق[۲۰] رفته است.

انتقال داده‌ها[۲۱]:

به‌دلیل این‌که حسگرها در اغلب موارد به‌صورت دائم داده تولید می‌کنند و حجم داده‌ها بیش از آن است که بتوان آن را ذخیره کرد، امروزه داده نه به‌صورت دوره‌ای، بلکه به‌صورت جریانی منتقل می‌شود. به‌طوری که در حین این انتقال نیز عمل تقلیل داده انجام می‌شود. امروزه چهارچوب‌های تکنولوژیکی شکل گرفته‌اند که ضمانت این انتقال را برای حجم بالا، بدون ازدست‌دادن داده، در شرایطی که حتی اتصال و سرعت شبکه قابل اطمینان نیست می‌دهند.[۲۲] از جمله‌ی این چهارچوب‌ها (به‌ترتیب شهرت) می‌توان اسپارک استریمینگ، کافکا و فلوم را نام برد. تمام این چهارچوب‌ها اجازه می‌دهند که در حین انتقال، داده پردازش و حتی تفسیر شود.

نگهداری داده[۲۳]:

تا کنون یکی از متداول‌ترین روش‌های نگهداری داده‌ها به‌صورت فایل استفاده از سیستم هدوپ بوده و اجرای نگاشت‌کاهش جهت پاسخ‌گویی هر گونه درخواست بوده است.[۲۴]. به‌دلیل پرهزینه‌بودن عملیات نگاشت‌کاهش، بسیاری از افراد توانسته‌اند با استفاده ازحجم بالای حافظه و تغییر فرمت داده به فرمت کلید-مقدار سرعت محاسبات و پاسخ به درخواست‌ها را به‌میزان قابل توجهی افزایش دهند و موتورهای پرس‌وجوی جدیدی چون اسپارک و پرستو و پایگاه‌داده‌های رابطه‌ای همچون Hbase، کاساندرا، پارکه، ایمپالا و ایروسپایک به بازارعرضه کنند. این پایگاه‌داده‌های جدید امکان تفسیر هم‌زمان را نیز - که یکی از موضوعات روز صنعت داده‌های بزرگ است - فراهم کرده‌اند.

تطابق داده[۲۵]:

یکی از مهم‌ترین مراحل معنادادن به داده از طریق تطابق و ارتباط داده‌ها از منابع متفاوت است؛ به‌طور مثال تطابق عوارض یک بیماری به جهش‌های ژنتیکی. روش‌های یادگیری ماشینی نقش اصلی را در این مرحله بازی می‌کنند. لازمه‌ی یک تحلیل درست داشتن مقدار زیادی داده از منابع متفاوت است. مثلاً تطبیق داده‌ی ژنوتیپ و فنوتیپ بیماری‌ای چون سرطان احتیاج به اطلاعات هزاران و گاهی میلیون‌ها نفر انسان بیمار و سالم دارد.

امنیت داده[۲۶]:

بزرگ‌ترین مانع پیشرفت تحلیل داده‌های بزرگ در دنیای پزشکی حل‌نشدن درست مسئله‌ی امنیت داده است. به‌طور مثال، قانون هیپا[۲۷] و ترس از تعقیبات قانونی در پی به‌اشتراک‌گذاری داده‌های بیماران تقریباً هر پروژه‌ی جدی ژنتیکی تحلیل بیماری را در نطفه خفه می‌کند.[۲۸] در دهه‌ی اخیر جریان‌هایی اجتماعی شکل گرفته است که در آن‌ها بیماران و حتی مردم سالم داوطلبانه اطلاعات ژنتیکی خود را در اختیار محققین و گاهی در دسترس عموم قرار می‌دهند.[۲۹] حل مشکلات زیربنایی امنیت و حریم خصوصی داده‌های پزشکی مسئله‌ای بسیار بغرنج است، چرا که هیچ داده‌ای به‌اندازه‌ی اطلاعات ژنتیکی نمی‌تواند متمایزکننده‌ی یک شخص باشد. در نتیجه حتی اگر اطلاعات شخصی (مانند نام و آدرس) از داده‌ی ژنتیکی جدا شود، به‌سادگی می‌توان صاحب داده را شناسایی کرد.[۳۰]

در خاتمه خوانندگان را به اسلایدهایی که در اردی‌بهشت ۹۴[۳۱] برای دانشگاه نورث‌وسترن ارائه دادم وهمچنین کنفرانسی که در این زمینه در دانشگاه استنفورد برگزار شده است ارجاع می‌دهم.[۳۲]

۷ - Statistics on malpractice http://www.medicalnewstoday.com/articles/248175.php
۸ - Watson's successful diagnosis rate for lung cancer is 90 percent, compared to 50 percent for human doctors http://goo.gl/U8HPVM
۹ - Ted talk: Using Nature to grow batteries http://goo.gl/NRxhC۲
۱۰ - Based on "Hack Tobacco Plants to Grow Synthetic Photovoltaic Cells" http://goo.gl/DSPnSg
۱۱ - Based on GloFish https://en.wikipedia.org/wiki/Genetically_modified_organism
۱۲ - Based on Optogenetics http://video.mit.edu/watch/optogenetics-controlling-the-brain-with-light-7659/
۱۳ - Google's current neural networks are now more than 30 layers deep, 16000 nodes and Ant brain has 250,000 neurons http://goo.gl/qucYvz, http://goo.gl/Z4hBV1, https://goo.gl/57kojL
۱۴ - Based on "Computers and Neurons Unite" http://goo.gl/fZdpS3
۱۵ - Data collection
۱۶ - Real-time
۱۷ - Batch process or cron jobs
۱۸ - https://en.wikipedia.org/wiki/Optical_coherence_tomography
۱۹ - Data reduction
۲۰ - Deep learning
۲۱ - Data streaming
۲۲ - Recent Evolution of Zero Data Loss Guarantee in Spark Streaming With Kafka http://goo.gl/NIAJxM
۲۳ - Data storage
۲۴ - Query of Hadoop file system starts a mapreduce job that may take minutes, if not hours
۲۵ - Data matching
۲۶ - Data security
۲۷ - HIPPA
۲۸ - To make all health data public has big benefits and big risks http://goo.gl/۹OXmBY
۲۹ - Dynamic Consent Open Framework http://dynamic-consent.info/
۳۰ - Scientists expose new vulnerabilities in the security of personal genetic information http://goo.gl/cH۶af۶
۳۱ - Nextgen DNA Sequencing, a disruptive technology https://goo.gl/YSJr۴o
۳۲ - http://bigdata.stanford.edu/

پردازش جریان داده

امیر صدیقی
sedighi@gmail.com

امیر صدیقی کارشناس ارشد نرم‌افزار است. وی با تجربه‌ی طولانی در توسعه‌ی سامانه‌های داده‌ای[۱] و سازمانی[۲]، در حوزه‌ی پردازش داده‌های بزرگ و جریان‌داده، راه‌حل‌هایی نوآور[۳]، توسعه داده‌شده، یا در حال توسعه دارد. در سه پروژه‌ی تألیف کتاب پیرامون اسپارک[۴] با یک ناشر بین‌المللی[۵]، به‌عنوان بازبین فنی[۶] همکاری دارد. پیش‌تر نیز در توسعه‌ی یک راه‌حل صنعتی پیشنهاددهنده[۷] و یک راه‌حل مدیریت گزارش[۸]، از تکنیک‌ها و ابزارهای پردازش جریان‌داده و یادگیری ماشین بهره گرفته است. در یک طرح تحقیقاتی برای ذخیره و بازیابی جریان‌های بزرگ داده در یک کتاب‌خانه‌ی دیجیتال[۹]، به پیاده‌سازی مدل اولیه توسط هدوپ پرداخته است. در گذشته‌ی دورتر نیز، فناوری‌های متن‌باز دنیای جاوا[۱۰] را برای پردازش جریان‌های بزرگ گزارش[۱۱] به‌کار گرفته است.

twitter:@amirsedighi

http://www.linkedin.com/in/amirsedighi

۱ - Data Centric
۲ - Enterprise
۳ - http://rayd.ir/solutions/
۴ - Spark Cookbook - Fast Data Processing with Spark _ Graph Processing using Spark
۵ - PACKT Publishing
۶ - Technical Reviewer
۷ - Http://recommender.ir
۸ - Log Management
۹ - NLAI
۱۰ - http://mvnrepository.com/artifact/org.tigris.jsapar
۱۱ - http://blog.hexican.com/2010/02/parsing-huge-text-files-using-java-and-jsapar/

مقدمه

در این مقاله، به معرفی برخی رویکردها، چهارچوب‌ها[۱] و ابزارهای دنیای پردازش جریان داده[۲] می‌پردازیم. در این راستا، نخست نگاهی به تاریخچه‌ی تکامل فناوری‌های پردازش جریان داده می‌اندازیم. سپس چند رویکرد متداول را بررسی و مقایسه خواهیم کرد. در انتها برخی ابزارهای متداول و ترفندهای پردازش جریان داده را معرفی می‌کنیم.

تاریخچه

سرآغاز بسیاری از نوآوری‌های دنیای پردازش اطلاعات به یونیکس بازمی‌گردد. با مراجعه به تاریخچه‌ی پرافتخار یونیکس، درمی‌یابیم پردازش جریان داده اساساً یک فناوری جدید نیست. توسعه‌ی مفهوم پایپ‌لاین و ابزار sed[۳] در یونیکس به قدمت نیاز به فناوری پردازش جریان داده در دنیای کامپیوتر اشاره دارد. برخورد کلاسیک و انتزاعی[۴] سیستم‌عامل‌های شبه‌یونیکس[۵] با جریان داده همچنان پرکاربرد است. یونیکس به فایل و جریان داده نگاهی یکسان دارد. این نگاه انتزاعی در سامانه‌های پیشرفته‌ی امروزی نیز همچنان راه‌گشاست.

با فراگیرشدن اینترنت، افزایش چشمگیر تعداد کاربران شبکه‌های اجتماعی و تنوع محتوا هنجارهای پردازش داده را تغییر داد. شبکه‌های اجتماعی محبوب، همچون توییتر، لینکداین و اینستاگرام، بستر انتشار نظرات ساکنین این سیاره شده‌اند. اشاره به توییتر خالی از لطف نیست. این شبکه ۳۵۰میلیون کاربر دارد. در صورتی که هر کاربر به‌طور متوسط در طول شبانه‌روز سه بار توییت کند، آن‌گاه در هر ثانیه ۱۲۵۰۰ توییت به‌سمت سرورهای توییتر ارسال می‌شود. از سوی دیگر، انعکاس تغییرات باید برای دنبال‌کنندگان[۶] و گیرندگان توییت نیز ارسال شود. پس این جریان چندین برابر می‌گردد. در سال ۲۰۰۹ در پی فوت خواننده‌ی مشهور آمریکایی، مایکل جکسون، سرورهای توییتر زیر فشار ۱۰۰هزار توییت در ثانیه از کار افتادا!

گسترش روزافزون ابعاد کسب‌وکار شرکت‌های بزرگ وب نیازهای جدید و متنوعی نسبت به پردازش جریان داده به‌وجود آورد. هر آن‌چه تا آن زمان توسعه یافته بود دیگر نمی‌توانست پاسخگوی نیازهای روبه‌توسعه‌ی شرکت‌های وب در حوزه‌ی پردازش داده و جریان داده باشد. مدل‌های پردازش داده که تا آن زمان مورد استفاده بود، به‌واسطه‌ی ضعف در مقیاس‌پذیری[۷]، کافی نمی‌نمودند. اقبال روزافزون شبکه‌های اجتماعی و افزایش تعداد کاربران منجر به توسعه‌ی زیرساخت‌های مخابراتی، شبکه و مدل‌های پردازش داده شد. در این بین شرکت‌های بزرگ وب بیشترین تغییرات را به‌وجود آوردند. در این زمان، تغییری بزرگ در رویکرد به توسعه‌ی زبان‌های برنامه‌نویسی در حال شکل‌گیری بود.

۱ - Frameworks
۲ - Stream processing
۳ - https://en.wikipedia.org/wiki/Sed
۴ - Abstract
۵ - Unix-like
۶ - Followers
۷ - Scalability

همزمان با نیاز روزافزون به مدل‌های مقیاس‌پذیر پردازش داده و محبوبیت پردازش ابری، از محبوبیت شیءگرایی به‌عنوان مدلی مناسب برای دنیای داده‌های بزرگ کاسته شد. شیءگرایی، با تمام مزایایی که به‌همراه آورده بود، برای پردازش حجم انبوه داده مناسب نمی‌نمود. بهای تسهیلاتی که زبان‌های شیءگرا برای برنامه‌نویسان فراهم می‌سازند افزون‌شدن پیچیدگی در لایه‌های زیرین و افزایش بار زمان اجرا است. سنگینی و زمان اجرای طاقت‌فرسای برنامه‌هایی که با اشیاء بزرگ و پیچیده‌ی شیءگرا توسعه یافته‌اند در هنگام اجرا بر روی سرورها مشخص می‌شود. از سوی دیگر، هزینه‌ی زباله‌روبی[۸] انبوه اشیاء پیچیده‌ای که به‌سادگی تولید کرده و برای زباله‌روب[۹] رها می‌کنیم، با افزون‌شدن بار پردازشی و تعداد اشیاء زنده در حافظه، به مشکلی پیچیده بدل می‌شود. لذا شیءگرایی، که زمانی راهکاری مناسب در توسعه‌ی سامانه‌های نرم‌افزاری می‌نمود، با ورود به عصر داده‌های بزرگ به راهکاری ناکارآمد بدل شد.

برای رهایی از پیچیدگی‌های شیءگرایی، بازگشتی هوشمندانه به‌سوی زبان‌های برنامه‌نویسی تابعی[۱۰] آغاز شد[۱۱]. بسیار پیش‌تر در سال ۱۹۵۰، جان مک‌کارتی[۱۲] با معرفی زبان پیشرفته‌ی لیسپ[۱۳] دنیای زبان‌های تابعی را گشوده بود.

در دهه‌ی گذشته، زبان‌های تابعی محبوبیت بسیاری یافته‌اند. این حرکت به توسعه‌ی زبان‌هایی مانند کلوژر[۱۴] (مشتق شده از لیسپ) و اسکالا[۱۵] منتهی شد. اسکالا مزایای زبان‌های تابعی و شیءگرا را توأماً عرضه می‌کند[۱۶]. هر دوی این زبان‌ها روی ماشین مجازی جاوا[۱۷] به‌خوبی کار می‌کنند. بی‌شک پدیدآمدن زبان‌های نوین تابعی توسعه‌ی چهارچوب‌های نوین پردازش داده و جریان داده را تسهیل کرد.

در این دوره چهارچوب‌هایی آوانگارد و مدرن همچون آپاچی استورم[۱۸] و آپاچی سامزا[۱۹] متولد شدند. استورم توسط ناتان مرز[۲۰]، بر پایه‌ی کلوژر و جاوا توسعه یافت. البته زبان‌های غیر JVM[۲۱] دیگری را نیز پشتیبانی می‌کند[۲۲]. ناتان مرز یکی از ایده‌پردازان و معماران فناوری‌های جریان داده است. وی از توسعه‌دهندگان معماری لاندا[۲۳] است. توییتر در سال ۲۰۱۱ این پروژه را خریداری و در توسعه‌ی آن سرمایه‌گذاری کرد. استورم تا چندی پیش به‌عنوان چهارچوب اصلی پردازش جریان داده‌ی توییتر به‌کار گرفته می‌شد. استورم از چهارچوب ZeroMQ[۲۴] به‌عنوان بستر ارتباطی داخلی خود بهره می‌گیرد. در ابتدای سال ۲۰۱۵ توییتر استورم را

۸ - Garbage Collection
۹ - Garbage Collector
۱۰ - Functional Programming Languages
۱۱ - http://www.ibm.com/developerworks/library/j-ft۲۰/
۱۲ - John MacCarthy
۱۳ - Lisp
۱۴ - Clojure
۱۵ - Scala
۱۶ - http://cacm.acm.org/magazines/2014/4/173220-unifying-functional-and-object-oriented-programming-with-scala/fulltext/
۱۷ - Java Virtual Machine
۱۸ - Apache Storm
۱۹ - Apache Samza
۲۰ - Nathan Marz
۲۱ - Java Virtual Machine
۲۲ - http://samza.apache.org/learn/documentation/0.7.0/comparisons/storm.html
۲۳ - http://lambda-architecture.net/
۲۴ - http://zeromq.org/

به‌دلایلی[۲۵] همچون عدم ارائه‌ی سازوکار اضافه‌بار[۲۶]، ضعف در کارایی و ناکارآمدی معماری کنار گذاشت[۲۷] و به توسعه‌ی هرون[۲۸] روی آورد. استورم در حالی بازنشسته شد که به بلوغ دست نیافته بود. با این حال تیم‌های بسیاری به‌عنوان چهارچوب اصلی پردازش جریان داده از آن بهره می‌برند[۲۹].

در سوی دیگر، لینکداین در توسعه‌ی سامزا از زبان‌های اسکالا و جاوا بهره گرفت. در حالی که ایده‌های بنیادین سامزا از استورم برگرفته شده است، سامزا در حرکتی هوشمندانه سعی بر حفظ سازگاری با اکوسیستم هدوپ[۳۰] دارد. اختلافاتی دیگر نیز در معماری این دو به‌چشم می‌خورد. سامزا در مدیریت وضعیت راهکار بهتری[۳۱] برگزید. در حالی که استورم مکانیزم انزوای سطح فرآیند[۳۲] یونیکس را عرضه می‌کند، سامزا از یارن[۳۳] برای ارائه‌ی انزوای سطح منبع[۳۴] بهره برده است. بنابراین پیشرفت‌هایی که در سازوکارهای یارن و خدمات آن به‌وقوع پیوست منجر به بهبود کارکردهای داخلی سامزا شد.

لینکداین در کنار توسعه‌ی سامزا، پروژه‌ی بسیار موفق دیگری به‌نام آپاچی کافکا[۳۵] را نیز معرفی کرد. کافکا یک کارگزار پیام[۳۶] مقیاس‌پذیر است که در دنیای پردازش داده‌های بزرگ و جریان داده محبوبیت بسیاری کسب نموده است. این محصولات که بر اساس نیاز روبه‌توسعه‌ی شرکت‌های بزرگ وب طراحی و تولید شدند، با مقیاس‌پذیر نمودن مدل‌های پردازش جریان داده بسیاری از محدودیت‌ها را کنار زدند.

از سوی دیگر، چهارچوب محاسباتی نگاشت‌کاهش[۳۷] که توسط گوگل استفاده می‌شد، و به‌وسیله‌ی چهارچوب محبوب و انقلابی هدوپ عمومی شده بود، به هنجاری بی‌مانند برای پردازش داده‌های بزرگ بدل شد. در حالی که نگاشت‌کاهش برای پردازش دسته‌ای[۳۸] مناسب می‌نمود، بهبودهایی که بر روش اجرای آن حاصل آمد منجر به توسعه‌ی سامانه‌های نوین پردازش جریان داده مبتنی بر مدل توزیع‌شده‌ی نگاشت‌کاهش گردید.

آپاچی هدوپ با تمرکز بر به حداکثر رساندن توان اجرایی پردازه‌های نگاشت‌کاهش بر روی ماشین‌هایی با دیسک‌های مکانیکی و حافظه‌ی محدود توسعه یافته بود. طی سال‌های اخیر تیم‌هایی نوآور از شرکت‌هایی همچون مپ‌آر[۳۹] و دیتابریکس[۴۰]، با هدف افزایش سرعت اجرای پردازه‌های مبتنی بر نگاشت‌کاهش، مدل محاسباتی هدوپ را دستخوش تغییراتی وسیع نموده‌اند. این تغییرات بر استفاده‌ی بهتر از حافظه‌ی رَم و استفاده‌ی بهینه از چرخه‌های نگاشت‌کاهش تمرکز دارد. نوآوری‌هایی که روی نگاشت‌کاهش انجام پذیرفت، با ارزان‌شدن فناوری، بر فراگیرشدن استفاده از SSD[۴۱] نیز تکیه دارد. بهبود مدیریت منابع خوشه[۴۲] توسط یارن و آپاچی مزوز[۴۳] نیز در افزایش کارایی نگاشت‌کاهش مؤثر بود.

آپاچی اسپارک[۴۴] بهترین نمونه از این دست است. اسپارک با زبان پیشرفته و تابعی اسکالا توسعه یافته است. این پروژه که در ابتدا در یکی از آزمایشگاه‌های دانشگاه برکلی توسط ماتی زهریا[۴۵] توسعه یافت اکنون به‌عنوان بخشی از اکوسیستم هدوپ کارایی کم‌نظیری در اجرای پردازه‌های نگاشت‌کاهش ارائه می‌کند[۴۶]. اسپارک با فراهم‌آوردن چهارچوبی ویژه برای پردازش جریان داده به‌نام اسپارک استریمینگ[۴۷] توان پردازشی خود را در اختیار کارکردهای پردازش جریان قرار می‌دهد. اسپارک استریمینگ از هسته‌ی آپاچی اسپارک بهره می‌گیرد. رویکرد اسپارک استریمینگ اساساً با سامزا و دنباله‌رواش، استورم، متفاوت است. آن‌ها هر پیام دریافتی را پردازش می‌کنند، حال آن‌که اسپارک استریمینگ پیام‌های دریافتی را با رویکردی دسته‌ای مورد پردازش قرار می‌دهد.

نام‌های گوناگون

از پردازش «جریان داده» با نام‌های متفاوتی یاد شده است. برخی آن را پردازش جریان خوانده‌اند. برخی از آن با رویداد سپاری[۴۸] یا CQRS[۴۹] نام برده‌اند. حتی برخی آن را پردازش رویداد پیچیده[۵۰] نامیده‌اند. همان‌طور که اشاره شد، ابزارهای توزیع‌شده‌ی پردازش جریان داده از شرکت‌های اینترنتی همچون لینکداین و توییتر سربرآورده‌اند. این اتفاقات از سال ۲۰۰۰ شروع شده بود. از سوی دیگر پردازش رویداد پیچیده در دانشگاه استنفورد طی پروژه‌ی رپید[۵۱] در سال ۱۹۹۰ متولد شد. رویداد سپاری ریشه در نگاهی دامنه‌محور[۵۲] به توسعه‌ی نرم‌افزارهای سازمانی[۵۳] با مدل‌های داده‌ای بسیار پیچیده، ولی در ابعادی کوچکتر از شرکت‌های وب، دارد. درواقع خاستگاه‌های گوناگون و نیازهای متفاوتی که برای این فناوری وجود داشته منجر به این تنوع نام‌گذاری شده است. از نظر نگارنده فناوری پردازش جریان داده همه‌ی فناوری‌های یادشده را در برمی‌گیرد.

فناوری پردازش جریان داده

در این‌جا سعی بر معرفی فناوری پردازش جریان داده داریم و سعی می‌کنیم پیرامون چگونگی استفاده از این فناوری در راستای افزایش مقیاس‌پذیری، قابلیت اطمینان[۵۴] و قابلیت نگهداری[۵۵] بحث کنیم. همان‌گونه که در مقدمه اشاره شد، ایده‌ی سامان‌دهی داده در قالب جریانی از رویدادها موضوعی جدید نیست و پیشینه‌ی آن به نگاه بدیع و نبوغ‌آمیز پدیدآورندگان یونیکس به فایل باز می‌گردد. کن تامپسون[۵۶] در سال ۱۹۷۳ پایپ‌لاین را در یونیکس نسخه‌ی ۳ پیاده‌سازی کرد. وی از ایده‌ی داگلاس مک‌ایلروی[۵۷] بهره گرفت.

در حالی که نمی‌توان از فناوری پردازش جریان داده به‌عنوان یک فناوری نوظهور یاد کرد، چشم‌پوشی از تغییرات و پیشرفت‌های شگرف آن طی دهه‌ی گذشته دور از انصاف است.

مهم‌ترین پیشرفت‌های این فناوری در پاسخ‌گویی به نیازهای عصر داده‌های بزرگ حاصل شده است. نیاز شرکت‌های بزرگ به مقیاس‌پذیری موتور پیش‌برنده‌ی این صنعت است. مهاجرت به زبان‌های تابعی نیز گامی بزرگ در جهت بهبود کارایی و پایداری این سامانه‌ها بود. در نهایت افزایش کارایی مدل محاسباتی نگاشت‌کاهش و پدیدآمدن چهارچوب‌های بسیار نوآور همچون اسپارک، این فناوری را به بلوغ رسانده است.

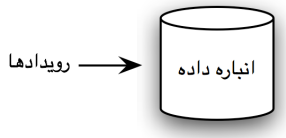
پردازش و تحلیل رویدادها به‌منظور حصول بینش[۵۸]

یکی از پرکاربردترین سناریوهای استفاده از فناوری پردازش جریان داده پردازش رویداد[۵۹] به‌منظور تحلیل‌های کامل بعدی است. جریان رویدادها حاوی اطلاعات گوناگون در حوزه‌ی کسب‌وکار است. در این مدل پردازش داده رویدادها از منابع گوناگون جمع‌آوری شده و برای پردازش به سرویس‌های محاسبه‌گر ارسال می‌شوند. این راهکار با گردآوری و پردازش جریان رویدادها در بخش‌هایی از کسب‌وکار می‌تواند مدیران سازمان و مسئولان سامانه‌ها را نسبت به آن‌چه در گذر است مطلع سازد. این تکنیک‌ها در کسب‌وکارهای مدرن آنلاین کاربردهای بسیاری دارند.

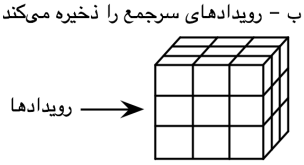
به‌طور کلی در پردازش جریان رویداد، دو رویکرد پایه‌ی زیر وجود دارد:

- الف- ذخیره‌سازی رویدادهای خام[۶۰]
- ب- ذخیره‌سازی داده‌های سرجمع[۶۱]

الف - رویدادهای خام را ذخیره می‌کند



ب - رویدادهای سرجمع را ذخیره می‌کند



۲۵ - http://dl.acm.org/citation.cfm_id=2742788
۲۶ - Backpressure
۲۷ - http://www.infoq.com/news/2015/06/twitter-storm-heron
۲۸ - http://blog.twitter.com/2015/flying-faster-with-twitter-heron
۲۹ - http://samza.apache.org/learn/documentation/0.7.0/comparisons/storm.html
۳۰ - Hadoop Ecosystem
۳۱ - http://samza.apache.org/learn/documentation/0.7.0/comparisons/introduction.html
۳۲ - Procss-Level Isolation
۳۳ - YARN
۳۴ - Resource-Level Isolation
۳۵ - Apache Kafka
۳۶ - Message Broker
۳۷ - MapReduce
۳۸ - Batch Processing
۳۹ - MAPR
۴۰ - Databricks
۴۱ - Solid-State Drive
۴۲ - Cluster
۴۳ - Apache Mesos
۴۴ - Apache Spark
۴۵ - Matei Zaharia
۴۶ - http://spark.apache.org/news/spark-wins-daytona-gray-sort-100tb-benchmark.html
۴۷ - Spark Streaming
۴۸ - Event Sourcing
۴۹ - Command Query Responsibility Segregation
۵۰ - Complex Event Processing
۵۱ - http://complexevents.com/stanford/rapide/
۵۲ - Domain-Driven
۵۳ - Enterprise Businesses
۵۴ - Reliability
۵۵ - Maintainability
۵۶ - Ken Thompson
۵۷ - Douglas McIlroy
۵۸ - Insight
۵۹ - Event Processing
۶۰ - Store Raw Events
۶۱ - Store Aggregated Data

الف- ذخیره‌سازی رویدادهای خام

در روش الف داده‌هایی که به‌شکل رویداد دریافت شده است عیناً یا با کمترین تغییرات در یک پایگاه داده با مقیاس‌پذیری بالا ذخیره می‌شود. از این پس قادر به اجرای پرسش‌های گوناگون و عملیات روی انبار داده[۶۲] خواهیم بود. این مدل در صورت وجود فضای ذخیره‌سازی کافی مدلی مناسب است. این مدل برای دریافت و نگهداری از رویدادها برای زمان‌های طولانی و مراجعات بعدی مناسب است. همچنین مدل پردازش داده در مدل الف مدل‌های غیربرخط است که توانایی پردازش داده به‌صورت یک جریان را نداشته و از همان ابتدای پردازش به تمامی داده‌ها نیاز دارد. این مدل عموماً برای پردازش‌های دسته‌ای یا مدل محاسباتی نگاشت‌کاهش مناسب است. قابلیت انعطاف بسیار بالا برای تحلیل‌های بعدی نیز از مزایای این روش است. داده‌های ذخیره‌شده در این روش می‌توانند بعدها نیز توسط الگوریتم‌ها و تکنیک‌های متنوع مورد پردازش قرار گیرند.

ب - ذخیره‌سازی داده‌های سرجمع

از روش ب در موقعیت‌هایی بهره می‌گیریم که ذخیره‌ی تمام جزئیات داده پرهزینه و ناممکن باشد. در این روش به‌جای ذخیره‌سازی داده‌های خام، به‌محض دریافت داده، محاسبات مورد نیاز بر روی داده انجام پذیرفته و به ذخیره‌ی نتیجه اکتفا می‌شود. برای مثال، در صورتی که به شمارش رویدادهایی می پردازیم، این روش بسیار مناسب است. چرا که کافی است مقادیر شمارنده‌ها را افزایش دهیم. در صورتی که تعداد شمارنده‌ها و متغیرهایی که به نگهداری مقادیر می‌پردازند به اندازه‌ی ابعاد مورد نیاز در کسب‌وکار باشد، پایگاه داده در این مدل می‌تواند همچون یک مکعب OLAP[۶۳] در نظر گرفته شود. یک مکعب چند بعدی را تصور کنید که ابعادی برای نام کاربری، جنسیت، زمان، موقعیت جغرافیایی، بردار علایق، آخرین مراجعات و ابعادی از این دست باشد. این ابعاد نقطه‌ای در فضای چندبعدی را مشخص می‌کنند. برای هر رویداد جدید تنها کافی است مختصات مربوطه به‌روز شوند. درواقع اثر رویدادها را محاسبه و نگهداری می‌کنیم؛ پس به نگهداری سیاهه‌ی اتفاقات قبلی نیازی نیست، چرا که همواره موقعیت فعلی را در دسترس داریم.

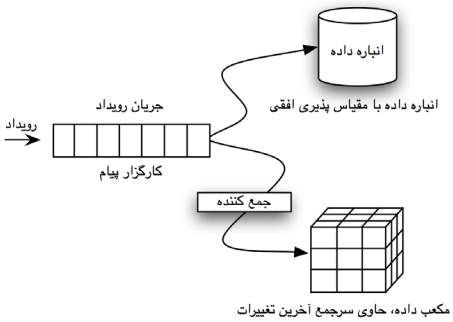
دسترسی به داده‌های مکعب بسیار سریع و کم‌هزینه است. در صورتی که مکعب در حافظه‌ی نهان[۶۴] نگهداری شود، دسترسی به مقادیر ابعاد و به‌روزآوری آن‌ها به اندازه‌ای سریع خواهد بود که راه‌حل بتواند به انبوه کاربران برخط پاسخ دهد.

در حالی که رویکرد الف در برابر رویکرد ب چندان جذاب و راه‌گشا به‌نظر نمی‌رسد، این روش نیز بسیار پرکاربرد است. در راه‌حل‌هایی همچون فارنزیک[۶۵] و سامانه‌های مدیریت گزارش، نیازمند تحلیل‌های دقیق با مستندات و دلایل کافی بر انبوه داده‌های قدیمی هستیم. از سوی دیگر در رویکرد الف، انبوه داده‌ی خامی که امروز در دسترس داریم می‌تواند توسط الگوریتم‌هایی که در آینده به‌دست خواهیم آورد نیز مورد پردازش قرار گیرد. این یعنی ما همواره قادر خواهیم بود اتفاقاتی که توسط حسگرها در قالب رویدادهای متنوع در گذشته جمع‌آوری شده‌اند را با آخرین روش‌ها بررسی کنیم. این رویکرد در استفاده از روش‌های یادگیری ماشین[۶۶] بسیار کارآمد و پرکاربرد است.

۶۲ - Data Warehouse
۶۳ - Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab
۶۴ - Cache
۶۵ - Forensic

معماری ترکیبی[۶۷]

به‌منظور بهره‌گیری از مزایای هر دو رویکرد الف و ب، می‌توانیم از یک معماری ترکیبی بهره بگیریم. فرض کنید جریانی از رویدادهای مربوط به کسب‌وکار را از طریق صفی از رویدادها دریافت می‌کنید. بدون شک برای استفاده از مزایای هر دور رویکرد الف و ب، ناچاریم رویدادها را به‌طور اشتراکی در اختیار هر دو رویکرد قرار دهیم. استفاده از یک چهارچوب کارگزار



پیام، همچون کافکا، قابلیت‌های متنوعی را در هدایت و مدیریت جریان رویداد به راه‌حل شما می‌افزاید. فرض کنید هر پیام حاوی داده‌هایی مربوط به یک رویداد منحصربه‌فرد است. کافی است تعداد پیامگیر[۶۸] رویداد را افزایش دهید. به این ترتیب، یک رویداد می‌تواند به‌طور هم‌زمان به‌صورت اشتراکی در اختیار چند پیامگیر قرار گیرد.

در ساده‌ترین شکل، معماری ترکیبی ما شامل یک پیامگیر برای ذخیره‌سازی رویدادهای خام، و پیامگیر دیگری برای محاسبه و ذخیره‌سازی سرجمع رویداد خواهد بود.

به‌طور معمول در این معماری، مخزنی که داده‌ی خام را نگهداری می‌کند بیشتر مورد عملیات نوشتن[۶۹]، و حافظه‌ی نهان، که حاوی نتایج محاسبات است، بیشتر مورد عملیات خواندن[۷۰] قرار می‌گیرد. توجه به این نکته ضروری است. رویدادهایی که به‌صورت خام در انبار مقیاس‌پذیر نگهداری می‌کنیم می‌توانند بعدها منبع عملیات دیگر شوند. این داده‌ها منبعی عظیم و دقیق از رویدادهای کسب‌وکار هستند، پس می‌توانند توسط سامانه‌های دیگری همچون کتاب‌خانه‌های یادگیری ماشین، انبار داده، استخراج، بارگیری، جابه‌جایی[۷۱] یا هوش تجاری[۷۲] مورد بهره‌برداری مجدد قرار گیرند. عملیات نوشتن در این رویکرد بسیار کم‌هزینه است، حال آن‌که عملیات خواندن هزینه‌ی محسوسی می‌طلبد.

از سوی دیگر، داده‌های سرجمع که در حافظه‌ی نهان نگهداری می‌کنیم برای خواندن‌های مکرر و سریع بسیار مناسب هستند. داده‌هایی که در این بخش از راه‌حل نگهداری می‌شوند، هنگام دریافت بلادرنگ پردازش می‌شوند و حاوی آخرین تغییرات هستند. عملیات خواندن در این مدل هزینه‌ی پردازشی ناچیزی در بر دارد. بنابراین برای ارائه‌ی خدمت در وبگاه‌هایی که انبوه کاربران برخط داریم، بسیار مناسب است. در یک راهکار بهینه، با هوشمندسازی منطق مصرف رویدادها در دو روش یادشده، می‌توان به کارایی و دقتی مناسب دست یافت. به این ترتیب که با محاسبه‌ی معیارهای[۷۳] زمان اجرای راه‌حل و همچنین در نظر گرفتن الزامات کسب‌وکار[۷۴] و یافتن مرزهای بحرانی و شکنندگی راه‌حل، می‌توان به دانش کافی برای تغذیه‌ی هر یک از پیامگیرها دست یافت. این دانش، حاوی اطلاعاتی مفید در مورد الگوها[۷۵]، بسامد[۷۶]، اندازه[۷۷] و میزان اهمیت[۷۸] هر پیام خواهد بود.

۶۶ - Machine Learning
۶۷ - Hybrid
۶۸ - Consumers
۶۹ - DB Writes
۷۰ - DB Reads
۷۱ - Extract Transfer Load
۷۲ - Business Intelligence

تعریف

پیش از معرفی چهارچوب‌ها و ابزارهای پردازش جریان داده، به ارائه‌ی تعریفی از بن‌سازه[۷۹]ی پردازش جریان داده می‌پردازیم:

یک بن‌سازه‌ی پردازش جریان داده به دو نیاز زیر پاسخ می‌دهد:

۱. یکپارچه‌سازی داده[۸۰]:

بن‌سازه‌ی پردازش جریان داده، با تسخیر پیوسته‌ی جریان‌های رویداد و داده، سامانه‌های دیگر را تغذیه می‌کند. این سامانه‌ها طیف متنوعی همچون پایگاه داده‌ی رابطه‌ای و غیررابطه‌ای، مخازن کلید-مقدار[۸۱]، هدوپ یا سامانه‌های انبار داده هستند.

۲. پردازش جریان[۸۲]:

پردازش پیوسته، بلادرنگ و اعمال تغییرات بر جریان‌های داده و تولید نتیجه برای دیگر سامانه‌ها توسط پردازش جریان صورت می‌گیرد.

برای آشنایی بیشتر و عمیق‌تر با چهارچوب‌ها و ابزارهای پردازش جریان داده و همچنین الگوریتم‌ها و ترفندهای مورد استفاده در این زمینه می‌توانید قسمت دوم این مقاله را در وبسایت اف‌یک مطالعه کنید.

۷۳ - Metrics
۷۴ - Business Constraints
۷۵ - Patterns
۷۶ - Frequency
۷۷ - Length
۷۸ - Priority
۷۹ - Platform
۸۰ - Data Integration
۸۱ - Key-Value Stores
۸۲ - Stream Processing

بررسی شباهت افراد بر اساس تاریخچه‌ی مکانی

امید ابراهیمی
متولد آذر ۱۳۶۹، بیرجند
دانش‌جوی کارشناسی ارشد
نرم‌افزار دانشگاه تهران
زمینه‌های پژوهشی:
Big Data, Data Mining,
Trajectory Computing
وبسایت: omid.ebrahimi.pro

امید ابراهیمی
omid@ebrahimi.pro

اینترنت اشیا، یکی از مهم‌ترین منابع تولید داده‌های بزرگ، در چند سال اخیر رشد به‌سزایی در زندگی روزمره‌ی افراد داشته است. در بحث اینترنت اشیا، شبکه‌های حسگر نقش اساسی را ایفا می‌کنند. در این میان حسگرهای مکان‌یاب یکی از پراستفاده‌ترین انواع حسگرها توسط عموم مردم هستند؛ به‌طوری که با افزایش سریع تبلت‌ها و گوشی‌های هوشمند در سرتاسر دنیا، سیستم موقعیت‌یاب جهانی[۱] نزدیک‌ترین ابزار تولید داده‌های حسگری در فضای اینترنت اشیا به‌نظر می‌رسد. در نگاه اول، سیستم موقعیت‌یاب جهانی تنها ابزاری برای مسیریابی به‌نظر می‌آید؛ اما ذخیره‌ی داده‌های مکانی مربوط به هر فرد پتانسیلی باورنکردنی برای استفاده در شبکه‌های اجتماعی، سایت‌های خریدوفروش اینترنتی، بازاریابی تلفنی[۲] و اینترنتی، یادگیری الکترونیکی[۳] به‌صورت تطبیقی و بسیاری از زمینه‌های مورد توجه در فناوری اطلاعات به‌همراه دارد. این پتانسیل با افزایش داده‌های مکانی و غنی‌ترشدن اطلاعات موجود در نقشه‌های الکترونیکی همچنان بیشتر و بیشتر می‌شود.

پس از ورود حسگرهای موقعیت‌یاب به زندگی روزانه‌ی افراد، تحقیقات جدیدی بر مبنای آن‌ها آغاز شد. برای اولین بار، یو ژنگ[۴] و همکارانش در سال ۲۰۰۸ یک معماری کلی ارئه دادند که نحوه‌ی دنبال‌کردن زندگی افراد را از طریق داده‌های مکانی بیان می‌کرد. این شروعی برای کارهای تحقیقاتی جالب و متعدد در این زمینه بود که ناگزیر از بیشتر آن‌ها می‌گذریم و روی پژوهشی که بر پایه‌ی این معماری اولیه در آزمایشگاه دکتر فاطمی در حال انجام است متمرکز می‌شویم.

در این مقاله بررسی میزان تشابه افراد با استفاده از تاریخچه‌ی حرکتی آن‌ها مورد نظر ماست. به‌عبارتی، می‌خواهیم میزان نزدیکی نحوه‌ی زندگی هر دو فرد را با توجه به حرکاتی که در طول زندگی انجام داده‌اند به‌دست آوریم. به این ترتیب برای هر فرد در جامعه می‌توانیم یک رتبه‌بندی ارائه دهیم که در آن نفر اول شبیه‌ترین فرد را نشان می دهد و در رتبه‌های بعدی میزان شباهت کم‌تر می‌شود؛ همچنین به شباهت هر دو فرد یک عدد نسبت داده می‌شود که میزان شباهت را نشان می‌دهد. برای این منظور تاریخچه‌ی حرکتی افراد نه تنها از نظر فیزیکی، بلکه از نظر معنایی نیز مورد بررسی قرار می‌گیرد و الگوی حرکتی هر فرد با سایر افراد مقایسه می‌شود.

۱ - GPS
۲ - Telemarketing
۳ - E-Learning
۴ - Yu Zheng

در ادامه به‌طور خلاصه نحوه‌ی پیداکردن شباهت فیزیکی افراد را توضیح می‌دهیم؛ زیرا این موضوع پیش‌زمینه‌ی درک روش معنایی است. پس از آن توضیح مختصری از این روش خواهیم داشت. در نهایت در مورد پیچیدگی‌های محاسباتی این الگوریتم‌ها و لزوم استفاده از تکنولوژی‌های داده‌های بزرگ توضیح خواهیم داد.

ابتدا لازم است چند مفهوم را تعریف کنیم:

POI Database: Point of Interest

پایگاه‌داده‌ای که POIها را نگهداری می‌کند. هر POI معمولاً شامل طول و عرض جغرافیایی، نام و نوع مکان مربوطه است.

Trajectory:

تعداد زیادی نقطه که هر کدام از آن‌ها با یک طول و عرض جغرافیایی و یک فیلد زمان مشخص شده‌اند و توالی آن‌ها یک مسیر حرکت را مشخص می‌کند.

به‌دست‌آوردن تشابه از دید فیزیکی

در الگوریتم کشف تشابه فیزیکی، ابتدا برای هر فرد نقاط توقف[۵] را به‌دست می‌آوریم. به‌طور خلاصه اگر یک فرد در یک محوطه با قطر R به‌مدت بیشتر از T باقی بماند مرکز این محوطه را به‌عنوان یک نقطه‌ی توقف در نظر می‌گیریم. در آزمایش‌های انجام شده بهترین مقدار برای R = ۲۰۰m و برای T, min = در نظر گرفته شده است. نمونه‌ای از نقاط توقف را در شکل شماره‌ی ۱ می‌بینید.

پس از به‌دست‌آوردن نقاط توقف افراد، همه‌ی آن‌ها را در یک مجموعه‌داده[۶] قرار می‌دهیم و به‌صورت سلسله‌مراتبی، با روش بالا به پایین[۷] خوشه‌بندی می‌کنیم. در این‌جا به‌طور خاص برای خوشه‌بندی از روش OPTICS استفاده شده است. با این کار، در هر لایه نقاط توقف نزدیک به هم برای افراد مختلف در خوشه‌های یکسانی قرار می‌گیرند. در واقع هر خوشه در هر لایه نشان‌دهنده‌ی مکانی است که افراد زیادی در آن توقف داشته‌اند؛ پس می‌توان نتیجه گرفت که این مکان مهم است و اشتراک افراد در آن می‌تواند نشانه‌ای از شباهت بیشتر آن‌ها باشد.

بعد از مرحله‌ی خوشه‌بندی، برای هر فرد در هر لایه خوشه‌هایی را که رفته است به‌ترتیب زمان به‌صورت یال‌های جهت‌دار به هم متصل می‌کنیم. به این ترتیب در هر لایه برای هر فرد یک گراف به‌دست می‌آید که گره‌های آن خوشه‌های به‌دست‌آمده در مرحله‌ی قبل هستند و یال‌هایش مسیر حرکت فرد را نشان می‌دهند. شکل شماره‌ی ۲ نمای کلی مرحله‌ی خوشه‌بندی را نشان می‌دهد. در لایه‌های بالاتر خوشه‌ها بزرگ‌ترند و تعدادشان کمتر است؛ هرچه به‌سمت پایین می‌رویم تعداد خوشه‌ها بیشتر شده و در عوض کوچک‌تر می‌شوند.

پس از این‌که گراف در سطوح مختلفِ خوشه‌بندی ساخته شد، برای هر فرد یک دنباله از گره‌ها خواهیم داشت و با استفاده از زمان ورودِ این فرد به نقاط توقفش و خروج از آن‌ها می‌توانیم بازه‌های زمانی بین هر دو گره را برایش پیدا کنیم. سپس در هر سطح از گراف برای هر فرد، نقاط توقف او را در هر گره به‌ترتیب زمان مرتب می‌کنیم. شکل شماره‌ی ۳ نمونه‌ای از دنباله‌ی کامل پیموده‌شده برای هر فرد را با در نظر گرفتن بُعد زمان نشان می‌دهد. در این شکل دایره‌های آبی که در هر گره با خط‌چین به هم متصل شده‌اند نقاط توقف فرد را در آن گره به‌ترتیب زمان نشان می‌دهند.

بعد از این مرحله برای هر فرد یک دنباله خواهیم داشت و بر اساس دو معیار زیر می‌توانیم شباهت را محاسبه کنیم:

۱. دنباله‌های مشترک در سطوح پایین‌تر، امتیازِ بالاتری نسبت به سطوح بالاتر دارند. از آن‌جا که مثلاً افرادی که در یک مرکز خرید مشترک توقف زیادی داشته‌اند نسبت به افرادی که در یک کشور مشترک توقف زیادی داشته‌اند به هم نزدیک‌ترند.
۲. در این‌جا کار ما به نوعی الگوریتم تطبیق دنباله[۸] تبدیل می‌شود که از پرداختن به جزئیات آن می‌گذریم.

به‌دست‌آوردن تشابه از دید معنایی

ما برای مدل‌کردن تاریخچه‌ی حرکتی افراد به‌صورت معنایی از POIها استفاده می‌کنیم. یعنی برای هر فرد دنباله‌ای از POIهایی که پیمایش کرده است را به‌دست می‌آوریم. به این ترتیب، برای هر فرد دنباله‌ای مثل «...→رستوران→سینما→دانشگاه» خواهیم داشت که با تطبیق این دنباله‌ها می‌توان شباهت را از نظر معنایی بدست آورد. به‌عبارتی در این روش دیگر لازم نیست افرادِ مورد بررسی در یک منطقه‌ی یکسان زندگی کنند؛ بلکه ممکن است دو فرد در دو کشور دور از هم، به‌دلیل تاریخچه‌ی حرکتی مشابه از نظر معنایی، مشابه تشخیص داده شود. شکل شماره‌ی ۴ نمونه‌ای از این تطابق معنایی را بین چند فرد نشان می‌دهد.

برای این‌که بفهمیم هر کدام از نقاط توقف مربوط به چه نوع مکانی است، ساده‌ترین راه این است که در پایگاه‌داده‌ی POI جستجو کنیم و نوع نزدیک‌ترین POI به هر نقطه‌ی توقف را به‌عنوان نوع آن نقطه‌ی توقف در نظر بگیریم. مثلا اگر نزدیک‌ترین POI به نقطه‌ی توقف مربوط به یک دانشگاه است می‌توانیم فرض کنیم که فرد در یک دانشگاه توقف داشته است. اما واضح است که این استدلال درستی نیست. هر مکان در پایگاه‌داده‌ی POI با یک طول و عرض جغرافیایی مشخص شده و محدوده‌ی آن مشخص نیست؛ بنابراین اگر شما در کلاسی در حاشیه‌ی دانشگاه نشسته باشید، ممکن است POI ثبت‌شده برای بیمارستان کنار دانشگاه از POI ثبت‌شده برای دانشگاه به شما نزدیک‌تر باشد و در نتیجه مکان شما به‌اشتباه بیمارستان تشخیص داده خواهد شد! برای حل این مشکل، برای هر نقطه‌ی توقف، یک بردار[۹] از مکان‌های اطراف آن نقطه تا یک شعاع مشخص تشکیل داده می‌شود و در واقع الگوریتم تطبیق دنباله روی دنباله‌هایی از بردارها اعمال خواهد شد. به‌دنبال این تغییر، در قسمت‌های دیگر الگوریتم، از جمله معیار فاصله، در خوشه‌بندی نیز تغییراتی ایجاد می‌شود؛ زیرا در این‌جا به‌جای نقاط، بردارها را خوشه‌بندی می‌کنیم.

۵ - Stay points
۶ - Dataset
۷ - Top-down

۸ - Sequence matching
۹ - Vector

پیچیدگی‌های محاسباتی

مجموعه‌داده‌ی استفاده‌شده در این الگوریتم مربوط به ۱۸۲ نفر از ساکنان شهر پکن[۱۰] است. این مجموعه‌داده در طول پنج سال جمع‌آوری شده و مکان این افراد را با فاصله‌های زمانی حدود پنج ثانیه ثبت کرده است. البته افراد به‌تدریج به این مجموعه اضافه شده‌اند و برای همه‌ی آن‌ها پنج سال کامل ثبت نشده است. این مجموعه شامل ۲۴,۸۷۶,۹۷۸ نقطه بوده و ۱,۲۹۲,۹۵۱ کیلومتر حرکت در آن ثبت شده است. شکل شماره‌ی ۵ نمایی گرافیکی از نقاط ثبت‌شده در پکن را نشان می‌دهد.

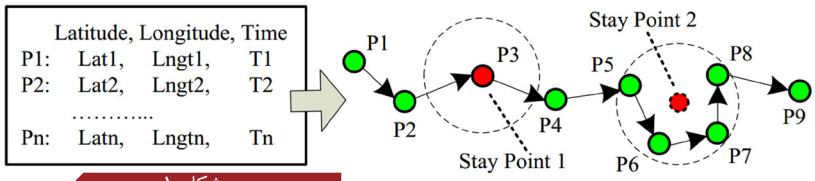
توجه داشته باشید که این مجموعه‌داده فقط مربوط به ۱۸۲ نفر است و تاریخچه‌ی همین افراد هم کامل نیست زیرا به‌تدریج اضافه شده‌اند. در حالی که به‌صورت کاربردی باید حداقل تعداد افراد را برابر یک جامعه همچون شهر، کشور یا در حالت ایده‌آل تمام دنیا در نظر گرفت و برای نتایج دقیق‌تر شاید لازم باشد مدت زمان بیشتری، مثلاً ده سال یا حتی کل تاریخچه‌ی حرکتی فرد را در طول زندگی وی بررسی کنیم. احتمالاً در مورد خوشه‌بندی و پیچیدگی‌های محاسباتی آن اطلاعاتی دارید و می‌دانید به‌طور کلی خوشه‌بندی جزو اعمال بسیار پرهزینه در داده‌کاوی به‌شمار می‌رود. حال اگر این تعداد بالای نقطه را در نظر بگیرید، می‌توانید حجم بالای محاسبات و حافظه و دیسک مورد نیاز برای این محاسبات را تصور کنید.

همان‌طور که گفته شد، در کشف تشابه افراد از دیدگاه معنایی از یک پایگاه‌دادهی POI نیز استفاده شده است. نسخه‌ی جدید این پایگاه داده به‌صورت هفتگی در وبسایت OpenStreetMap[۱۱] قرار می‌گیرد و پس از خارج‌کردن آن از حالت فشرده، یک فایل XML با حجم حدود ۶۰۰ گیگابایت خواهیم داشت که برای هر نقطه‌ی توقف باید در این پایگاه‌داده جستجو کنیم و POIهایی را که در یک شعاع مشخص از این نقطه قرار دارند به‌دست آوریم.

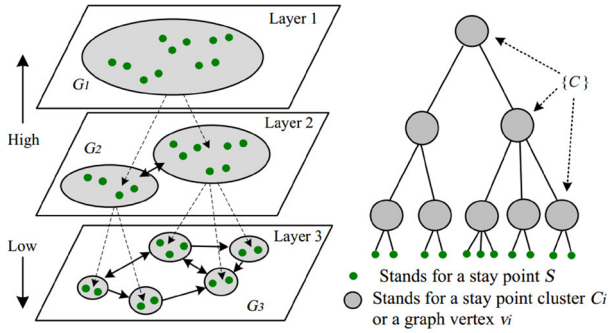
با توجه به این توضیحات، واضح است که چنین الگوریتمی با همین مجموعه‌داده‌ی ۱۸۲نفره در یک سیستم عادی قابل اجرا نیست؛ حال اگر تعداد افراد را در یک مقیاس کاربردی‌تر در نظر بگیریم چه خواهد شد؟! مقیاس این مسئله به‌حدی بالاست که راهی جز استفاده از تکنولوژی‌های داده‌های بزرگ باقی نمی‌ماند. پژوهشگران در آزمایشگاه دکتر امید فاطمی به‌دنبال ارائه‌ی روشی بر مبنای تکنولوژی‌های داده‌های بزرگ هستند تا بتوانند اجرای این الگوریتم را در ابعاد واقعی به واقعیت نزدیک‌تر کنند.

تقاضای همکاری:

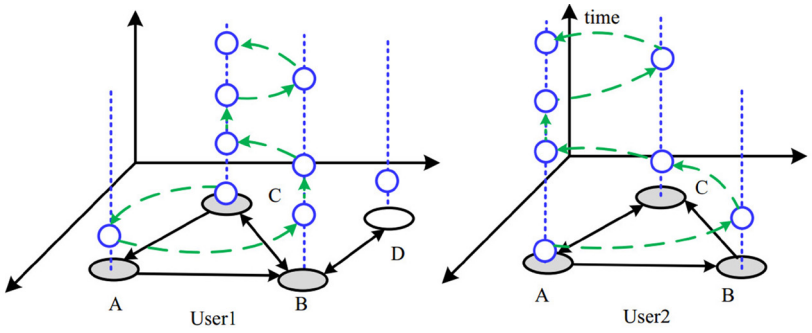
برای ارزیابی الگوریتمی که توضیح داده شد، نیاز به داده‌های مکانی تعدادی از افراد است که دسترسی به آن‌ها مقدور باشد. در صورتی که می‌توانید در جمع‌آوری داده همکاری نمایید، لطفاً به آدرس ای‌میل ebrahimi.omid@ut.ac.ir بزنید. در انتهای این پروژه به هر یک از افرادی که در جمع‌آوری این داده‌های مکانی مشارکت داشته باشند شبیه‌ترین افراد به خودش در قالب یک رتبه‌بندی ارائه خواهد شد.



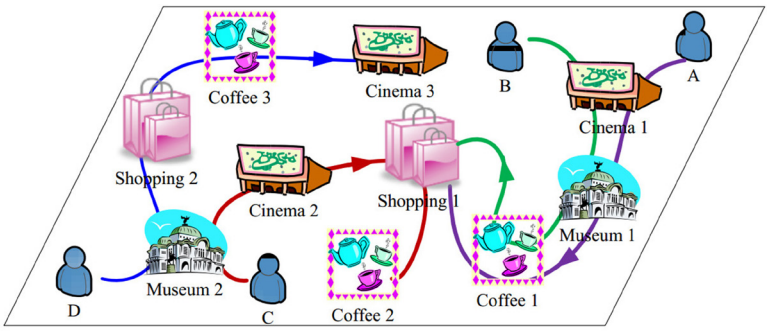
شکل ۱



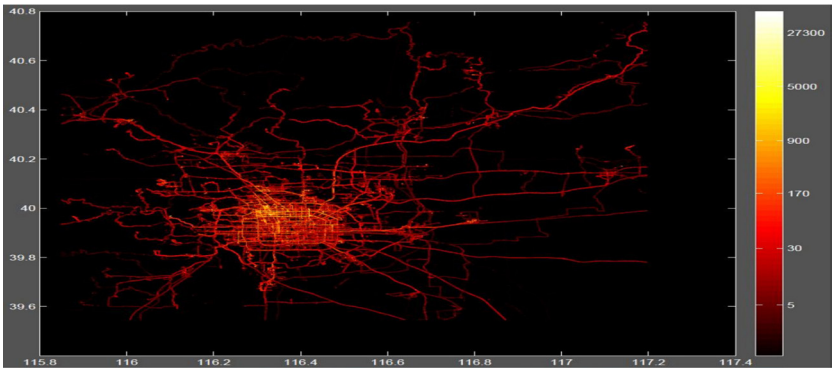
شکل ۲



شکل ۳



شکل ۴



شکل ۵

۱۰ - Beijing
۱۱ - http://planet.openstreetmap.org

داستان یک استخدام

جادی یک گیک خوشحال و دوست‌دار دنیای تکنولوژی و ارتباط آن با جامعه است؛ متخصص لینوکس، برنامه‌نویس و علاقه‌مند به لایه های پایین‌تر تکنولوژی. او خارج از کار اصلی‌اش، به ساختن آموزش‌های ویدیویی، تولید پادکست و نوشتن در وبلاگ شخصی‌اش می‌پردازد.

مترجم: امیر میرمیرانی(جادی)
jadi@jadi.net

تام، دیک و هری برای مصاحبه‌ی استخدامی در سالن انتظار یک استارت‌آپ نشسته‌اند. این استارت‌آپ امکان خرید از میوه‌فروشی و سوپرمارکت را توسط یک اپلیکیشن برای مشتریان ممکن می‌کند. در اتاق انتظار یک وایت‌برد، یک لپ‌تاپ و یک دفتر یادداشت گذاشته شده تا هر کس به هر روشی که دوست دارد کار کند. مصاحبه‌کننده، دانشمند داده‌ی ارشد این استارت‌آپ، جان است.

مصاحبه‌ی تام

جان بحث را شروع می‌کند و می‌گوید خوب است در مورد میوه‌ها و سبزیجات گپ بزنند. تام موافق است. یک شروع مؤدبانه. جان می‌گوید:

– فرض کن می‌خوای بری خرید. یک کیسه‌ی خرید همراه داری و مقداری پول و قراره برای کل هفته خرید کنی. مسئله اینه:

۱) حداکثر ده پوند می‌تونی با خودت میوه حمل کنی.

۲) حداکثر ۱۰۰ دلار پول داری.

۳) روزانه ۲۰۰۰ کالری لازم داری پس خرید هفتگی باید ۱۴۰۰۰ کالری داشته باشه.

۴) از هر چیز باید حداقل چهار اونس بخری. (هر پوند ۱۶ اونس است، پس از هر جنس باید حداقل یک-چهارم پوند خرید.)

قیمت هر پوند

- گوشت: ۴ دلار
- کاهو: ۱.۵ دلار
- پنیر: ۵ دلار
- تن: ۲۰ دلار
- نان: ۱.۲۰ دلار

میزان کالری در هر پوند

- گوشت: ۶۵۰ کالری
- کاهو: ۷۰ کالری
- پنیر: ۱۶۷۰ کالری
- تن: ۸۳۰ کالری
- نان: ۱۳۰۰ کالری

و این هم اطلاعات مواد است:

وقتت را استفاده کن و ببین به چند روش می‌شود خرید کرد.

جان از اتاق خارج می‌شود و تام کمی فکر می‌کند، بعد به سراغ لپ‌تاپ می‌رود و ویرایشگر مورد علاقه‌اش را باز می‌کند و کمی به اسکالا کد می‌نویسد. خروجی:

```
$ scalac Groceries.scala
$ scala Groceries
0.Ham 0.25 Lettuce 0.25 Cheese 4.50 Tuna 0.25 Bread 4.75
Weight 10.00 Cost 34.58 Calories 14077.50
1.Ham 0.25 Lettuce 0.25 Cheese 4.75 Tuna 0.50 Bread 4.25 Weight
 10.00 Cost 40.23 Calories 14052.50
2.Ham 0.25 Lettuce 0.25 Cheese 5.00 Tuna 0.75 Bread 3.75 Weight
 10.00 Cost 45.88 Calories 14027.50
3.Ham 0.25 Lettuce 0.25 Cheese 5.25 Tuna 0.25 Bread 3.75 Weight
 9.75 Cost 37.13 Calories 14030.00
4.Ham 0.25 Lettuce 0.25 Cheese 5.25 Tuna 1.00 Bread 3.25 Weight
 10.00 Cost 51.53 Calories 14002.50
....
163.Ham 1.75 Lettuce 0.25 Cheese 7.25 Tuna 0.50 Bread 0.25 Weight
 10.00 Cost 53.93 Calories 14002.50
```

```
object Groceries extends App {
  val maxCost = 100.0
  val (fourOunce, tenPound) = (4/16.0, 10.0) // 16 ounces per pound
  type D = Double

  // price given weights per pound
  def price(ham:D, lettuce:D, cheese:D, tuna:D, bread:D) =
    ham * 4 + lettuce * 1.5 + cheese * 5 + tuna * 20 + bread * 1.2

  // calories given weights per pound
  def cals(ham:D, lettuce:D, cheese:D, tuna:D, bread:D) =
    ham*650+lettuce*70+cheese*1670+tuna*830+bread* 1300

  def solver = {
    val range = (fourOunce to tenPound by fourOunce)
    for {

      ham     <- range lettuce <- range cheese  <- range
      tuna    <- range bread   <- range

      weight =  (ham+lettuce+cheese+tuna+bread)
      cost = price(ham,lettuce,cheese,tuna,bread)
      calories = cals(ham,lettuce,cheese,tuna,bread)
      diff = calories - 14000
      if ( weight <= 10.0) // weight below 10 lbs
      if ( cost <= 100.0) // cost below $100
      if ((diff >= 0.0) && (diff < 100.0)) // calories approx 14000

    }yield{(ham,lettuce,cheese,tuna,bread, weight, cost, calories)}
  }

  def mkStr( x:(D,D,D,D,D,D,D,D)) = {
    val (ham,lettuce,cheese,tuna,bread, weight, cost, calories) = x
    "Ham %.2f Lettuce %.2f Cheese %.2f Tuna %.2f Bread %.2f
      Weight %.2f Cost %.2f Calories %.2f"
    .format( ham,lettuce,cheese,tuna,bread,weight,cost,calories)
  }

  solver.map { mkStr }.distinct.zipWithIndex
  .foreach{ case (str, idx) => println(idx + "." + str) }
}
```

کد تام

جان نگاهی به برنامه می‌اندازد و می‌پرسد چند حالت برای جواب پیدا شده؟

– ۱۶۴ حالت.

– خب از این ۱۶۴ حالت کدام از همه ارزان‌تر است؟

– ساده است... کمی کد:

```
println(mkStr(solver.maxBy{ case
 (ham,lettuce,cheese,tuna,bread, weight, cost, calories)
 => cost })) // most expensive
Ham 0.25 Lettuce 0.25 Cheese 7.00 Tuna 2.25 Bread 0.25 Weight
 10.00 Cost 81.68 Calories 14062.50
```

– عالی. پس در این حالت ۸۲ دلار مصرف می‌کنیم، حالا می‌توانی بگویی کم‌چربی‌ترین راه‌حل چیست؟

– نباید سخت باشد. احتمالاً باید گوشت و پنیر و تن را کم کنم.

```
println(mkStr(solver.minBy{ case
 (ham,lettuce,cheese,tuna,bread, weight, cost, calories)
 => ham+cheese+tuna })) // least fatty
Ham 0.25 Lettuce 0.25 Cheese 4.50 Tuna 0.25 Bread 4.75
 Weight 10.00 Cost 34.58 Calories 14077.50
```

پس کافی است ۷.۵ پوند نان و تقریباً همین‌قدر پنیر بخریم که فقط ۳۵ دلار

هزینه خواهد داشت. چه جالب!

جان می‌گوید:

- بسیار عالی تام. این چیزی بود که من در مورد تو دوست داشتم. با روحیه داخل شدی و اصلاً نمی‌دانستی ممکن است چه چیزی بپرسم. من مسئله را مطرح کردم و کمی برنامه نوشتی و جواب‌هایی واقعی پیدا کردی. جواب‌ها درست بودند و جالب و با کدنویسی هم مشکلی نداشتی. تقریباً در ۴۵ خط کد که در ۴۵ دقیقه نوشته شد جواب همه‌ی سؤال‌ها را کشف کردی.

- خوشحالم. من هر روز برنامه می‌نویسم و در نتیجه کدزدن برایم راحت است. این‌جا هم عملاً فقط یک حلقه نوشتم که کار سختی نیست.

- واقعاً هر روز کد می‌نویسی؟

- بله. همیشه در حال برنامه‌نویسی هستم!

- پس چه زمانی فرصت می‌کنی چیز جدید یاد بگیری؟ منظورم ریاضی، یادگیری ماشین، آمار و این چیزها است. یادگرفتن این‌ها نیاز به زمان نسبتاً زیادی دارد. اگر همیشه در حال برنامه‌نویسی هستی چه زمانی برای پیشرفت خودت می‌گذاری؟

تام کمی مِن‌مِن می‌کند و می‌گوید:

- خب من یک برنامه‌نویس خودآموز هستم. با کدنوشتن چیزهای جدید هم یاد می‌گیرم. اگر هم قرار باشد چیزی یاد بگیرم، در موردش یک برنامه می‌نویسم. این روش من است.

- یعنی هیچ‌وقت یک کتاب ریاضی برنمی‌داری بخوانی؟ یا روی کاغذ مسئله حل نمی‌کنی؟ مثلاً با خودکار و...

- مگر در مدرسه هستیم؟ نه. من معمولاً فقط کد می‌نویسم.

- گاهی لازم است تحقیق کنیم یا مثلاً مقاله‌های منتشرشده در حوزه‌ی علوم کامپیوتر را بخوانیم و تقریباً همه‌ی این تحقیق‌ها و مقاله‌ها مبتنی بر ریاضیات هستند... اثبات چیزها!

- این کارها با من هماهنگ نیست جان. من دوست دارم برنامه بنویسم. یک الگوریتم به من بدهید و من سریع و مؤثر برای‌تان پیاده‌اش می‌کنم.

جان حرف مهمی برای گفتن دارد. چند لحظه مکث می‌کند و می‌گوید:

- خب حالا تصور کن به‌جای گوشت و کاهو و نان و پنیر و تن یعنی به‌جای این پنج قلم، واقعاً با یک سوپرمارکت طرف باشیم که هزار جنس مختلف دارد.

- خب؟

- خب؟! در آن حالت چه‌طور می‌خواهی برنامه‌ات را اجرا کنی؟

- متوجه نمی‌شوم. چه مشکلی هست؟ حلقه اجرا می‌شود دیگر.

- خب تو در واقع مشغول بروت‌فورس[1]کردن مسئله هستی. امتحان‌کردن همه‌ی حالت‌های ممکن. این روش برای پنج قلم جنس کار می‌کند، ولی برای هزار قلم نه.

- کار که می‌کند، فقط بیشتر طول می‌کشد.

- و این مشکلی نیست؟

- نه. اگر هم واقعاً طولانی شد می‌شود سراغ نگاشت‌کاهش[2] یا روش‌های دیگر پردازش موازی رفت.

- واقعاً می‌گویی که اگر هزار قلم جنس هم بود همین برنامه را می‌نوشتی و در زمان معقولی جواب می‌گرفتیم؟

- جواب نمی‌گرفتیم؟ پردازنده‌ها ارزان و سریع شده‌اند!

جان راضی نیست. خوشحالی اولیه جایش را به شک داده است. می‌گوید:

- فکر می‌کنم متوجه وسعت جریان نیستی. وقتی گوشت را بررسی می‌کنی و از چهار تا ده پوند را چک می‌کنی، چند تا گزینه داریم؟

- اوم...

```
scala> (10-0.25)/0.25
res16: Double = 39.0
```

- پس با قدم اول، ۴۰ حالت مختلف را بررسی می‌کنیم.

- بله. و؟

- پس با ۴۰ گزینه برای هر کالا، پنج کالا به چند بررسی نیاز دارد؟

- الان...

```
scala> printf("%.2f", math.pow(40,5))
102400000.00
```

- می بینی؟ ۱۰۲میلیون حالت را بررسی کرده‌ای.

- متوجه شدم. پس با هزار قلم جنس... من...

```
scala> printf("%.2f", math.pow(40,1000))
Infinity
```

- هاهاها...

- واو... بله. فکرش را نکرده بودم. حتی رپل[3] هم گفت بی‌نهایت.

- هههوها... بله... پس در این مورد باید چه کار کنیم؟

- نمی‌دانم! از من نپرسید. احتمالاً باید با ریاضیات و این تیپ چیزها حلش کرد. متوجه هستم چه می‌گویید، اما از تخصص من خارج است. من کد می‌نویسم و شغلم تحقیق و توسعه و ریاضیات و این چیزها نیست. واقعاً اهل خواندن مقاله و کتاب درسی نیستم. ممنونم از وقتی که گذاشتید.

- ممنون از تو تام... ممکن است با شما تماس بگیریم.

- به هر حال... فعلاً.

مصاحبه‌ی دیک

در این مصاحبه هم جان در نقش دانشمند داده‌ی ارشد وارد می‌شود و باز هم می‌گوید قرار است در مورد خرید صحبت کنند. در کمال تعجب دیک جواب می‌دهد:

- چرا؟

- اوم... خب این شغل ما در این‌جا است. ما یک استارت‌آپ مواد غذایی هستیم. فرض کن وارد یک فروشگاه می‌شوی و یک کیسه‌ی خرید به همراه داری و مقداری پول، و می‌خواهی برای یک هفته خرید کنی. چند قانون هم داریم:

۱) کیسه‌ی خرید فقط ظرفیت ده پوند بار را دارد.

۲) فقط ۱۰۰ دلار پول داری.

۳) در روز به ۲۰۰۰ کالری نیاز داری، در نتیجه در هفته...

دیک صحبت جان را قطع می‌کند:

- من معمولاً درگیر این الگوریتم‌های عجیب‌وغریب نمی‌شوم.

- متوجه نشدم... خب این تیپ کاری است که ما در این‌جا می‌کنیم و لازم داریم.

- بله ولی لطفاً در مورد اصل کار صحبت کنید. چرا من این‌جا هستم؟

- اصل کار؟!

دیک کمی تهاجمی است. می‌گوید:

- اصل کار. هدوپ. داده‌های بزرگ. خوشه. بحث و سؤال در مورد تعداد نگارنده‌های مناسب و تعداد کاهنده‌ها[4]. گپ در مورد پایپ‌لاین داده‌ای. من تجربه‌ی گسترده‌ای دارم و می‌توانیم به‌خوبی بحث‌های تخصصی کنیم. تریفت در مقابل پروتوباف. ابزارشناسی، دوری‌سنجی[5]، سیستم‌های ساخت[6]. روند توسعه و ساخت شما چگونه است؟ آزمون‌های واحد را چه‌طور می‌نویسید؟ گزارش‌های سیستم چه‌طور کار می‌کند و آیا آزمون الف/ب دارید؟ داشبوردها را با چه چیزی می‌سازید و کد را چه‌طور به بازار می‌فرستید؟ بررسی کد چه روندی دارد؟ می‌دانید؟ ترجیح می‌دهم در مورد دنیای واقعی نرم‌افزار حرف بزنم نه در مورد الگوریتم‌های مسخره‌ی خرید.

- اوم... هوم... خب فکر نمی کنید کمی برنامه‌نویسی مفید باشد؟ می‌توانید اصولاً با شبه‌کد ایده‌تان را نشان دهید. من می‌توانم کمی در نوشتن کمک کنم ولی بد نیست اگر کمی برنامه بنویسیم.

- رفیق! من مطمئن هستم که یک تابع جادویی وجود دارد که می‌توانی اطلاعات را به آن بدهی و بگوید چه‌قدر از چه چیزی لازم است. البته این چیزی نیست که من در این مصاحبه کشفش کنم و احتمالاً درست‌کردن و اشکال‌زدایی و آزمودنش به چند ماه برنامه‌نویسی نیاز خواهد داشت.

- نه نه. مطمئن هستم که در همین مصاحبه می‌توانیم یک نسخه‌ی کوچک درست کنیم.

- من با نسخه‌های کوچک کاری ندارم. من با سیستم‌های توزیع‌شده با

۱ - Brute force
۲ - MapReduce
۳ - Read Eval Print Loop (REPL)
۴ - Mappers and reducers
۵ - Telemetry
۶ - Build systems

بیشتر از صد خوشه کار می‌کنم. من مدرک معتبر کلاودرا[۷] دارم و در سه کنفرانس مستقل داده‌های بزرگ سخنران اول بوده‌ام.

– تشکر دیک. با شما تماس می‌گیریم و نتیجه را می‌گوییم.

دیک در لحظه‌ی اول شوکه می‌شود، بعد با نارضایتی بلند شده و از اتاق خارج می‌شود.

مصاحبه‌ی هری

در آخرین مصاحبه هم جان وارد می‌شود و می‌گوید:

– بیایید در مورد خرید صحبت کنیم.

– حتماً. البته فکر می‌کردم قرار است در مورد داده حرف بزنیم.

– بله. اما ما یک استارت‌آپ مواد غذایی هستیم و حل‌کردن این تیپ مسائل از اولویت‌ها است.

در ادامه، جان شرایط مسئله را دوباره بیان می‌کند و می‌گوید:

– از وقتت استفاده کن و روش‌های مختلف خرید بر اساس این شرایط را بگو.

هری دفتر یادداشت روی میز را به‌سمت خودش می‌کشد و صفحه‌ی اول را باز می‌کند. مدادی از پشت گوشش برمی‌دارد و سریعاً چند معادله می‌نویسد و قبل از این‌که جان از پشت میز بلند شود، برگه را به‌سمت او می‌گیرد:

```
X = (h l c t b ...) = 1xn vector
A = [ (1,650,4), (1,70,1.5),
 (1,1670,5),...] = nx3 matrix
B = (10 14000 100) = 1x3 vector
Solve XA = B
```

هری اضافه می‌کند:

– کافی است A را تغییر دهید تا به جواب‌های مختلف برای وزن‌ها و قیمت‌های متفاوت برسید.

جان لبخندی می‌زند و می‌پرسد:

– در این‌جا n چه‌کاره است؟

– خب حدس می‌زنم این مسئله شکل خاصی از مسئله‌ای است که در آن هزاران کالا قرار است تحلیل شود. در این‌جا n تعداد این اقلام است.

– عالی! و چرا A؟

– مطمئناً وزن و قیمت روزبه‌روز تغییر خواهد کرد. شاید حتی در طول یک روز. احتمالاً فروشنده‌ها در روزهایی که سر خودشان شلوغ است هزینه‌ی بیشتری برای شما در نظر خواهند گرفت و خوب است همه‌ی این تغییرات را در یک ماتریس خلاصه کنیم.

– دید شما در مورد واقعیت کسب‌وکار ما عالی است. البته فکر می‌کنم شاید حل‌کردن XA = B راه‌حل خوبی نباشد. در این حالت فقط به یک جواب می‌رسیم و نه به همه‌ی راه‌حل‌های ممکن.

هری کمی فکر می‌کند و بعد با لبخند می‌گوید:

– درست است. این ماتریس فرض می‌کند که یک راه‌حل خاص داریم که در آن دقیقاً ده پوند خرید با قیمت ۱۰۰ دلار بهترین حالت است. در واقعیت کافی است قیمت را زیر ۱۰۰ دلار نگه داریم و اگر وزن هم کمتر از ده پوند بشود مشکلی نخواهد بود. منطقاً باید Bهای مختلفی را تصور کنیم.

– کاملاً صحیح و این Bها را از قبل نداریم پس این روش اصولاً پاسخ‌گو نیست.

– قبول. پس بهتر است سراغ گاوس–جردن نرویم و از روش دانتزیگ[۸] مسئله را حل کنیم.

– خیلی عالی.

– احتمالاً باید یک حل‌کننده‌ی QP[۹] صنعتی بخریم. شاید یک CPLEX یا GAMS یا شاید هم بشود در خود شرکت یکی با NAG ساخت.

– بله. در نهایت راه‌حل این است. اما ممکن است علی‌الحساب با سیمپلکس

```
object simplex extends App {
  type D = Double

  def mkStr( x:(D,D,D,D,D,D,D)) = {
    val (ham,lettuce,cheese,tuna,bread, weight, cost, calories) = x
    "Ham %.2f Lettuce %.2f Cheese %.2f Tuna %.2f Bread %.2f Weight
    %.2f Cost %.2f Calories %.2f"
    .format( ham,lettuce,cheese,tuna,bread,weight,cost,calories)
  }

  case class Item(price:D, calories:D, weight:D = 1.0)
  val (ham, lettuce, cheese, tuna, bread) = (Item(4, 650),
    Item(1.5, 70), Item(5, 1670), Item(20, 830), Item(1.2, 1300))
  val groceries = Array(ham, lettuce, cheese, tuna, bread)
  val prices = groceries.map{ g => g.price }
  val calories = groceries.map{ g => g.calories }
  val weights = groceries.map{ g => g.weight }
  val (maxCost, minCalories, maxCalories) = (100.0, 14000.0, 14100.0)
  val (zeroPound, fourOunce, tenPound) =
  (0.0, 4/16.0, 10.0) // 16 ounces per pound

  def solver = {
    val range = (fourOunce to tenPound by fourOunce)
    for{
      h <- range; l <- range; c <- range; t <- range; b <- range
      // cartesian product of ham, lettuce, cheese, ...
      all = List(h,l,c,t,b)
      if ((all.sum > zeroPound) && (all.sum <= tenPound))
       // basic sanity checks

      // make ten constraints
      cost  = new LinearObjectiveFunction(prices, maxCost)
      // constraint 1. prices <= $100
      nonneg = new NonNegativeConstraint(true)
      // constraint 2. Only want positive weights
      calMin = new LinearConstraint(calories, Relationship.GEQ, minCalories)
       // constraint 3. calories >= 14000
      calMax = new LinearConstraint(calories, Relationship.LEQ, maxCalories)
       // constraint 4. calories <= 14100
      totalWeight = new LinearConstraint(weights, Relationship.LEQ, tenPound)
       // constraint 5. weights <= 10 lb
      constraints =
        new LinearConstraintSet( { List(calMin, calMax, totalWeight) ++
          { all // constraints 6 thru 10. Want each grocery item
            //to have a minimum weight
            .zipWithIndex
            .map { case (min, idx) =>
              (min, Array.tabulate[Double](5)
                { i=> if (i == idx) 1.0 else 0.0 }) }
            .map { case (min, array) =>
              new LinearConstraint(array, Relationship.GEQ,min) }
          }}.asJava )
       optimum = try {
          Some(new SimplexSolver().optimize(cost, constraints, nonneg).getFirst)
        } catch { case e:Exception => None }
    } yield optimum
  }

  solver
  .flatten
  .map{ optimum =>
    val (h,l,c,t,b) = (optimum(0), optimum(1), optimum(2), optimum(3), optimum(4))
    val totalWeight = optimum.sum
    val totalCost = optimum.zip(prices).map{ case (weight,price) =>
     weight * price }.sum
    val totalCalories = optimum.zip(calories).map{ case (weight,cals) =>
     weight * cals }.sum
    mkStr((h,l,c,t,b,totalWeight, totalCost, totalCalories))
  }
  .distinct
  .zipWithIndex
  .foreach { case (str,idx) => println(idx + "." + str) }
}
```

کد هری

هری با اعتمادبه‌نفس می‌گوید:

– حتماً. از کتاب‌خانه‌ی ریاضی آزاد آپاچی استفاده می‌کنم. حل‌کننده‌ی سیمپلکس خوبی دارد.

جان می‌گوید:

– برو ببینم چه کار می‌کنی. می‌توانی حتی از کدی که از برنامه‌ی نفر قبلی مانده استفاده کنی.

و اتاق را ترک می‌کند.

هری یک ساعتی کار می‌کند و کد می نویسد.

جان برای چند ثانیه به کد نگاه می‌کند و بعد از هری خواهش می‌کند که کد را

۷ – Cloudera

۸ – یک روش برنامه‌ریزی خطی

۹ – Quadratic Programming

برایش توضیح دهد. هری می‌گوید:

- در واقع من یک کارتزین از محصولات ساخته‌ام؛ با یک حلقه در خط ۲۵. مجموعه‌ی بزرگی شده و بعد ده شرط را به آن اعمال کرده‌ام. شرط اول می‌گوید فقط ۱۰۰ دلار داریم. این هدف تابع خط ۳۰ است. شرط دوم در خط ۳۱ به‌سادگی می‌گوید وزن منفی نداریم. شرط سوم و چهارم در خطوط ۳۲ و ۳۳ کالری‌ها را در محدوده‌ی ۱۴۰۰۰ کالری در هفته محدود کرده و محدودیت پنجم هم در خط ۳۴ تعریف شده که می‌گوید مجموع کل خریدها نباید از ۱۰ پوند بیشتر شود.

- و پنج محدودیت دیگر چه؟

- فرض کنید که می‌خواهید بگویید گوشت باید حداقل x پوند باشد. در این حالت باید یک بردار به‌شکل <1,0,0,0,...> داشته باشید که گوشت در آن ۱ و بقیه‌ی چیزها صفر است. ضرب این بردار در بردار وزن یعنی <x,y,z,w,...> برابر x می‌شود و در نتیجه می‌توانید محدودیتی روی حداقل مقدار گوشت بگذارید. برای همه‌ی مواد دیگر هم همین مسئله را رعایت کرده‌ام. خط ۳۶ تا ۳۹.

- و این ده محدودیت را چه کار می‌کنید؟

- خب کافی است من این ده محدودیت را به حل‌کننده‌ی سیمپلکس بدهم. خط ۴۳ را نگاه کنید و دعا کنید به جواب برسیم. خوشبختانه جواب را پیدا کرده. مطمئناً ممکن است هیچ جواب قابل‌قبولی پیدا نشود که در آن صورت خطا خواهد داد که در خط ۴۴ جمع‌وجورش می‌کنم.

- متوجهم. و چند جواب پیدا شده؟

```
$ scalac -cp commons-math3-3.3.jar simplex.scala
$ scala -cp commons-math3-3.3.jar:. simplex
0.Ham 0.25 Lettuce 0.25 Cheese 4.29 Tuna 0.25 Bread 4.96
    Weight 10.00 Cost 33.78 Calories 14000.00
1.Ham 0.25 Lettuce 0.25 Cheese 4.61 Tuna 0.50 Bread 4.39
    Weight 10.00 Cost 39.69 Calories 14000.00
2.Ham 0.25 Lettuce 0.25 Cheese 4.93 Tuna 0.75 Bread 3.82
    Weight 10.00 Cost 45.59 Calories 14000.00
3.Ham 0.25 Lettuce 0.25 Cheese 5.24 Tuna 1.00 Bread 3.26
    Weight 10.00 Cost 51.50 Calories 14000.00
...
597.Ham 1.75 Lettuce 0.25 Cheese 7.25 Tuna 0.25 Bread 0.41
    Weight 9.91 Cost 49.11 Calories 14000.00
598.Ham 1.75 Lettuce 0.25 Cheese 7.25 Tuna 0.50 Bread 0.25
    Weight 10.00 Cost 53.93 Calories 14002.50
```

جان می‌گوید:

- واو! ۵۹۹ جواب! تجربه‌ی زیادی در برنامه‌نویسی دارید؟

- نه. واقعاً نه. من خیلی برنامه‌نویس نیستم.

- واقعاً؟!

- من بیشتر می‌خوانم. در مورد یادگیری ماشین زیاد خوانده‌ام.

- هیچ‌وقت برنامه نمی‌نویسید؟

- خب... گاهی کد می‌نویسم. برای اثبات نتایج. اما علاقه‌مند به برنامه‌نویسی و تحویل نرم‌افزار و استفاده از گیت و این چیزها نیستم.

- پس گیت بلد نیستید؟

- اسمش را بلدم!

- هاهاها!

هری توضیح می‌دهد:

- من معمولاً با کامپیوترها درگیر نیستم. من بیشتر اهل کاغذ و مدادم. بعضی مواقع لازم است آدم کد بنویسد که می‌نویسم. اما بخش جذاب کار من کدنوشتن نیست.

- می‌دانید که؟ شغلی که دارید برای آن مصاحبه می‌کنید، شغل برنامه‌نویسی است؛ هرچند که باید در دانش اطلاعات هم خوب باشید.

- بله ولی همه‌ی دانشمندان داده که برنامه‌نویس نیستند. من کدنوشتن را دوست ندارم.

جان دستی به صورتش می‌کشد و می‌گوید:

- هوم...

نتایج

جان و مدیر شرکت در حال بررسی کاندیداها هستند. جان می‌گوید: «اگر وضع‌مان خوب بود باید هر سه را استخدام می‌کردیم!» مدیر با تعجب می‌گوید: «هر سه؟ حتی دیک؟»

جان با خونسردی جواب می‌دهد:

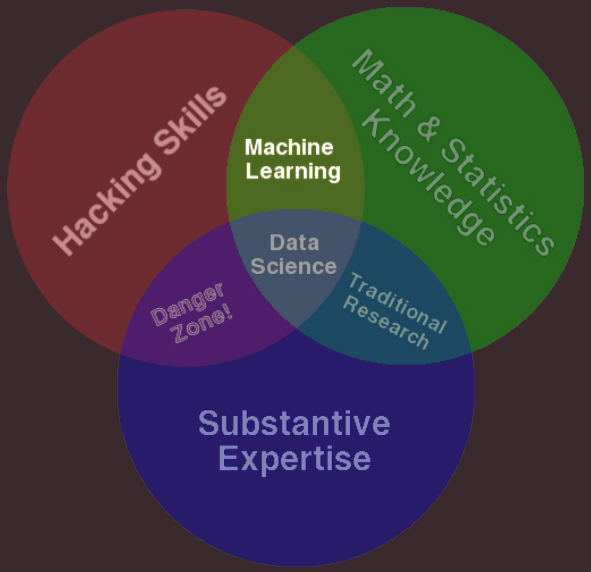
- خب دیک عملاً آدم نحسی بود، اما بله! اگر در مراحل جلوتری بودیم و شرکت بزرگ شده بود، به آدم‌هایی مثل دیک نیاز داشتیم که بتوانند پایپ‌لاین‌های داده‌ای قرص‌ومحکمی بسازند و خودشان را نگران الگوریتم‌های QP نکنند. مشخصاً پیداکردن آدم خوش‌اخلاق و فهیم در تیم بسیار مهم است، اما اگر هیچ‌کس دیگری را پیدا نمی‌کردیم، دیک گزینه‌ی قابل‌قبولی بود. اما حالا نیاز به کسی مثل هری داریم که بتواند مدل‌سازی کند و فرمول طلایی شرکت را دربیاورد. معلوم است که کسی مثل تام را هم لازم داشتیم که کد نهایی را تحویل دهد.

در نهایت شرکت پیشنهاد خوبی از حقوق و سهام به هری داد تا به تیم ملحق شود. همچنین از تام خواست در صورت علاقه‌مندی به یادگیری بیشتر در حوزه‌ی یادگیری ماشین، با حقوق ثابت سر کار بیاید و کم‌کم مسیر تبدیل‌شدن به یک دانشمند داده را طی کند و در طول راه نیز نوشتن کدهای مربوط به پیاده‌سازی الگوریتم‌های تهیه‌شده توسط هری را بر عهده بگیرد.

ستون
دانشمند داده

دانشمند داده کسی است که با استفاده از مهارت و تخصص خود در رشته‌هایی همچون ریاضیات، آمار وعلوم کامپیوتر سعی می‌کند مسائل پیچیده‌ی مطرح در حوزه‌ی داده‌ها را تحلیل کند. در کنار مهارت‌های عملی، داشتن مهارت گفت‌وگو و برقراری ارتباط نیز برای انتقال نتایج یافت‌شده بسیار مهم است.

یک دانشمند داده سعی دارد که فقط به یک منبع داده بسنده نکند و با ترکیب و تحلیل چندین منبع و با استفاده از دانش و مهارت‌های خود برداشتی جدید و کاربردی از داده‌ها عرضه کند. یکی از حوزه‌هایی که تاثیرگذاری علوم داده در آن به وضوح مشاهده می‌شود کسب‌وکار است؛ دانشمندان داده با استخراج بینش از داده‌های بزرگ سازمان‌ها و مدیران را به‌سوی تصمیمات آگاهانه‌تر هدایت می‌کنند.



BIG DATA

طراح: سیدعلی سادات اخوانی

داده‌های بزرگ چه هستند؟

«داده‌های بزرگ» اشاره به مجموعه داده‌های بسیار بزرگ و پیچیده‌ای دارد که باید به روش‌های نوآورانه و استراتژیک پردازش شوند.

۳ نکته‌ی مهم در مورد آن‌ها

حجم

۹۰٪

از داده‌های جهان در ۲ سال اخیر ایجاد شده‌اند

سرعت

۲٫۲میلیارد گیگابایت

داده در هر روز تولید می‌شود

تنوع

داده‌های بی‌ساختار

عکس‌ها، ویدیوها و به‌طور کلی «داده‌های انسانی» همچون ای‌میل‌ها و اطلاعات شبکه‌های اجتماعی

و

داده‌های ساختارمند

داده‌های مرسوم که می‌توانند با ساختار مشخصی در پایگاه داده قرار بگیرند. همانند مشخصات دانش‌جویان

اطلاعات به این سو می‌رود که برای نسل ما تبدیل به یکی از منابع طبیعی شود، دقیقاً مانند جایگاه بخار در قرن ۱۹ام!

جینی رومتی
مدیرعامل IBM

نمونه‌ای جالب

۵۰ پتابایت

برای آنالیزکردن داده‌های بزرگ مشتریان، شرکت آتونومی (یکی از زیرشرکت‌های HP) در سال ۲۰۱۲، بزرگ‌ترین فضای ابری جهان را با حجم ۵۰ پتابایت ایجاد کرد

۲۵تریلیون صفحه

۵۰ پتابایت می‌تواند اطلاعات ۲۵ تریلیون صفحه چاپ شده را در خود جای‌دهد

۶۶۵ سال ویدیوی HD

۵۰ پتابایت می‌تواند ۶۶۵ سال ویدیو با کیفیت HD را شامل شود

۱۷۹ کتاب‌خانه‌ی ملی

۵۰ پتابایت می‌تواند ۱۷۹ بار تمامی آرشیو الکترونیکی کتابخانه‌ی ملی کنگره‌ی آمریکا را شامل شود

مقالات

هنر و صنعت در بازی‌های ویدیویی :: برنامه‌نویسی وب را از کجا آغاز کنیم؟ :: وقتی از استارت‌آپ حرف می‌زنیم از چه حرف می‌زنیم؟ :: مروری بر دستاوردهای آلن تورینگ :: فلسفه‌ی آزادی‌خواهی در نرم‌افزار :: این برنامه احساسات شما را می‌فهمد :: ترند فصل

در این شماره سعی داشته‌ایم به موضوعاتی بپردازیم که در حال حاضر مورد توجه بسیاری از دانش‌جویان قرار دارد. از بررسی فراز و نشیب‌های راه‌اندازی یک استارت‌آپ گرفته تا معرفی دنیای نرم‌افزارهای متن‌باز و حتی روند پیش‌بردن ساخت یک بازی. مقالات مربوط به تحقیقات دانش‌جویی نیز که مانند همیشه در مجله قرار دارند.

دانش‌جویان می‌توانند خود دست‌به‌قلم شده و مطالبی که از نظر آن‌ها مفید است را با دیگران به‌اشتراک بگذارند. مطالب این بخش می‌توانند ترجمه یا اقتباسی از یک مقاله‌ی دیگر باشند یا حاصل دستاوردها و تحقیقات فردی خود نویسنده. معرفی ایده‌های نو، مقالات تخصصی و حتی یک تحقیق دانشجویی از موضوعاتی است که در این بخش به آن‌ها پرداخته می‌شود.

هنر و صنعت در بازی‌های ویدیویی

پیمان وحیدخواه
peyman.vahidkhah@gmail.com

بخش اول:
بازی چیست؟

کریس کراوفورد[1] می‌گوید: «بازی گونه‌ای از سرگرمی است که در آن بازیکنان سعی می‌کنند به هدف خود برسند و دیگران را از رسیدن به هدف‌شان بازدارند (شرط برد و باخت). این تلاش و هدف در چیزهای متنوعی مانند اسباب‌بازی‌ها، داستان‌ها، رقابت‌ها و... نمود پیدا می‌کند.»

سید مایر[2] می‌گوید: «بازی مجموعه‌ای از تصمیمات بامعنا و متوالی است.»

ارنست آدامز[3] و اندرو رولینگز[4] می‌گویند: «بازی مجموعه‌ای از یک یا چند مجموعه چالش به‌هم‌پیوسته است که در یک محیط مجازی یا حقیقی نمود پیدا می‌کند.»

کتی سالن[5] این مفاهیم را سامان می‌بخشد و می‌گوید: «بازی یک سیستم است. سیستمی که در آن یک بازیکن درگیر برخوردها و چالش‌های مجازی می‌شود. این برخوردها و چالش‌ها به‌کمک قوانینی که از قبل برای بازی تعریف می‌شود، به‌وجود می‌آید. خروجی این سیستم سرگرمی‌ست.»

با توجه به مطالب ذکرشده و دیگر تعاریفی که از مفهوم بازی در سال‌های اخیر شده است، می‌توان این تعریف را که برگرفته از کتاب Level Up نوشته‌ی اسکات راجرز[6] است به‌عنوان تعریفی جامع از بازی در نظر گرفت: «بازی فعالیتی است که حداقل به یک بازیکن نیاز دارد، قوانینی بر آن حاکم است که بازیکن از آن آگاه است و شرط یا شروط برد و باخت دارد.»

با داشتن اطلاعات کافی در این سه زمینه (تعداد بازیکنان، قوانین بازی و شروط برد و باخت) می‌توان بازی را فهمید و آن را بازی کرد. پس می‌توان گفت که برای شناخت یک بازی کافی‌ست این سه جنبه را در مورد آن بررسی کنیم. اکثر بازی‌های اطراف ما با این تعریف قابل بررسی هستند. در جدولی که در ادامه می‌آید مثال‌هایی را در این رابطه می‌بینید.

با ظهور بازی‌های کامپیوتری، این رویکرد به این دنیای جدید نیز وارد شد و ساختار بازی‌ها را تحت تاثیر خود قرار داد. نسل اول بازی‌های کامپیوتری به‌شدت به این تعریف از بازی وفادار است. در این گزارش، این دسته از بازی‌های کامپیوتری را - که بازی‌های کامپیوتری سنتی[7] می‌نامیم - مورد بررسی قرار می‌دهیم.

۱ - Chris Crawford: طراح برخی از بازی‌های مشهور دهه‌ی ۸۰ همچون Eastern Front
۲ - Sid Meier: طراح مجموعه بازی‌های Civilization
۳ - Ernest Adams: بنیان‌گذار انجمن بین‌المللی توسعه‌دهندگان بازی‌های رایانه‌ای
۴ - Andrew Rollings: نویسنده‌ی کتاب طراحی و معماری بازی
۵ - Katie Salen: استاد رسانه‌های دیجیتال و نویسنده‌ی کتاب‌هایی در زمینه‌ی طراحی بازی
۶ - Scot Rogers: طراح بازی‌های مشهوری همچون خدای جنگ و Pac-Man
۷ - Traditional video games

نام بازی	تعداد بازیکن	کلیت قوانین	شروط برد و باخت
شطرنج	دو	محیط بازی، مهره‌ها (رنگ و تعداد آن‌ها)، حرکت هر مهره، نوبت‌دهی	کیش‌ومات‌کردن حریف شرط برد است.
زو	دو تیم (هر تیم یک یا چند نفر)	ساختار زمین بازی، ورود به زمین حریف، خروج از زمین حریف	باید تمام افراد تیم مقابل را از بازی خارج کرد.
فوتبال	دو تیم (هر تیم ۱۱ نفر)	ساختار زمین بازی، آفساید، مفهوم گل، مدت بازی، حاکمیت داور، و...	تیمی که در مدت مقرر بیشتر گل زده باشد برنده است.
بازی کامپیوتری ماریو	یک نفر در مقابل کامپیوتر	ساختار مراحل، ساختار دشمنان، اعمال بازیکن اصلی، برخورد بازیکن با دشمن، برخورد با دشمن اصلی (Boss)	باید مرحله را تمام کرد و معشوقه‌ی خود را از دست دشمن نجات داد؛ برخورد با دشمن یا سقوط شرط باخت است.
ماشین سواری	دو و بیشتر	ساختار مسیر مسابقه، قوانینی مربوط به ماشین‌ها، و...	نفر اول برنده‌ی بازی است.

بخش دوم:
آن‌چه باید در مورد بازی‌های کامپیوتری بدانیم

بازی‌های کامپیوتری گونه‌ی بسیار پیشرفته، پیچیده و مدرن از بازی در دنیای امروز هستند. سازندگان بازی‌ها به‌کمک دنیای دیجیتال می‌توانند رؤیاهای خود را گسترش دهند. آن‌ها علاوه بر این‌که می‌توانند بازی‌های سنتی و ساده‌ی خود را در محیط دیجیتال با کیفیتی بالاتر و گستردگی بیشتر تولید کنند (مثل بازی شطرنج که به‌صورت آنلاین در سراسر دنیا برگزار می‌شود)، می‌توانند آن‌چه در دنیای واقعی قابل رخدادن نیست را نیز به مخاطب عرضه کنند (بازی‌های علمی-تخیلی از این دست هستند)، و از این جهت، صنعت بازی‌های کامپیوتری گرایش‌هایی به صنعت سینما نیز داشته است.

گفتیم که این رسانه می‌تواند چیزهایی فراتر از واقعیت را به مخاطب عرضه کند. همین امر راه را برای ورود هنر به این عرصه هموار نموده است. در بازی‌های امروزی، هنرهایی نظیر نقاشی دیجیتال، مجسمه‌سازی دیجیتال، فیلم‌نامه‌نویسی، موسیقی و... به‌کار گرفته می‌شود. در سال‌های اخیر، علومی نظیر علم احتمالات ریاضی و علم روان‌شناسی نیز راه خود را به دنیای بازی پیدا کرده‌اند. به‌طور کلی می‌توان بازی‌های کامپیوتری را رسانه‌ای دانست که درگیری و دخالت مخاطب در آن بیش از هر رسانه‌ی دیداری و شنیداری دیگر است.

در این بخش به بررسی ستون‌های اصلی بازی‌های کامپیوتری می‌پردازیم و در مورد هر بخش توضیح مختصری ارائه می‌دهیم.

ساختار کلی یک بازی ویدیویی

به‌طور جامع، یک بازی کامپیوتری به سه بخش اصلی تقسیم می‌شود:

۱. Design یا همان طرح بازی
۲. Art یا همان نمایش هنری بازی
۳. Tech یا همان قسمت فنی بازی

طرح بازی:

گام اول طرح بازی پاسخی به سه پرسش اصلی تعریف بازی است که در بخش قبل آن را بررسی کردیم:

۱. بازی برای چند نفر طراحی شده است؟
۲. قوانین بازی چیست؟
۳. شروط برد و باخت چه هستند؟

در گام بعدی باید فضای کلی بازی برای افرادی که قرار است آن را بسازند مشخص شود. مثلاً این‌که بازی در کدام یک از سبک‌ها می‌گنجد (اکشن، کمدی، ماجرایی و...). برای مشخص‌کردن فضای بازی نیاز است طراح اطلاعاتی - هرچند مختصر - در مورد این سبک‌های مختلف داشته باشد.

در گام بعدی، طراح باید شکل هنری بازی را مشخص کند. در این بخش، طراح باید با افرادی مختصص در زمینه‌ی هنر بازی مشورت داشته باشد. داستان بازی - در صورت نیاز - و نحوه‌ی روایت آن نیز در این قسمت نهایی خواهد شد. در نهایت، طرح باید شامل توضیحاتی - هرچند اندک - در مورد شیوه‌ی فنی اجرای بازی باشد. در این بخش، طراح باید با افراد متخصص در زمینه‌ی فنی مشورت کند.

پس از گذر از مراحل بالا، آن‌چه حاصل می‌شود توضیحاتی تقریباً کامل درباره‌ی بازی‌ای است که قرار است ساخته شود. لازم به ذکر است که این توضیحات باید هر سه بخش طراحی، فنی و نمایش هنری را پوشش دهد. به گزارشی که شامل این توضیحات است GDD [۱] گفته می‌شود. به‌طور کلی دو نوع GDD برای یک بازی وجود دارد:

نوع اول: GDD - 10 pager

در این نسخه تیم بازی‌سازی باید طی ۱۰ صفحه خلاصه‌ای از تمام آن‌چه در بازی قرار است اتفاق بیفتد را بازگو کند. این ۱۰ صفحه باید هر سه بخش طراحی، فنی و نمایش هنری را به‌طور خلاصه (حتی تیتروار) پوشش دهد.

نوع دوم: GDD - Design/Art

در این نسخه به‌طور مفصل هر آن‌چه در دو بخش طرح و نمایش هنری قرار است در بازی به‌کار گرفته شود توضیح داده خواهد شد.

GDD نوع اول بیشتر برای کارهای کوچک و پروژه‌های کوتاه استفاده می‌شود. اما هرچه پروژه بزرگ‌تر و طولانی‌تر باشد، نیاز به GDD مفصل‌تری داریم. لازم به ذکر است که در بعضی موارد خاص، یک GDD برای بخش فنی نیز نوشته می‌شود. در بازی‌هایی که مقیاس کوچکی دارند، صرفاً نوشتن یک 10 Pager کفایت می‌کند.

۱ - Game Design Document

نمایش هنری بازی:

در این بخش، گام اول پس از مشخص‌شدن ایده‌ی اصلی بازی (مرحله‌ی اول بخش طراحی) برداشته می‌شود. بخش نمایش هنری باید فضای کار (این فضا می‌تواند رئالیستی، سوررئالیستی، مینیمالیستی یا... باشد)، نحوه‌ی اجرای آن، ابزار مورد نیاز برای اجرای ایده و... را مشخص کند. فضای کار شامل شکل شخصیت‌ها، شکل محیط (متناسب با سبک بازی)، نحوه‌ی روایت داستان (در صورت وجود)، موسیقی، تکنیک‌های اجرایی، نحوه‌ی اجرای انیمیشن (در صورت نیاز) و... می‌شود.

بخش نمایش هنری باید با بخش فنی هماهنگی ویژه‌ای داشته باشد. چرا که نمایش هر آن‌چه در بخش هنری تولید می‌شود و در بخش طراحی برای آن هدف‌گذاری می‌شود، بر عهده‌ی بخش فنی خواهد بود.

بخش فنی بازی:

پس از مشخص‌شدن ایده‌ی بازی، در بخش فنی در مورد نحوه‌ی اجرای بازی و ابزارهای مورد نیاز آن تصمیم‌گیری می‌شود. این تصمیم‌گیری بر دو بخش دیگر تاثیرگذار است، پس باید با آن‌ها هماهنگ باشد. تیم فنی معمولاً قبل از نهایی‌شدن بخش هنری، با گرافیکی ساده بازی را پیش می‌برد تا ایرادات مربوط به طراحی زودتر از نهایی‌شدن نمایش هنری مشخص شده و از دوباره‌کاری جلوگیری شود.

ذکر دو نکته در این‌جا لازم است:

نکته‌ی اول) در گروه‌های بازی‌سازی کوچک، افراد معمولا بیش از یک نقش به‌عهده می‌گیرند (مثلا در بخش هنری و طراحی، افرادی مشترک هستند).

نکته‌ی دوم) هر بازی نیاز به GDD دارد.

اشاره به این نکته ضروری است که تولید یک بازی با کیفیت بالا فرآیندی وقت‌گیر است. اکثر بازی‌هایی که در بازار روز دنیا با کیفیت بسیار بالا عرضه می‌شود حاصل بیش از ۱۰ سال کار گروه‌های بزرگ بازی‌سازی است. هدف گروه‌های بازی‌سازی کوچک تولید یک محصول قابل عرضه و با کیفیت مطلوب است. ساختارمندبودن این محصول خود موجبات پیشرفتش را در آینده فراهم می‌کند.

در ادامه، رشد و توسعه‌ی یکی از معروف‌ترین بازی‌های مبارزه‌ای دنیا یعنی Mortal Kombat را مشاهده می‌کنیم. در این‌جا اثرگذاری ساختارمندبودن پروژه روی کیفیت پیشرفت آن کاملاً مشهود است.

نگاهی به روند پیشرفت بازی Mortal Kombat:

نسخه‌ی اول این بازی در سال ۱۹۹۲ وارد بازار شد. این بازی مبارزه‌ای دارای هفت کاراکتر مبارز بود.

نسخه‌ی دوم این بازی در سال ۱۹۹۳ ارائه شد. تعداد کاراکترها به ۱۲ عدد افزایش پیدا کرد و کیفیت بازی نیز بهبود یافت.

نسخه‌ی سوم این بازی در سال ۱۹۹۵ با ۲۲ کاراکتر وارد بازار شد. این نسخه در اوج کیفیت بازی‌های دوبعدی دوره‌ی خود بود.

نسخه‌ی چهارم نیز در سال ۱۹۹۷ با ۱۵ کاراکتر به بازار آمد. با ساخت این نسخه، این بازی نیز وارد رقابت بین بازی‌های سه‌بعدی شد.

پس از ارائه‌ی چهار نسخه‌ی دیگر Mortal Kombat: Deadly Alliance در سال ۲۰۰۲ با گرافیک قابل قبول و ۲۲ شخصیت وارد بازار شد.

پس از انتشار شش نسخه با کیفیتی مشابه، در سال ۲۰۱۱ نسخه‌ی جدید این بازی با نام Mortal Kombat با گرافیکی بسیار زیبا و ۳۶ شخصیت منتشر شد.

هم‌اکنون نسخه‌ی Mortal Kombat X وارد بازار شده است که از نظر کیفی در رتبه‌ی بالایی قرار دارد.

سخن آخر

صرف‌نظر از حمایت‌های مالی که توسط ناشران از این بازی شده است، کندبودن روند پیشرفت در این صنعت مشهود است، اما آن‌چه اهمیت دارد ساختارمندی این پیشرفت است. در دوره‌هایی که سازندگان این بازی توانایی ارائه‌ی کیفیت جدیدی از بازی را نداشتند، صرفاً به تولید یک نسخه‌ی جدید با همان کیفیت اکتفا می‌کردند. این اتفاق یک‌بار بین سال‌های ۱۹۹۷ تا ۲۰۰۲ و یک‌بار بین سال‌های ۲۰۰۳ تا ۲۰۱۱ برای این بازی رخ داده است.

نکته‌ی حائز اهمیت آن است که اگر نسخه‌های قبلی ساختارمند نبودند، قابلیت انعطاف‌پذیری نسبت به ایده‌های جدید را نداشتند و در نتیجه چنین سرعتی در تولید نسخه‌ها با کیفیت یکسان امکان‌پذیر نبود و تولید یک نسخه‌ی جدید بیشتر به‌طول می‌انجامید. رعایت این نکته در تولید یک بازی بسیار ضروری است. بازی باید قابلیت گسترش داشته باشد و گسترش آن نباید هزینه‌ای به اندازه‌ی تولید آن برای ما داشته باشد. همچنین این گسترش نباید روی دیگر بخش‌های بازی که قبلا طراحی شده است تاثیر منفی بگذارد (مثلاً باعث حذف آن‌ها یا تغییر پایه‌ای آن‌ها شود).

در این مقاله کلیتی از ساختار بازی ارائه شد و با ذکر مثالی اهمیت ساختارمندی ایده نیز مطرح شد. علاقه‌مندان به حرفه‌ی بازی‌سازی باید ابتدا ساختار تولید بازی را بشناسند، با اجزای سازنده‌ی آن آشنا باشند، علاقه‌مندی یا مهارت خود را در این ساختار پیدا کنند و همیشه این نکته را در نظر داشته باشند که بازی‌های ویدیویی یک هنر-صنعت است و به همان اندازه که وجود افراد فنی در ساختار نیاز است، وجود افراد هنری نیز حائز اهمیت است. یک ایده‌ی خوب الزاماً منجر به تولید یک بازی خوب نمی‌شود. این ساختارمندیِ تولید و هماهنگی میان اجزای معرفی‌شده است که یک بازی خوب را به مخاطب ارائه می‌دهد. در گروه‌های بازی‌سازی هیچ بخشی بر دیگری برتری موقعیتی ندارد و تعادل میان سه بخش اصلی بازی یعنی بخش فنی، هنری و طراحی بازی است که بهترین نتیجه را در پی خواهد داشت.

همه‌ساله جشنواره‌ها و مسابقاتی در سراسر جهان در زمینه‌ی بازی‌سازی برگزار می‌شود و این فرصتی مناسب برای آشنایی با این هنر-صنعت و شناخت یا بازشناسی مهارت‌های فرد در این عرصه است. شرکت در این مسابقات، علاوه بر جاذبه‌های سرگرم‌کننده‌ای که در حین کار دارد، می‌تواند پلکانی برای پیشرفت در این زمینه و ورود به بازار جهانی بازی‌های ویدیویی نیز باشد.

برنامه‌نویسی وب را از کجا آغاز کنیم؟

همواره افراد برای شروع یک زبان برنامه‌نویسی جدید و نحوه‌ی یادگیری آن دچار مشکل هستند. یکی از بزرگ‌ترین این مشکلات که در هنگام شروع برنامه‌نویسی وب نیز برای افراد مختلف پدید می‌آید، پیداکردن منابع مناسب است. به‌عنوان یکی از اولین راه‌ها، می‌توان از جست‌وجو در گوگل و وبسایت‌های دیگر نام برد. این کار سبب می‌شود تا دیدتان نسبت به مسائلِ حول موضوع جست‌وجوشده وسیع‌تر شود و محدود به یک موضوع خاص نباشد. ولی از طرفی، این کار زمان بیشتری را نسبت به خواندن یک منبع مشخص از شما خواهد گرفت.

سهیل شیری
soheil.shiri@gmail.com

زبان‌های HTML و CSS ریشه و پایه‌ی اینترنت‌اند و تمامی صفحات وب به زبان HTML نوشته شده است. این زبان به مرورگر می‌گوید که چه اجزایی را هنگام ورود به وبسایت برای کاربر نمایش دهد. برای مثال، به‌وسیله‌ی این زبان مشخص می‌شود که تعدادی عکس و لینک در یک صفحه کنار یکدیگر قرار بگیرند. وظیفه‌ی CSS تعریف شکل ظاهری و تنظیم نحوه‌ی چینش اجزایی است که به‌وسیله‌ی HTML در صفحه قرار گرفته‌اند. برای هر برنامه‌نویس وب ضروری است که به این دو زبان و نحوه‌ی ارتباط آن‌ها با یکدیگر مسلط باشد.

یکی از بهترین منابع از نظر بسیاری از متخصصان برای آموزش HTML و CSS وبسایت shayhowe[1] است. پس از مطالعه و پیداکردنِ دید کلی نسبت به مباحث، توصیه می‌شود حتماً یک وبسایت با توجه به مطالبی که آموخته‌اید طراحی کرده و فقط به خواندن مطالب آموزشی اکتفا نکنید. مثلاً وبسایتی را به‌طور دلخواه باز کرده و سعی کنید خودتان مانند آن را پیاده‌سازی کنید. یکی از ابزارهایی که در آموزش HTML و CSS به شما بسیار کمک می‌کند ابزار توسعه‌دهندگان کروم[2] است. از طرفی، برنامه‌نویسان دیگر تعدادی چهارچوب[3] را به‌صورت متن‌باز برای استفاده‌ی عموم تحت عنوان UI framework ارائه کرده‌اند که به طراحی وبسایت بسیار کمک می‌کند. از معروف‌ترین این چهارچوب‌ها می‌توان به Bootstrap و Semantic اشاره نمود.

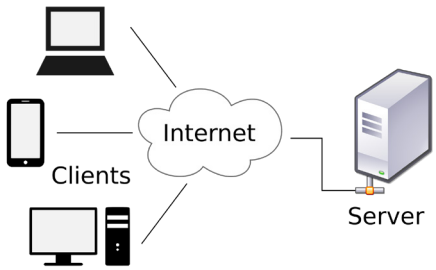
پس از یادگیری HTML و CSS، نوبت به جاوااسکریپت می‌رسد. جاوااسکریپت زبانی است که به‌وسیله‌ی آن اجزا و صفحاتی که توسط HTML ساخته شده‌اند می‌توانند با یکدیگر تعامل داشته باشند. برای مثال، به‌وسیله‌ی این زبان می‌توان تعیین کرد هنگامی که روی عبارت مشخصی کلیک می‌شود، صفحه به بالا اسکرول شود. از طرفی، جاوااسکریپت برخلاف زبان‌های قبلی می‌تواند محاسبات منطقی انجام دهد که به‌نوعی تفسیرشدن[4] آن‌ها توسط خود مرورگر صورت می‌گیرد. برای نمونه، توسط این زبان می‌توان دو عبارت را در صفحه خواند و با یکدیگر جمع نمود. وبسایت توسعه‌دهندگان موزیلا[5] یکی از کامل‌ترین مراجع برای آموزش جاوااسکریپت است.

جاوااسکریپت دارای چهارچوب‌ها و کتاب‌خانه‌های بی‌شماری است که یادگیری آن‌ها خالی از لطف نیست. از معروف‌ترین این کتاب‌خانه‌ها می‌توان jQuery را نام برد که هدف آن کوتاه‌کردن و ساده‌سازی دستورات جاوااسکریپت است. یکی دیگر از چهارچوب‌های معروف این زبان، Angular JS است که به‌وسیله‌ی آن می‌توان وب‌اپلیکیشن‌هایی را با صفحه‌های پویا به‌صورت بسیار ساده تولید نمود. پیشنهاد می‌شود قبل از رفتن به مراحل بعدی آموزش، یک ماشین‌حساب را با استفاده از این سه زبان پیاده‌سازی کنید تا مطالبی که خوانده‌اید برای‌تان روشن‌تر و قابل‌استفاده شود.

برنامه‌نویسی وب به‌طور کلی به دو بخش front-end و back-end تقسیم می‌شود. front-end در سمت کاربر است و هدف آن در واقع چیزی است که به کاربرانی که وارد وبسایت شده‌اند نمایش داده می‌شود. back-end در سمت سرور قرار دارد و هدف آن پردازش‌هایی است که در سرور صورت می‌گیرد تا به درخواست کلاینت پاسخ داده شود. با توجه به علایق‌تان می‌توانید هر یک از این فیلدها را به‌صورت تخصصی دنبال کنید. برای این‌که توسعه‌دهنده‌ی سمت کاربر شوید، باید به تسلط خوبی بر زبان‌های HTML، CSS و جاوااسکریپت برسید و درباره‌ی چهارچوب‌های آن‌ها بیشتر بدانید. علاوه بر این‌ها، برنامه‌نویس front-end به‌دلیل طراحی صفحه‌ای که کاربر مشاهده می‌کند، باید با اصول طراحی و تجربه‌ی کاربری نیز آشنا باشد.

برنامه‌نویس سمت سرور علاوه بر آشنایی با HTML، می‌تواند برای محاسبات و پردازش‌های انجام‌شده در سرور، از زبان‌های مختلفی مانند ASP، PHP، Ruby و... استفاده کند. به‌طور کلی، برنامه‌نویس باید بستری را درون سرور فراهم آورد تا نیازهای سمت کلاینت را برطرف کند. برای مثال، هنگامی که کاربری بخواهد در وبسایت ثبت‌نام کند، پس از آن‌که اطلاعات از طریق کلاینت فرستاده‌شد، کار سرور ثبت اطلاعات پس از بررسی مسائل امنیتی و بررسی درستی اطلاعات فرستاده شده است.

به‌عنوان آخرین پیشنهاد، برای آموزش و یادگیری هر زبان برنامه‌نویسی، سعی کنید پروژه‌های مختلفی را برای خود انجام دهید و همواره سخت‌ترین و پیچیده‌ترین پروژه‌هایی را که به ذهن‌تان می‌رسد انتخاب کنید؛ زیرا در حین انجام پروژه به مشکلات زیادی برخورد خواهید کرد و برای حل آن‌ها مطالب زیادی خواهید آموخت که شاید با خواندن صرف مطالب هیچ‌گاه با آن‌ها مواجه نمی‌شدید.



۱ - http://learn.shayhowe.com/HTML-CSS
۲ - Chrome Developer Tools
می‌توانید راهنمایی مناسبی برای این ابزار را در وبسایت www.hongkiat.com/blog/chrome-developer-tools بیابید.
۳ - Framework
۴ - Interpret
۵ - http://developer.mozilla.org

وقتی از استارت‌آپ حرف می‌زنیم، از چه حرف می‌زنیم؟[1]

هدف این نوشته نصیحت و کلی‌گویی نیست، بلکه ارائه‌ی تصویری است از زندگی در فضای کارآفرینی ایران. با توجه به این‌که مخاطب اصلی این نوشته دانش‌جویان رشته‌های مهندسی برق و کامپیوتر هستند و تجربه‌ی حرفه‌ای نگارنده هم در همین فضای ICT بوده، خواه‌ناخواه مطالب گفته‌شده با در نظر داشتن استارت‌آپ‌های همین حوزه نوشته شده است؛ هرچند از دید نگارنده بسیاری از مواردْ قابل تعمیم به استارت‌آپ‌های مرتبط با فناوری‌های دیگر نیز هست. در مورد لذت‌های بسیارِ خلق یک کسب‌وکار جدید مقاله‌های زیادی نوشته شده است، در نتیجه، تمرکز این نوشته بیشتر بر نشان‌دادن بعضی از واقعیت‌های عمیق‌تر این نوع زندگی در قالب پاسخ به چند سؤال رایج است.

احمدعلی فرهودی – مدیرعامل شرکت صباسل و عضو هیئت‌مدیره‌ی شرکت پارس‌سامان
farhoodi@SabaCell.ir

برخلاف تصور رایج در جامعه، استارت‌آپ مفهوم جدیدی نیست و سابقه‌ای به‌قدمت کار تجاری بشر دارد. به‌احتمال خوبی می‌توان حدس زد که در زمان غارنشینیِ اجداد ما هم افرادی تلاش می‌کرده‌اند که مستقل از ساختارهای تثبیت‌شده‌ی شکار و کشاورزی، خودشان سازمان و تیم برای زنده‌ماندن درست کنند. در فضای تجاری-صنعتی مدرن کشورِ ما هم این پدیده دست‌کم قدمتی ۵۰ساله دارد و انبوهی از تجربه‌های تلخ و شیرین برای بنیان‌گذاران این شرکت‌ها به‌همراه داشته است. اما چیزی که در ۱۵ سال اخیر (پس از سال ۲۰۰۰ میلادی) تیترهای خبری بسیاری از رسانه‌ها را ساخته، موج بزرگ رشد استارت‌آپ‌های حوزه‌ی فناوری اطلاعات و ارتباطات است که در رسانه‌های غیرتخصصی، عبارت کلی Tech Start-Up برای توصیف آن‌ها استفاده می‌شود.

آیا همه‌ی استارت‌آپ‌ها High-Tech هستند؟

خیر. این یکی از اشتباهات رایج در زمینه‌ی کارآفرینی است. بسیاری از شرکت‌های نوپای مشهور در ایران و جهان از فناوری‌های جدید (مثلاً روش‌های جدید برنامه‌نویسی) استفاده می‌کنند، ولی کار آن‌ها توسعه‌ی تکنولوژی نیست. در نتیجه، از نظر فنی، یک فروشگاه اینترنتی یا سرویس اشتراک‌گذاری عکس یک کسب‌وکار High-Tech محسوب نمی‌شود. در این موارد، تکنولوژی‌های جدید اصطلاحاً توانمندساز[2] کسب‌وکار هستند.

چرا این مسئله مهم است؟ اگر شما ذوق و شوق و انرژی توسعه‌ی تکنولوژی دارید، نباید به‌دنبال این قبیل کسب‌وکارها که به‌اشتباه High-Tech خوانده می‌شوند بروید، چون ممکن است بعد از مدت کوتاهی دچار افسردگی شوید. در حقیقت این قبیل کارها از نظر فکری شما را ارضا نخواهد کرد.

آیا من باید استارت‌آپ راه بیندازم؟

به این سؤال فقط خودتان می‌توانید جواب بدهید. در ادامه تلاش شده است تا چند مورد مهم در این زمینه روشن شود:

۱. راه‌انداختن یک کسب‌وکار نوپا از نظر عاطفی و مالی دارای ریسک بالایی است ولی در مقابل احتمال بازگشت سرمایه و خلقِ ثروت زیادی وجود دارد. همان‌طور که موفقیت صرفاً جنبه‌ی مالی ندارد، شکست هم می‌تواند اثرات مالی و احساسی زیادی بر زندگی شما داشته باشد. لذت راه‌انداختن یک استارت‌آپ، لذت خلق و رشد است. در صورتی وارد این مسیر شوید که در خودتان ظرفیت احساسی و عاطفی خوبی می‌بینید.

۲. اگر به‌دنبال یک زندگی ساده و کم‌دغدغه هستید و نیاز به آرامش دارید، سراغ این کار نروید. بزرگ‌کردن یک استارت‌آپ هیچ‌چیزی کم از بزرگ‌کردن یک نوزاد ندارد. بی‌خوابی، استرس و سطح بالای آدرنالین، غم‌وغصه‌ی ناشی از بیماری این نوزاد و حتی مرگ آن حداقل اتفاقاتی هستند که باید منتظرش باشید. وقتی به راه‌اندازی یک کسب‌وکار جدید فکر می‌کنید، در نظر داشته باشید که باید لااقل ۱۰ سال از بهترین دوران زندگی‌تان را صرف آن کنید. اگر قصد ادامه‌ی تحصیل در خارج از کشور در چند سال آینده را دارید، راه‌اندازی استارت‌آپ کار شما نیست.

۳. هیچ تضمینی وجود ندارد که شرکت شما موفق شود و حتی بعد از پنج سال، درآمد شرکت کفاف حقوق کارمندانش را بدهد. بعضی از گزارش‌ها نرخ شکست استارت‌آپ‌های دره‌ی سیلیکون در آمریکا را تا ۹۰٪ تخمین می‌زنند. آمار قابل استنادی از این نرخ در ایران وجود ندارد، اما انتظار نمی‌رود که این نرخ با آمارهای جهانی فاصله‌ی زیادی داشته‌باشد. در مقابل، موارد زیادی در جهان و موارد کمتری در ایران وجود دارد که یک تیم خوب و جوان موفقیت‌های بزرگی کسب کرده است.

۴. مهم‌تر از ایده و سرمایه، داشتن تیم خوب است و بدون آن ورود به دنیای استارت‌آپ منطقی نیست. در ادامه در این مورد بیشتر بحث خواهد شد.

۱- تقلیدی از نام داستان مشهور ریموند کارور: وقتی از عشق حرف می‌زنیم، از چه حرف می‌زنیم؟
(What We Talk About When We Talk About Love؟)

۲- Enabler

مگر من کمتر از جابز و زاکربرگ هستم؟

ذهن انسان به‌صورت پایه‌ای در محاسبه‌ی احتمال رخدادها ضعیف عمل می‌کند. داستان‌های نقل‌شده از کارآفرینان افسانه‌ای باعث محاسبه‌ی احتمال موفقیت از طریق یک قانون سرانگشتی به نام Availability Heuristic در ذهن ما می‌شود. به این معنی که وقتی به راه‌انداختن استارت‌آپ فکر می‌کنیم، نمونه‌های مشهور موفق که احتمالاً در یکی-دو روز اخیر در موردشان خبری خوانده‌ایم به‌یادمان می‌آید و این به‌طور خودکار باعث می‌شود احتمال موفقیت بالاتر به‌نظر برسد. همین پدیده در تصمیم‌گیری خیلی از افراد برای اپلای‌کردن مؤثر است. شما به‌احتمال زیاد از نظر هوش و استعداد چیزی کمتر از زاکربرگ و گیتس و جابز و کردستانی ندارید، ولی این معنی خاصی ندارد چون پارامترهای مهمی مانند ریسک‌های محیطی، فراز و نشیب‌های بازار، شانس و اقبال و نظایر آن را نمی‌توان با شاخص بهره‌ی هوشی اندازه‌گیری کرد.

تیم خوب چگونه است؟

تیم خوب یعنی تیمی که توانایی‌های اعضای آن مکمل هم باشد یا به‌اصطلاح گروه خونی آن‌ها به هم بخورد. کار تیمی نیازمند ظرفیت، ارتباط شفاف و مناسب، دلسوزی و پشتیبانی از یکدیگر است. به‌طور ساده، یک تیم خوبِ استارت‌آپی باید لااقل شامل یک نفر با توانایی‌ها و علاقه به مدیریت تجاری و یک نفر با توانمندی مدیریت فنی و توسعه‌ی تکنولوژی باشد. این گزاره اصلاً به‌معنی کافی‌بودن یک تیم دونفره در همه‌ی موارد نیست. برای کارهای بزرگ، گاهی لازم است تیم بنیان‌گذار بزرگ‌تر و دارای توانایی‌های متنوع‌تر باشد.

فراموش نکنید که یک سرمایه‌گذار حرفه‌ای (مثلاً یک سرمایه‌گذار خطرپذیر[۳] یا یک شتاب‌دهنده[۴]) قبل از هر چیز، تیم شما را بررسی می‌کند، نه ایده یا مدل تجاری‌تان را. یک تیم خوب در طول زمان انعطاف‌پذیری لازم برای خلق ایده‌های جدید و مقابله با طوفان‌های محیط را پیدا می‌کند.

آیا حتماً طرح تجاری[۵] لازم دارم؟ مدل تجاری[۶] من چگونه باشد؟

در صنایعی که ثبات نسبی از نظر بازار و فناوری دارند (مثل صنعت سیمان یا پتروشیمی)، داشتن طرح تجاری خوب برای هر کاری - هرچند کوچک - بسیار ضروری است. در بسیاری از زمینه‌های مربوط به فناوری‌های نو مانند ICT، داشتن طرح تجاری با جزئیات زیاد کاملاً بی‌معنی است. سرعت تغییرات در این حوزه‌ها به‌حدی است که در بازه‌ی یکی-دوماهه‌ی تهیه‌ی طرح تجاری ممکن است تغییرات محیطی زیادی رخ دهد. سرمایه‌گذاران حرفه‌ای هم در این زمینه به جزئیات طرح تجاری توجه نمی‌کنند.

این نکته اصلاً به معنی آن نیست که استارت‌آپ یک مدل تجاری مناسب لازم ندارد و کل کار باید بدون برنامه پیش برود. یک تیم استارت‌آپ باید ایده‌ای مناسب - هرچند نه‌چندان دقیق - از محصول نهایی خود، بازار هدف و مدل کسب درآمد داشته باشد که به این ترکیب اصطلاحاً مدل تجاری گفته می‌شود. مدل تجاری در بسیاری از موارد پس از گذشت مدت کوتاهی از شروع کار دچار تحولات انقلابی می‌شود، ولی در هر مرحله، داشتن چهارچوبی از آن به‌عنوان قطب‌نمای مسیر توسعه لازم است.

مدل‌های تجاری متداول در ایران، که در بسیاری از موارد کپی ناقصی از مدل‌های موجود در کشورهای دیگر هستند، لزوماً مدل‌های مناسب و پایداری نیستند. در صورت در نظر نگرفتن شرایط خاص کشور ما، این مدل‌ها می‌توانند به‌سادگی شکست بخورند، ضمن این‌که بعضی از نمونه‌های موفق فعلی نیز به‌دلیل نداشتن یک عنصر خاص متناسب با جامعه‌ی ایران، قدرت رقابت در شرایط بعد از رفع تحریم‌ها را نخواهند داشت.

یک نگاه خوب برای استارت‌آپ‌ها، داشتن یک مدل تجاری صادرات‌محور است. حوزه‌ی ICT به خیلی از شرکت‌های کوچک امکان می‌دهد که بدون سرمایه‌گذاری بالا بتوانند از خارج از کشور خود درآمدهای خوبی کسب کنند.

> کار تیمی نیازمند ظرفیت، ارتباط شفاف و مناسب، دلسوزی و پشتیبانی از یکدیگر است. به‌طور ساده، یک تیم خوبِ استارت‌آپی باید لااقل شامل یک نفر با توانایی‌ها و علاقه به مدیریت تجاری و یک نفر با توانمندی مدیریت فنی و توسعه‌ی تکنولوژی باشد.

آیا من حتماً باید مدرک MBA داشته باشم؟

خیر. دوره‌های MBA فقط می‌توانند فرآیند یادگیری کسب‌وکار را برای شما «تسریع» کنند، یعنی کمک کنند که با داشتن یک تجربه‌ی مقدماتی و ترجیحاً یک مورد شکست تجاری، نحوه‌ی رشد یک کسب‌وکار جدید را زودتر یاد بگیرید. افرادی که به شما می‌گویند داشتن مدرک MBA رمز موفقیت است یا صداقت ندارند یا نتوانسته‌اند مفهوم آموزش کسب‌وکار را درک کنند.

من ایده‌ی خوبی دارم؛ ارزش مالی آن چه‌قدر است؟

این سؤالی است که برای بسیاری از دانش‌جویان و فارغ‌التحصیلان جوان پیش می‌آید. واقعیت این است که ارزش‌گذاری روزِ یک ایده چندان معنی‌دار نیست و بعید است که فردی حاضر باشد روی یک ایده‌ی بدون تیم سرمایه‌گذاری کند. به بیان دیگر، پاسخ‌دادن به این سؤال اگر امکان‌پذیر هم باشد به نتیجه‌ی خاصی منجر نمی‌شود. در سال‌های گذشته، تلاش‌هایی نظیر بورس ایده در جهت تشکیل بازار برای ایده‌های تجاری-فنی صورت گرفت که با شکست مواجه شد.

سرمایه‌ی لازم را از کجا بیاورم؟

برای شروع هر کاری، باید تخمین قابل قبولی (مثلاً با ۳۰٪ خطا) از سرمایه‌ی لازم برای مراحل اول تا رسیدن به یک حداقل محصول قابل عرضه[۷] داشته باشید. بسیاری از ایده‌های اینترنتی سرمایه‌ی زیادی لازم ندارند و فقط باید حقوق کارکنان را پرداخت کنند، یعنی اگر بنیان‌گذاران توان فنی و تجاری کافی را داشته باشند و نیازی به دریافت حقوق در بازه‌ی یکی-دوساله نداشته باشند، این هزینه به‌سمت صفر میل می‌کند.

در جهان، معمولاً هزینه‌های ابتدای کار توسط دوستان و خانواده تأمین می‌شود و بعد از آن سرمایه‌گذاران حقیقی و حقوقی مختلف در مراحل متفاوت رشد شرکت در آن سرمایه‌گذاری می‌کنند. اکوسیستم تأمین مالی استارت‌آپ‌ها (سرمایه‌گذاری‌های خطرپذیر) در ایران در حال شکل‌گیری است. در حال حاضر امکان استفاده از خدمات شتاب‌دهنده‌های مختلف و چند صندوق سرمایه‌گذاری فراهم شده است و در آینده بیشتر هم خواهد شد. بعضی از پارک‌های علم و فناوری نیز امکانات خوبی از قبیل دفتر کار، اینترنت و مشاوره در اختیار استارت‌آپ‌ها قرار می‌دهند. همچنین نهادهای دولتی نظیر معاونت علمی و فناوری ریاست جمهوری و سازمان فناوری اطلاعات در بعضی مقاطع زمانی وام‌های کم‌بهره در اختیار این شرکت‌ها می‌گذارند.

حرف آخر

با ۱۴ سال سابقه‌ی کار در استارت‌آپ‌های حوزه‌ی مخابرات و فناوری اطلاعات و مشارکت در راه‌اندازی دو شرکت از نقطه‌ی صفر تا سودآوری مناسب، مقصود نگارنده از نوشتن این متن اصلاً دل‌سردکردن خواننده نبود. هدف این بود که در شرایطی که «تب» استارت‌آپ دانش‌جویان و فارغ‌التحصیلان را گرفته و احتمال «لرز» بعد از آن هم بسیار بالاست، بعضی از پیچیدگی‌های این سبک زندگی به خواننده نشان داده شود تا با دید بازتری امکان انتخاب داشته باشد. از دید نگارنده رشد کشور ما جز از طریق توسعه‌ی استارت‌آپ‌های موفق و صادرات‌محور در زمینه‌های مختلف فناوری ممکن نیست، و این کار نیازمند صبر، توان تحمل بالا و هوش استراتژیک خوب است.

۳- Venture Capital (VC)
۴- Accelerator
۵- Business plan
۶- Business model
۷- Minimum Viable Product

فلسفه‌ی آزادی‌خواهی در نرم‌افزار: گذری بر یک مرام و نشیب

پویا مرادی
pooya.moradi@acm.org

۱. مقدمه

اگر بخواهیم نتایج حاصل از دهه‌ی اخیر را در صنعت توسعه‌ی نرم‌افزار نام ببریم، قطعاً یکی از این موارد مفهوم نرم‌افزار آزاد[۱] خواهد بود. هر شرکتی تلاش می‌کند که در پروژه‌های آزاد همکاری کند. هر تیمی در تلاش است بعضی قسمت‌های چهارچوب[۲] خود را تحت عنوان یک نرم‌افزار آزاد و متن‌باز[۳] در اینترنت منتشر کند. تازه‌کارهای دنیای لینوکس مدام از چیزی با عنوان نرم‌افزارهای آزاد دم می‌زنند. به‌راستی این نرم‌افزار آزاد چیست؟ متن‌بازبودن آن به چه معناست؟ چه مزایا و معایبی نسبت به یک نرم‌افزار غیرآزاد دارد؟ در این مقاله سعی خواهیم کرد که به برخی پرسش‌های سنتی از این قبیل پاسخ دهیم.

۲. تاریخچه

در دهه‌های ۵۰ و ۶۰ میلادی کامپایلرها و سیستم‌عامل یک کامپیوتر به‌همراه سخت‌افزار آن عرضه می‌شدند و به‌عنوان یک کالای جدا خریداری نمی‌شدند. در این سال‌ها، کد منبع این نرم‌افزارها به‌همراه این مجموعه توزیع می‌شد تا امکان رفع مشکل یا بهبود آن وجود داشته باشد. دانشگاه‌ها استفاده‌کنندگان اصلی این نرم‌افزارها بودند و در نتیجه، تغییرات گسترده‌ای در این نرم‌افزارها ایجاد می‌کردند که آن‌ها را با یکدیگر به‌اشتراک می‌گذاشتند. می‌توان گفت که این موضوع اولین مثال از نرم‌افزارهای آزاد بود. نرم‌افزارهایی که به گفته‌ی ریچارد استالمن[۴]، از رهبران جنبش آزادی نرم‌افزار[۵]، دارای ۴ آزادی اساسی هستند[۶]:

آزادی اول: شما اجازه دارید که نرم‌افزار را اجرا کنید، مستقل از این‌که چه هدفی دارید.

آزادی دوم: شما این آزادی را دارید که نرم‌افزار را متناسب با نیازهای خود تغییر بدهید. این تغییر مستلزم این است که به کد منبع آن دسترسی داشته باشید.

آزادی سوم: شما این آزادی را دارید که نرم‌افزار را توزیع کنید، خواه رایگان و خواه با دریافت هزینه.

آزادی چهارم: شما این آزادی را دارید که نرم‌افزار حاصل از اعمال تغییرات خود را توزیع کنید تا جامعه‌ی آن نرم‌افزار، از پیشرفت‌های شما بهره‌مند شود.

البته ناگفته نماند که مفهوم آزادی در به‌اشتراک‌گذاری از مدت‌ها قبل از پدیدآمدن کامپیوترها وجود داشته است. به‌عنوان مثال، دستور پخت یک غذا می‌تواند تغییر پیدا کند و به‌اشتراک گذاشته شود.

تبدیل ریچارد استالمن به رهبر جنبش آزادی نرم‌افزار به سال ۱۹۷۰ بازمی‌گردد. در آن زمان او در آزمایشگاه هوش مصنوعی دانشگاه MIT فعالیت می‌کرد. آن‌ها یک سیستم عامل تحت عنوان ITS برای آزمایشگاه خود طراحی کرده بودند که از آن برای تحقیقات‌شان استفاده می‌کردند. در آن سال‌ها، استالمن تصمیم به بهبود درایور پرینتری از شرکت زیراکس[۷] که در اختیار دانشگاه MIT بود گرفت؛ اما او قادر به انجام این کار نبود چرا که شرکت زیراکس کد منبع درایور را منتشر نمی‌کرد. چند سال بعد، سیستم‌عامل‌های اختصاصی جایگزین سیستم‌عامل‌هایی شدند که خود آن‌ها نوشته بودند. او شاهد سقوط جامعه‌ای بود که ساخته بودند و از این‌جا بود که جنبش آزادی نرم‌افزار شروع شد. او تصمیم گرفت سیستم‌عاملی کامل (مجهز به ویرایشگرها، کامپایلرها، و...) طراحی کند که با سیستم‌عامل تجاری یونیکس (سیستم‌عامل رایج آن دوران که توسط افرادی مانند دنیس ریچی[۸]، در آزمایشگاه بل توسعه یافته بود)

۱ - Free software
۲ - Framework
۳ - Open source software
۴ - http://en.wikipedia.org/wiki/Richard_Stallman
۵ - htto://en.wikipedia.org/wiki/Free_software_movement
۶ - http:// www.gnu.org/philosophy/free-sw.html
۷ - Xerox
۸ - خالق زبان C

> Proprietary software is an injustice


ریچارد استالمن

همخوانی داشته باشه تا برنامه‌نویسان بدون زحمت یادگیری خاصی، بتوانند از آن استفاده کنند.[۹] او نام گنو (GNU) را، که مخفف "GNU is not UNIX" است (که یک نام بازگشتی[۱۰]ست)، روی این پروژه گذاشت. تصمیم او برای انجام‌دادن این پروژه آن‌قدر جدی بود که از سِمَت خود در دانشگاه MIT استعفا داد.

او این پروژه را با نوشتن هسته[۱۱] شروع نکرد و آن را به آخر کار واگذار کرد[۱۲]. اولین پروژه‌ای که بر روی آن کار کرد، ویرایشگر گنو ایمکس[۱۳] بود که در سال ۱۹۸۵ اولین نسخه‌ی آن منتشر شد. استالمن این ویرایشگر را با پروتکل FTP به اشتراک گذاشت، اما به‌دلیل این‌که اینترنت در آن زمان چندان جا نیفتاده بود، او با دریافت ۱۵۰ دلار این نرم‌افزار را برای مردم ارسال می‌کرد و به‌واسطه‌ی هشت تا ده سفارشی که در ماه داشت، هزینه‌های زندگی‌ش تأمین می‌شد. سؤالی که در آن زمان اغلب از او پرسیده می‌شد این بود که چگونه این نرم‌افزار "free" است در حالی‌که ۱۵۰ دلار هزینه دارد؟ پاسخی که استالمن به این سؤال می‌داد این بود:

Think of free speech, not free beer.

به این مفهوم که آزادی در کارهایی‌ست که می‌توانید با این نرم‌افزار انجام بدهید (مثلا ویرایش و انتشار دوباره‌ی آن) و نه به‌معنی رایگان‌بودن.

دستاورد این کار استالمن، در کنار پدیدآمدن «جنبش آزادی نرم‌افزار»، افزایش تعداد برنامه‌نویسانی بود که از گنو ایمکس استفاده می‌کردند. به‌دلیل ماهیت آزادبودن این نرم‌افزار، برنامه‌نویسان شروع به گزارش باگ‌های آن کردند و همچنین به ارائه‌ی راهکارهایی برای رفع این باگ‌ها پرداختند. در کنار این موضوعات، او هر وقت که می‌توانست از پروژه‌های آزاد دیگران در پروژه‌ی گنو استفاده می‌کرد. به‌عنوان مثال، پروژه‌ی رابطه‌ی گرافیکی X[۱۴]، پروژه‌ای بود که از ابتدا توسط استالمن نوشته نشد بلکه از بیرون وارد پروژه‌ی گنو شد.

جنبش آزادی نرم‌افزار استالمن یک نوع جنبش اجتماعی بود (در رابطه با این موضوع، در قسمت‌های بعدی صحبت خواهیم کرد.). استفاده‌شدن نرم‌افزارهای آزاد در نرم‌افزارهای اختصاصی، مانع مستقیم این جنبش اجتماعی بود. دلیل این موضوع آن است که تغییرات اختصاصی در یک نرم‌افزار باعث انحصاری‌شدن نسخه‌ی ویرایش‌شده می‌شود، حال آن‌که استالمن می‌خواست تمام نسخه‌های پروژه‌های GNU همچنان آزاد بماند.

استالمن در سال ۱۹۸۵، در این رابطه گفت:

«هر کسی اجازه دارد که گنو را ویرایش و بازتوزیع کند اما هیچ توزیع‌کننده‌ای حق ندارد روی توزیع‌های بعدی محدودیت بگذارد. این به آن معنی‌ست که اجازه‌ی اعمال تغییرات انحصاری در نرم‌افزار داده نمی‌شود. من می‌خواهم مطمئن شوم که تمام نسخه‌های گنو آزاد باقی خواهند ماند.»

به‌عنوان مثال استالمن چند سال بر روی یک مفسر زبان لیسپ کار کرده بود. شرکت Symbolics از او خواست که اجازه‌ی استفاده از این مفسر را به آن‌ها بدهد. او نیز موافقت کرد. بعدها این شرکت بر روی مفسر کار کرد و آن را بهبود و توسعه داد. بعد از این توسعه‌ها، استالمن درخواست دسترسی به نسخه‌ی بهبود یافته و توسعه‌داده‌شده‌ی این مفسر را کرد، اما شرکت Symbolic با این کار موافقت نکرد.

از این رو استالمن به‌دنبال راهی بود که جلوی تغییرات اختصاصی و انحصاری‌شدن ویرایش‌های مختلف یک نرم‌افزار آزاد را بگیرد. برای این موضوع، او راهکاری تعیین کرد به‌صورتی که ابتدا یک نرم‌افزار دارای کپی‌رایت می‌شود و در نتیجه نمی‌تواند داخل یک نرم‌افزار اختصاصی انحصاری شود. سپس به مجوز بندهایی اضافه می‌شود تا به استفاده‌کننده از نرم‌افزار این اجازه را بدهد که آن را ویرایش کرده یا به‌شرط حفظ بندهای مربوط به توزیع نرم‌افزار آن را منتشر کند. کپی‌رایت به‌صورت کلی مکانیزمی‌ست که توسعه‌دهندگان برای جلوگیری از آزادی کاربران به‌کار می‌برند. از آن‌جا که در حرکت ذکرشده از کپی‌رایت برای دفاع از آزادی کاربران استفاده شده است و عملاً به‌صورت عکس کار می‌کند، امروزه به این کار کپی‌لفت[۱۵] گفته می‌شود. این مکانیزم استالمن در مجوز GNU GPL (که امروزه مجوز بسیاری از نرم‌افزارهای آزاد است) نمود پیدا کرد.

سال ۱۹۹۱ سالی بود که پروژه‌ی گنو تکمیل شد. در آن سال تمام ابزارهایی که پروژه‌ی گنو لازم داشت آماده شده بود و فقط قسمت هسته باقی مانده بود. در این زمان، لینوس تروالدز - که یک دانش‌جوی علوم کامپیوتر در فنلاند بود - هسته‌ی خودش را تحت عنوان «لینوکس» منتشر کرد. اضافه‌شدن هسته‌ی لینوکس به پروژه‌ی گنو از اتفاقات مهم دنیای آزاد بود و این پروژه با نام جدید گنو لینوکس به‌عنوان یک نرم‌افزار آزاد منتشر شد.

با در نظر گرفتن تمام این فراز و نشیب‌ها، حرف استالمن درباره‌ی سال ۱۹۷۰ به‌خوبی معنا پیدا می‌کند[۱۶]:

من خوش‌شانس بودم که در سال ۱۹۷۰ عضو جامعه‌ای از برنامه‌نویسان بودم که نرم‌افزارها را به‌اشتراک می‌گذاشتند. آن موقع به‌اشتراک‌گذاری نرم‌افزار در جوامع برنامه‌نویسی زیاد رایج نبود؛ اما ما دیگر حالت افراطی داشتیم، چرا که سیستم‌عامل مورد استفاده در آزمایشگاه‌مان در MIT را خودمان نوشته بودیم و آن را به هر کسی می‌دادیم. هر کسی اجازه داشت که نگاهی به آن بیندازد و یک کپی از آن را بگیرد و هر کاری که خواست با آن بکند. این نرم‌افزارها هیچ کپی‌رایتی نداشتند. همکاری شیوه‌ی زندگی ما بود، و ما در این شیوه‌ی زندگی امنیت داشتیم. ما برایش نجنگیدیم و مجبور نبودیم برایش بجنگیم. چرا که ما اساساً به این شکل زندگی می‌کردیم.

۳. پدیدآمدن مفهوم نرم‌افزار متن‌باز

این‌که چگونه جنبشی با عنوان جنبش نرم‌افزار متن‌باز به‌راه افتاد و عبارت «نرم‌افزار متن‌باز» به مرور زمان در کنار عبارت «نرم‌افزار آزاد» قرار گرفت به سال ۱۹۹۷ و اوایل ۱۹۹۸ برمی‌گردد. در آن زمان، تازه‌کارهایی که قصد ورود به فرهنگ نرم‌افزارهای آزاد را داشتند، گمان می‌کردند منظور از کلمه‌ی «آزاد»، رایگان‌بودن است و نه داشتن آزادی. در آن موقع، کریستین پترسن[۱۷] در جمع افرادی از جمله اریک ریموند[۱۸] اولین بار از عبارت نرم‌افزار «متن‌باز» استفاده کرد و ظاهراً مورد تأیید جمع قرار گرفت. (هر چند که ریموند بعدها گفت که این عبارت خیلی مورد تأیید او نبوده است.)

در ابتدا، عبارت «متن‌باز» تنها جهت شفاف‌سازی کلمه‌ی «free» استفاده می‌شد، اما به مرور زمان، جنبش متن‌باز دارای فلسفه‌ای متفاوت با فلسفه‌ی جنبش نرم‌افزار آزاد شد. ریچارد استالمن در مقاله‌ای[۱۹] به این موضوع می‌پردازد. بهترین تعریف برای بیان تفاوت فلسفی این دو جنبش این است که: «متن‌باز» یک روش توسعه‌ی نرم‌افزار است، حال

۹ - http://www.gnu.org/gnu/gnu.html
۱۰ - Recursive
۱۱ - Kernel
۱۲ - هسته مهم‌ترین قسمت یک سیستم‌عامل است که وظیفه‌ی مدیریت منابع سیستم، ارتباط بین نرم‌افزارها و سخت‌افزارها، و... را دارد.
۱۳ - GNU Emacs
۱۴ - X Window System
۱۵ - http://gnu.org/philosophy/pragmatic.html
۱۶ - http://gnu.org/events/rms-nyu-2001-transcript.txt
۱۷ - Christine Peterson
۱۸ - http://en.wikipedia.org/wiki/Eric_S._Raymond
۱۹ - http://www.gnu.org/philosophy/open-source-misses-the-point.html

> The explanation for free software is simple --a person who has grasped the idea of free speech, not free beer will not get it wrong again

آن‌که نرم‌افزار آزاد یک جنبش اجتماعی، یک ضرورت اخلاقی و احترام به آزادی کاربر است. فلسفه‌ی متن‌باز بر این اساس است که چگونه یک نرم‌افزارِ بهتر، از جهت فنی، توسعه یابد.

در ابتدا، تفاوت‌های غیرفلسفی بین یک نرم‌افزار آزاد و یک نرم‌افزار متن‌باز وجود نداشت اما به مرور زمان، در نتیجه‌ی متفاوت‌بودن دو فلسفه، دارای تفاوت‌هایی از بعد مجوز نیز شدند.[20] از بعد فنی، تمام نرم‌افزارهای آزادْ متن‌باز نیز هستند. همچنین، اکثر نرم‌افزارهای متن‌بازْ آزاد نیز هستند اما استثناهایی نیز وجود دارد. به‌عنوان مثال، Open Watcom یک نرم‌افزار غیرآزاد ولی متن‌باز است چرا که به کاربر اجازه نمی‌دهد که نرم‌افزار را مطابق میلش ویرایش کند و به‌صورت خصوصی استفاده کند.

جای دیگری که تفاوت عملی این دو عبارت به‌خوبی مشخص می‌شود در دستگاه‌های ستمگر[21] می‌باشد. نرم‌افزار ستمگر نرم‌افزاری‌ست که هر چند دارای مجوزی از نوع کپی‌لفت مانند GPL است، اما از محدودیت سخت‌افزاری استفاده می‌کند تا کاربر را محدود کند که از نسخه‌ی ویرایش‌شده‌ی نرم‌افزار روی یک سخت‌افزار خاص استفاده نکند. بسیاری از دستگاه‌های اندرویدی ستمگرند. عبارت ستمگر توسط بنیاد نرم‌افزار آزاد[22] برای سخت‌افزارهای tivoized[23] ابداع شد. ماجرای عبارت tivoized به دستگاه ضبط تصویر دیجیتال برند TiVo برمی‌گردد که نرم‌افزارِ روی دوربین‌ها را با مجوز یک نرم‌افزار آزاد منتشر کردند، اما سخت‌افزار آن‌ها اجازه‌ی اجراشدن نرم‌افزار ویرایش‌شده را به کاربر نمی‌داد. این کار اغلب با محدودکردن سخت‌افزار به اجراکردن نرم‌افزار با یک مقدارِ کنترلی[24] خاص صورت می‌پذیرد.

۲۰ - www.gnu.org/philosophy/free-open-overlap.en.html
۲۱ - tyrant devices
۲۲ - free software foundation
۲۳ - en.wikipedia.org/wiki/Tivoization
۲۴ - checksum

۴. مدل‌های درآمدزایی نرم‌افزارهای آزاد

به تاریخچه و فلسفه‌ی نرم‌افزارهای آزاد و متن‌باز پرداختیم و بارها تأکید کردیم که آزاد به‌معنای رایگان نیست. حال بهتر است کمی به شیوه‌های درآمدزایی نرم‌افزارهای آزاد بپردازیم. با توجه به این مفهوم، در وهله‌ی اول، شما می‌توانید نرم‌افزار آزاد خود را با قیمت مشخصی بفروشید![25] البته با تعیین قیمت برای نرم‌افزار ممکن است جمعیت کاربران کاهش پیدا کند و پیشرفت نرم‌افزار از طریق بهبودهای فردی (که از نتایج متن‌بازبودن نرم‌افزار آزاد است) آن‌طور که مورد انتظار است حاصل نشود. راه‌های دیگری نیز برای کسب درآمد از طریق نرم‌افزارهای آزاد وجود دارد. یکی از رایج‌ترین راه‌ها فروش سرویس‌های مختلف مرتبط با نرم‌افزار است. این سرویس‌ها می‌توانند شامل توسعه‌ی نرم‌افزار برای یک شرکت خاص، آموزش استفاده از نرم‌افزار، فروش مستندات نرم‌افزار یا مهم‌تر از همه، پشتیبانی باشد. به‌عنوان مثال، شرکت اوراکل با ارائه‌ی پشتیبانی قوی‌تر و نصف قیمت نسبت به RedHat، باعث کاهش ارزش سهام این شرکت در سال ۲۰۰۶ شد.[26]

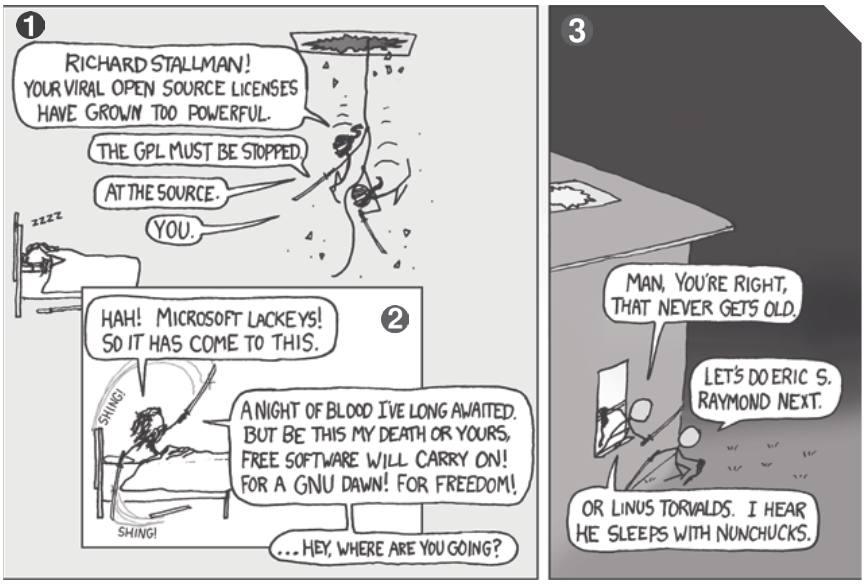
یکی از ویژگی‌های مهم نرم‌افزارهای آزاد نسبت به نرم‌افزارهای غیرآزاد روحیه‌ی جامعه‌ی کاربری آن‌ها است. به‌دلیل احترامی که این نرم‌افزارها به آزادی کاربران می‌گذارند، بسیاری از کاربران حاضر می‌شوند که با کمک‌های مالی خود[27] باعث پیشرفت پروژه شوند. به‌عنوان مثال، پروژه‌ی VLC Player بخش اعظمی از نیاز مالی خود را از طریق کمک‌های مالی مردم تأمین کرد. نرم‌افزار Blender یک نرم‌افزار آزاد است که شیوه‌ی جالبی را برای کسب درآمد اتخاذ کرد. آن‌ها بیان کردند که پروژه متن‌باز خواهد شد در صورتی که حداقل مقدار مشخصی درآمد کسب کنند؛ چه از طریق کمک مالی و چه از طریق فروش.

۲۵ - http://www.gnu.org/philosophy/selling.html
۲۶ - http://www.cnet.com/news/oracle-to-offer-red-hat-linux-support/
۲۷ - Donation

۵. برای مطالعه‌ی بیشتر

در این مقاله، به تاریخ پر فراز و نشیب نرم‌افزارهای آزاد و فلسفه‌ی آن‌ها پرداخته شد. ماهیت آزادی‌خواهی این فلسفه همواره باعث جذب کاربران بسیاری شده است. ویژگی تمام این کاربران وجود صفت آزادی‌خواهی در آن‌ها است. در صورتی که به فلسفه‌ی جنبش نرم‌افزار آزاد علاقه‌مند هستید، مقالات بسیاری در اینترنت در رابطه با آن‌ها وجود دارد که می‌توانید از آن‌ها لذت ببرید. اگر می‌خواهید با مجوزهای نرم‌افزارهای آزاد آشنا شوید، یا در صورتی که یک توسعه‌دهنده هستید و این سؤال برای‌تان مطرح هست که چرا باید یک نرم‌افزار آزاد توسعه داد، می‌توانید به دو مقاله از سایت گنو[28] مراجعه کنید. همچنین مقاله‌ی[29] ریچاد استالمن درباره‌ی این‌که «چرا یک نرم‌افزار باید آزاد باشد» را حتماً مطالعه کنید. به‌عنوان نکته‌ی پایانی، می‌توانید بسیاری از سخنرانی‌ها و مصاحبه‌های بنیاد نرم‌افزارهای آزاد یا پروژه‌ی گنو را در صفحه‌ی سخنرانی‌های گنو[30] بیابید.

۲۸ - http://www.gnu.org/licenses/license-list.html و www.gnu.org/philosophy/fs-motives.html
۲۹ - http://www.gnu.org/philosophy/shouldbefree.html
۳۰ - http://www.gnu.org/philosophy/speeches-and-interview.html



1 هی ریچارد استالمن! مجوزهای کثیف متن‌باز تو زیادی قدرتمند شدن. GPL باید متوقف بشه. از ریشه‌ش. تو.

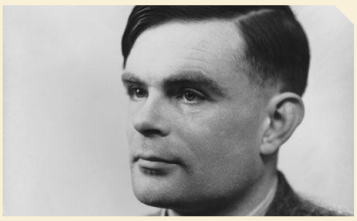
2 ها! نوچه‌های مایکروسافت‌پس بالأخره کار به اینجا کشید. شب خونینی که مدت‌هاست منتظرشم. چه من بمیرم چه شماها، نرم‌افزارهای آزاد ادامه پیدا می‌کنن! برای طلوع گنو! برای آزادی! هی، کجا دارین می‌رین؟

3 راست می‌گی پسر. هیچ‌وقت کهنه نمی‌شه. بیا دفعه‌ی بعد بریم سراغ اریک ریموند. می‌شه سراغ لینوس تورووالدز هم رفت. شنیدم با نانچیکو می‌خوابه.

مروری بر دستاوردهای آلن تورینگ

بعید است کسی در حوزه‌ی دانش کامپیوتر فعالیت یا مطالعه‌ای داشته باشد و نام آلن تورینگ و آزمون وی را نشنیده باشد. بزرگان حوزه‌ی دانش و تکنولوژی عموماً بر این باورند که غول‌های تکنولوژی امروز، از اپل تا مایکروسافت، وام‌دار تورینگ هستند. زمانی که دنیا در فکر ساختن وسایلی بود که ضرب و تقسیم و جمع و تفریق را سریع‌تر انجام دهد، او نوشت: «آیا کامپیوترها در آینده می‌توانند فکر کنند؟ به‌نظرم می‌توان با استفاده از آن‌ها، نورون‌های مغز را شبیه‌سازی کرد و شکلی از کورتکس مغز انسان را توسط آن مدل کرد.» در نوشته‌ی زیر سعی داریم مروری هرچند کوتاه بر تحقیقات انجام‌شده توسط این دانشمند بزرگ داشته باشیم.

فائزه شعبانعلی فمی
f.fami.2005@gmail.com



اعداد رایانش‌پذیر، با کاربردی بر مسئله‌ی تصمیم‌پذیری

هیلبرت دانشمند در آغاز قرن بیستم شرط‌هایی را برای گزاره‌های ریاضی متناهی معتبر تعیین کرده بود: خودسازگاری، تمامیت و تصمیم‌پذیری. گودل نشان داده بود که خودسازگاری و تمامیت به‌طور هم‌زمان قابل دست‌یابی نیستند و اثبات او بر پایه‌ی این فرض بود که هر گزاره‌ای درباره‌ی اعداد را می‌توان با اعداد نمایش داد. همین نوع اندیشه‌ها بود که به یکی از مهم‌ترین آثار تورینگ انجامید. قضیه‌های گودل مسئله‌ی تصمیم‌پذیری را بی‌پاسخ گذاشتند، یعنی این مسئله که آیا اساساً همیشه روشی وجود دارد که بتوان با آن اثبات‌پذیری یک گزاره را تحقیق کرد یا خیر. پاسخ‌دادن به این پرسش نیازمند تعریفی دقیق از «روش» بود و بنابراین در مرز مشترک ریاضی و فلسفه قرار می‌گرفت.

تورینگ پرسش هیلبرت را نیز در همین چهارچوب تازه بازگو می‌کند و به آن پاسخ می‌دهد. مقاله با این پرسش آغاز می‌شود که چگونه می‌توان رشته‌ای نامتناهی (مانند رقم‌های عدد پی) را در قالبی متناهی بیان کرد؟ چگونه می‌توان گفت روش مشخصی برای یافتن آن وجود دارد؟ برای پاسخ‌دادن به این پرسش، تورینگ «کامپیوتر» را به‌شکل یک ماشین تصویر می‌کند: ماشینی با تعداد حالت‌های مشخص (یعنی شماری معلوم از دستورالعمل‌ها) که با یک نوار کاغذی به‌عنوان حافظه کار می‌کند. تورینگ سپس از مفهوم این ماشین (که امروزه ماشین تورینگ خوانده می‌شود) و این‌که مجموعه‌ی عملکردهای پایه‌ای آن متناهی است استفاده می‌کند تا ثابت کند که عددهایی «رایانش‌ناپذیر» وجود دارند: هر عدد رایانش‌پذیری را می‌توان با چنین ماشینی حساب کرد، هرچند شمار رقم‌های آن نامتناهی باشد.

شکستن رمزنگار آلمانی

در جنگ جهانی دوم، آلمانی‌ها برای کدگذاری پیام‌های مورس خود از دستگاهی به نام انیگما استفاده می‌کردند. دستگاه کوچک و قابل حمل و توانمندی که رمزگشایی آن خصوصاً در آخرین مدلش که توسط نیروی دریایی طراحی شده بود عملاً غیرممکن به‌نظر می‌رسید. آلن تورینگ مدت‌ها برای رمزگشایی کدهای این دستگاه تلاش کرد و نهایتاً با استفاده از دستگاه پیچیده‌ای که ساخته بود موفق به رمزگشایی آن شد. وینستون چرچیل گفته است که هیچ فردی به‌تنهایی به اندازه‌ی او در جنگ جهانی دوم تاثیرگذار نبوده است. برآورد می‌شود که کار او باعث شد که این جنگ دو تا چهار سال زودتر به پایان برسد.

موتور محاسبه‌ی خودکار

تورینگ از سال ۱۹۴۵ تا ۱۹۴۷ در آزمایشگاه ملی فیزیک مشغول به کار می‌شود و به طراحی ACE [1] (موتور محاسبه‌ی خودکار) می‌پردازد. در نوشته‌های تورینگ در این دوره می‌توان کم‌کم رگه‌های یک نظریه‌ی هوش مصنوعی را دید: تورینگ این پرسش را مطرح کرد که آیا می‌توان ماشینی ساخت که به‌خوبی شطرنج بازی کند؟ سپس استدلال می‌کند که چنین طرحی ممکن است، به‌شرط آن‌که گاهی امکان خطا را نیز برای ماشین در نظر بگیریم و بپذیریم. او به این موضوع می‌اندیشید که به‌جای شبیه‌سازی ذهن یک انسان بزرگسال، می‌توان ذهن یک کودک را شبیه‌سازی کرد و آن را آموزش داد (ایده‌ای که امروزه در نظریه‌ی شبکه‌های عصبی می‌توان همانندش را یافت). او سپس به بررسی روند آموزش می‌پردازد و آن را به مجموعه‌ای از تشویق‌ها (برای تثبیت یک رفتار) و تنبیه‌ها (برای تغییر آن) تقسیم کرده و روی آن‌ها بحث می‌کند. این گزارش تورینگ برای آزمایشگاه ملی فیزیک در سال ۱۹۴۸ ارائه می‌شود و هیچ واکنش خاصی را برنمی‌انگیزد (این گزارش تا سال ۱۹۶۸ منتشر نشد). تورینگ در این زمان از این نهاد جدا می‌شود.

ماشین تورینگ

ماشین تورینگ دستگاهی است که بر اساس قواعدی که در یک جدول آورده شده، علامت‌های نوشته‌شده روی یک نوار کاغذی را می‌خواند و بر اساس توالی آن‌ها، محاسباتی را انجام می‌دهد. با وجود سادگی، این ماشین می‌تواند منطق هر الگوریتم کامپیوتری را شبیه‌سازی کند و به‌طور خاص، یک مثال خوب در توضیح عملکرد پردازنده‌های مرکزی درون کامپیوترها به‌شمار می‌آید. تورینگ بعدها و در سال ۱۹۴۸ در مقاله‌ای با عنوان سازوکار رایانش و هوشمندی [2] تعریف دقیق‌تری از ماشین محاسبه‌ی خودکار خود ارائه کرد و نام آن را ماشین محاسبات منطقی گذارد. وی در این مقاله چنین می‌گوید: «...یک ظرفیت بی‌نهایت حافظه که به‌صورت نواری کاغذی با طول بی‌نهایت و تقسیم‌شده به مربع‌های مساوی که روی هر کدام امکان نوشتن علامت‌هایی وجود دارد. در هر لحظه، یک علامت در ماشین قرار می‌گیرد که علامت اسکن‌شده نامیده می‌شود. ماشین می‌تواند علامت مذکور را تغییر دهد و در اصل، رفتار آن برگرفته از همان علامت است و دیگر علامت‌های موجود روی نوار نمی‌توانند تأثیری روی عملکرد ماشین بگذارند. نوار کاغذی می‌تواند به عقب و جلو حرکت کند و این یکی از اصول اولیه‌ی کاری ماشین خواهد بود... .»

ماشین پیشگوی تورینگ

در کنار این دو ماشین، تورینگ صحبت از وجود نوع دیگری از ماشین را به‌میان می‌آورد که از آن‌ها به ماشین پیشگو [3] یا O-Machine تعبیر می‌شود. ماشین پیشگو ماشین تورینگی است که می تواند از یک جعبه‌ی سیاه یا «پیشگو» برای تکمیل محاسبات خود و حل مسائل تصمیم‌گیری استفاده کند. این پیشگو که مفهومی نامشخص است و شاید نتوان آن را ماشین به‌شمار آورد می‌تواند مسائل خاصی از هر رده‌ی پیچیدگی (حتی مسائل تصمیم‌ناپذیر) را تنها طی یک مرحله حل کند. این ایده هم‌اکنون به‌شدت مورد توجه و استفاده‌ی ریاضی‌دانان است. یکی از کاربردهای چنین مفاهیمی امور مرتبط با رمزنگاری است که در آن از پیشگوها برای استدلال در زمینه‌ی امنیت پروتکل‌های رمزنگاری‌ای که بر پایه‌ی توابع درهم‌ساز [4] هستند استفاده می‌شود.

در این حالت، از یک پیشگوی تصادفی به‌جای یک تابع درهم‌ساز استفاده می‌شود که به پرس‌وجوهای یکسان پاسخ‌های یکسانی می‌دهد و از این طریق، میزان امنیت پروتکل مربوطه بررسی می‌شود. دسترسی به پیشگوی تصادفی مذکور همانند تابع درهم‌ساز برای تمام استفاده‌کنندگان - حتی مهاجمان - فراهم خواهد بود. چنین کاری نشان می‌دهد که مهاجمان در صورت ناتوانی از حل مسئله‌ی مشکل پایه‌ای که پروتکل بر آن استوار است مجبورند از ویژگی‌های خاص تابع درهم‌ساز استفاده کنند و نمی‌توانند به آن به چشم یک جعبه‌ی سیاه نگاه کنند.

در پایان

تورینگ ۴۲ سال زندگی کرد و در سال ۱۹۵۴ از دنیا رفت. از او به‌عنوان پدر علم کامپیوتر و به‌طور خاص هوش مصنوعی، ریاضیدان، دانشمند علم منطق، رمزنگار و تحلیلگر علوم کامپیوتر، فیلسوف، زیست‌شناس با رهیافت ریاضی، و دونده‌ی دوی ماراتن نام برده‌اند. نبوغ کمتر کسی در قرن گذشته به‌اندازه‌ی نبوغ آلن تورینگ مورد اتفاق نظر همگان است. محققان و فعالان حوزه‌ی IT می‌گویند که تورینگ با هر سال زندگی، شاید می‌توانست دنیای امروز ما را یک دهه جلوتر ببرد. شاید اگر تورینگ حتی یک دهه بیشتر زنده بود، امروز کامپیوتری که در دستان من و شماست شکل دیگری داشت. شاید گوشی‌های هوشمند به‌شکل دیگری با ما حرف می‌زدند. شاید رمزنگاری و رمزگشایی در نقطه‌ی دیگری بود. مجله‌ی تایم او را یکی از صد انسان مهم جهان در قرن بیستم معرفی کرد و نوشت که هر کسی دست به صفحه‌کلید می‌برد یا برنامه‌های تایپ و جدول‌های اعداد و اطلاعات را باز می‌کند، به‌نوعی دستاوردهای فکر او را لمس می‌کند. در پایان نیز می‌توان به یادبودها و نمادهای فراوانی که به‌نام آلن تورینگ بر پا شده است اشاره کرد، که مشهورتر از همه احتمالاً جایزه‌ی تورینگ است که از سال ۱۹۶۶ به‌طور سالانه توسط انجمن ماشین‌های محاسباتی [5] به یک نفر برای دستاوردهای فنی‌اش در زمینه‌ی رایانش داده می‌شود. و به‌اعتقاد بسیاری، بالاترین جایزه در زمینه‌ی رایانش بوده که با جایزه‌ی نوبل قابل مقایسه است.

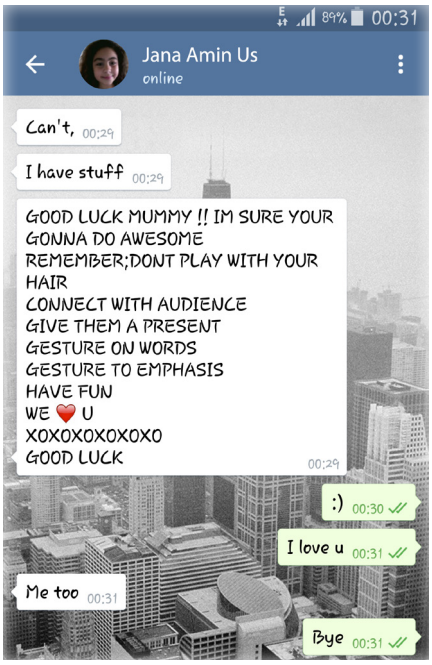
۱ - Automatic Computing Engine
۲ - Computing Machinery and Intelligence
۴ - Hash function
۵ - ACM

TED Ideas worth spreading

این متن برگرفته از یکی از ویدیوهای سایت www.TED.com است. تد (TED) یک سازمان غیرانتفاعی است که با هدف «ترویج ایده‌های ارزشمند» در قالب صحبت‌های کوتاه و پرقدرت ایجاد شده است. تد کار خود را در سال ۱۹۸۴ با یک کنفرانس در زمینه‌های تکنولوژی، سرگرمی و طراحی آغاز کرد و اکنون کار خود را با برگزاری کنفرانس در سرتاسر دنیا در بیش از ۱۰۰ زبان و در قالب موضوعاتی بسیار گسترده‌تر (از علم تا تجارت و مشکلات بین‌المللی) ادامه می‌دهد. از راه‌کارهای رسانه‌ای تد می‌توان به وبگاه آن اشاره کرد، جایی که سخنرانی‌های تد (TEDTalks) به‌صورت ویدیویی، روزانه و رایگان در آن قرار می‌گیرد.

ترجمه و تلخیص: کیمیا کیانی
kimiak94@yahoo.com

احساسات ما روی همه‌ی جنبه‌های زندگی‌مان تأثیر می‌گذارد. از سلامت و چگونگی یادگیری گرفته، تا شیوه‌ی کسب‌وکار و تصمیم‌گیری؛ از ریز تا درشت. احساسات‌مان روی برقراری ارتباط‌مان با دیگری نیز تأثیر می‌گذارد. این پیامک را دخترم شب گذشته برایم فرستاده است، در جهانی که خالی از احساسات است!



من مأموریت دارم که تغییری در وضعیت ایجاد کنم. می‌خواهم احساسات را به تجربه‌ی دیجیتال بیاورم. این راه را ۱۵ سال پیش آغاز کردم. زمانی که یک دانشمند داده در مصر بودم و به‌تازگی در یک برنامه‌ی دکترا در دانشگاه کمبریج پذیرفته شده بودم. من کاری به‌نسبت غیرمعمول برای یک زن مسلمان مصری تازه‌عروس انجام دادم. با پشتیبانی همسرم - که باید در مصر می‌ماند - ساک‌هایم را برداشتم و به انگلیس رفتم. در کمبریج، هزاران مایل دور از خانه، فهمیدم که بیشتر ساعات را با لپ‌تاپم سپری می‌کنم تا با دیگر انسان‌ها. با وجود این صمیمیت، لپ‌تاپ من هیچ ایده‌ای درباره‌ی آن‌چه من احساس می‌کردم نداشت. هیچ ایده‌ای درباره‌ی این‌که من خوشحالم، روز بدی را گذرانده‌ام، استرس داشته‌ام یا گیجم. و خب، این‌ها ناامیدکننده شد. حتی بدتر، هنگامی که با خانواده‌ام به‌صورت آنلاین ارتباط برقرار می‌کردم، احساس می‌کردم که همه‌ی احساسات من در فضای سایبری ناپدید می‌شوند. من دلتنگ خانه بودم، تنها بودم، و بعضی روزها واقعاً گریه می‌کردم. ولی تنها راه ابراز احساساتْ یک شکلک گریان بود.

تکنولوژی امروز بهره‌ی هوشی[۱] بالایی دارد، ولی بهره‌ی احساسی[۲] ندارد. این مرا به فکر فرو برد. اگر تکنولوژی می‌توانست احساسات ما را بفهمد، چه می‌شد؟ چه می‌شد اگر وسایل ما می‌توانستند بفهند چه حسی داریم و با توجه به آن‌ها واکنش نشان دهند، مثل هر یک از دوستان ما که هوش احساسی دارد و واکنش نشان می‌دهد؟ این سؤالات، من و تیمم را به‌سمت ساخت تکنولوژی‌هایی که بتوانند احساسات ما را بخوانند و به آن‌ها پاسخ دهند سوق داد.

نقطه‌ی شروع ما صورت انسان‌ها بود. صورت انسان یکی از قوی‌ترین کانال‌هاست و همه‌ی ما از آن استفاده می‌کنیم تا حالت‌های اجتماعی و احساسی را ابراز کنیم. همه‌چیز، شوق، شگفتی، همدلی و کنجکاوی‌مان.

در دانش احساسات، ما به هر حرکت صورت یک واحد حرکت[۳] می‌گوییم. در نتیجه، برای مثال، حرکتِ واحدِ ۱۲ نام یک فیلم پرفروش هالیوودی نیست، بلکه در حقیقت بالارفتن گوشه‌ی لب است، که جزء اصلی لبخند به‌حساب می‌آید. همه امتحانش کنید، بیایید چندین لبخند داشته باشیم! یک مثال دیگر، حرکتِ واحدِ ۴ است، که چروک‌انداختن ابروست؛ زمانی که شما ابروهایتان را در هم می‌کشید و و به صورت‌تان چین می‌اندازید. ما اخم را دوست نداریم، ولی این کار نشانگری قوی برای احساسات منفی است. ما حدود ۴۵ واحد حرکتی داریم، و همه‌ی آن‌ها با هم ترکیب می‌شوند تا صدها احساس را ابراز و بیان کنند.

آموزش خواندن احساسات صورت به یک کامپیوتر سخت است. زیرا این واحدهای حرکتی می‌توانند سریع، دقیق و ماهرانه باشند. برای مثال، لبخند و پوزخندزدن را درنظر بگیرید؛ آن‌ها از جنبه‌هایی شبیه به هم هستند، ولی معانی کاملاً متفاوتی دارند؛ لبخند مثبت است و پوزخند معمولاً منفی. برای یک کامپیوتر بسیار مهم است که بتواند دو حالت متفاوت را تمییز دهد.

حال، ما این کار را چگونه انجام می‌دهیم؟ ما ده‌ها مثال از صدها مثالی را که مردم از فرهنگ‌ها، سنین و جنسیت‌های متفاوت در آن‌ها می‌خندند به الگوریتم‌هایمان می‌دهیم، و همین کار را برای پوزخندها نیز اجرا می‌کنیم. سپس به‌کمک یادگیری عمیق[۴]، الگوریتم به همه‌ی این الگوها و چین‌وچروک‌ها و تغییرات قالب صورت‌مان نگاه می‌کند و به‌طور اساسی یاد می‌گیرد که همه‌ی لبخندها ویژگی‌های مشابهی دارند، و همه‌ی پوزخندها ویژگی‌های ماهرانه اما متفاوتی دارند. دفعه‌ی بعد که یک صورت را می‌بیند، ضرورتاً به‌یاد می‌آورد که این صورت ویژگی‌های یک لبخند را دارد و می‌گوید: «آها! من این رو تشخیص می‌دم. این یه لبخنده.» پس بهترین راه برای نشان‌دادن چگونگی عملکرد این تکنولوژی اجرای یک نمایش زنده از آن است.

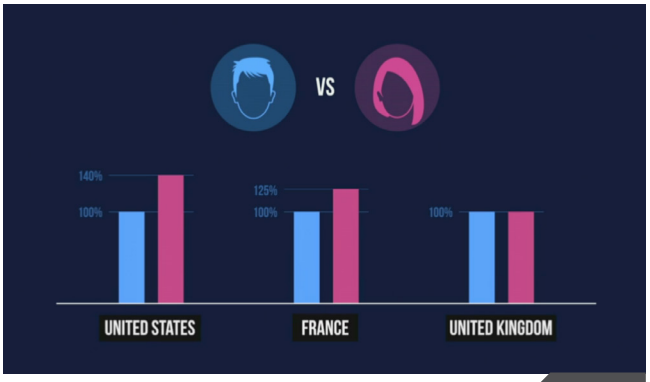
من یک داوطلب می‌خواهم. ترجیحاً فردی با یک صورت! کلویی[۵] داوطلب امروز من خواهد بود. در پنج سال گذشته، ما با تلاش بسیار از یک پروژه‌ی تحقیقاتی در MIT به یک شرکت تبدیل شده‌ایم و تمام تلاش خود را کرده‌ایم تا تکنولوژی‌مان بتواند در فضاهای مختلف به‌خوبی کار کند. و همچنین ما این تکنولوژی را کوچک کرده‌ایم تا هسته‌ی موتور احساسات[۶] روی هر دستگاه موبایل یا دوربین کار کند، مثل این آی‌پد. پس بگذارید این را امتحان کنیم. الگوریتمْ صورت کلویی را پیدا کرده است و کادر سفیدی خصوصیات اصلی صورتش را دنبال می‌کند؛ ابروها، چشم‌ها، بینی و دهانش. سؤال این است که آیا می‌تواند احساساتش را هم شناسایی کند؟ پس ما دستگاه را تست می‌کنیم. اول از همه از کلویی می‌خواهیم

۱ - IQ: Intelligence Quotient
۲ - EQ: Emotional Quotient
۳ - Action unit

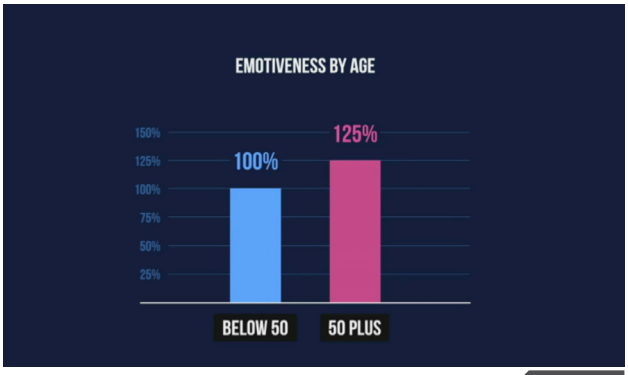
۴ - Deep Learning
۵ - Chloe
۶ - Core motion engine

شکل ۱

شکل ۲



شکل ۳



شکل ۴

صورت بدون احساسش را نشان دهد. نوار سبز با خندیدن او بزرگ می‌شود. این یک خنده‌ی بزرگ بود! کامپیوتر خنده‌ی ملایم‌تر را هم شناسایی می‌کند؛ ما واقعاً سخت کار کرده‌ایم تا تشخیص همه‌ی واحدهای حرکتی مختلف را ممکن کنیم، از جمله بالارفتن ابروها که نشان‌دهنده‌ی شگفتی است، چین‌خوردگی پیشانی نشانگر گیجی است، و اخم‌کردن.

این‌جا تنها یک نمایش کوچک‌شده را مشاهده می‌کنیم. ما به هر کدام از حالات خوانده‌شده یک واحد داده‌ای احساسات[۷] می‌گوییم. این داده‌ها می‌توانند با هم ترکیب شوند تا نشان‌دهنده‌ی احساسات متفاوت باشند. نرم‌افزار نشانگرهایی را برای معیارهای مختلف نمایش می‌دهد. معیار ظرفیت[۸] مثبت یا منفی‌بودن یک تجربه را نشان می‌دهد، و معیار درگیری[۹] بیانگر میزان تأثیرگذاری فرد است.

تا به حال، ما ۱۲ میلیارد از این واحدهای داده‌ای احساسات را گردآوری کرده‌ایم (شکل ۱)، که بزرگ‌ترین پایگاه‌داده‌ی از این نوع در دنیاست. این پایگاه‌داده را از ۲٫۹ میلیون ویدیو از چهره‌های افرادی از ۷۵ کشور جهان جمع‌آوری کرده‌ایم که با دراختیارگذاشتن احساسات‌شان موافقت کرده‌اند. این مجموعه هر روز روبه‌رشد است. این‌که ما می‌توانیم چیزی شخصی مثل احساسات‌مان را اندازه‌گیری کنیم بسیار شگفت‌انگیز است.

تا کنون به چه نتایجی دست یافته‌ایم؟

- **جنسیت:** داده‌های ما چیزهایی را که شما ممکن است در آن‌ها شک داشته باشید تأیید می‌کند. خانم‌ها واضح‌تر از آقایان ابراز احساسات می‌کنند. نه‌تنها بیشتر لبخند می‌زنند، بلکه لبخندهایشان ماندگارتر است (شکل ۲). ما می‌توانیم واقعاً ارزیابی کنیم که زنان و مردان به چه چیزی واکنش‌های متفاوتی نشان‌می‌دهند.
- **فرهنگ:** در ایالات متحده‌ی آمریکا زنان ۴۰٪ بیشتر از مردان در بیان احساسات خود صریح هستند. ولی جالب است که هیچ تفاوتی بین زنان و مردان در انگلستان مشاهده نمی‌شود! (شکل ۳)
- **سن:** کسانی که ۵۰ سال یا بیشتر سن دارند ۲۵٪ بیشتر از افراد جوان‌تر احساساتی هستند (شکل ۴). زنان در دهه‌ی ۲۰سالگی‌شان، بسیار بیشتر از مردانِ با سن مشابه لبخند می‌زنند. اما آن‌چه از همه بیشتر درباره‌ی این داده‌ها شگفت‌زده‌مان کرد، این بود که ما تمام مدت درحال ابراز احساساتیم. حتی زمانی که به‌تنهایی در مقابل دستگاه‌مان نشسته‌ایم، و نه‌تنها زمانی که داریم ویدیوی یک گربه را روی فیس‌بوک می‌بینیم، بلکه همیشه. ما هنگام ای‌میل‌زدن، پیامک‌زدن، خرید آنلاین و حتی هنگام پرداخت مالیات، در حال ابراز احساسات هستیم.

امروزه، این داده‌ها در فهم این‌که ما چگونه با رسانه‌ها در ارتباطیم و بنابراین در فهم الگوهای به‌اشتراک‌گذاری ویدیوها و رأی‌دادن، و همچنین تکنولوژی‌های فعال‌کننده‌ی احساسات[۱۰] یا قدرت‌دهنده استفاده می‌شود. می‌خواهم چند مثال با شما در میان بگذارم که قلبم را درهم می‌فشارند. عینک‌های هوشمند در زمینه‌ی احساسات[۱۱] می‌توانند به افرادی که مشکل بینایی دارند کمک کنند که احساسات دیگران را تشخیص دهند. این تکنولوژی می‌تواند به افراد مبتلا به اوتیسم نیز در قالب مترجم احساسات کمک کند. مشکلی که این افراد به‌طور جدی با آن دست‌وپنجه نرم می‌کنند.

اگر شما در حال یادگیری هستید، اپلیکیشن‌ها سردرگمی شما را احساس می‌کنند و سرعت‌شان کم می‌شود. یا زمانی که حوصله‌تان سر رفته است سرعت‌شان بالا می‌رود، مانند یک معلم عالی در کلاس درس. چه می‌شد اگر ساعت مچی شما احساسات‌تان را درک می‌کرد؟! یا ماشین‌تان می‌فهمید که شما خسته هستید؟! یا شاید یخچال‌تان می‌فهمید که شما مضطرب هستید و به‌طور خودکار بسته می‌شد تا از پرخوری شما جلوگیری کند؟! همه‌ی وسایل ما یک تراشه‌ی احساسات خواهند داشت و ما دیگر زمانی را به‌یاد نخواهیم داشت که نمی‌توانستیم به وسیله‌مان اخم کنیم و وسیله‌مان به ما بگوید: «هممم... این را دوست نداشتی، نه؟».

بزرگ‌ترین چالش ما کاربردهای زیاد این تکنولوژی است. من و تیمم به این نتیجه رسیدیم که نمی‌توانیم همه‌ی این برنامه‌های کاربردی را خودمان بسازیم. پس این تکنولوژی را در دسترس قرار دادیم تا سایر توسعه‌دهندگان بتوانند این برنامه‌های کاربردی را بسازند و خلاق باشند. ما متوجه شدیم که امکان مخاطره و نیز سوءاستفاده وجود دارد. ولی به‌شخصه، بعد از چندین سال انجام این کار، باور دارم که این تکنولوژی مزایایی برای انسانیت دارد. من شما را دعوت می‌کنم که بخشی از این گفتمان شوید. اگر افراد بیشتر در رابطه با این تکنولوژی بدانند، بیشتر می‌توانیم بگوییم که چگونه از آن استفاده می‌شود. همان‌طور که زندگی‌مان بیشتر و بیشتر دیجیتالی می‌شود، در یک جنگ شکست‌خورده مبارزه می‌کنیم تا استفاده از وسایل الکترونیکی را در زندگی‌مان کم کنیم، به این منظور که احساسات‌مان را بازگردانیم. چیزی که من در عوض برایش تلاش‌می‌کنم این است که احساسات را به درون تکنولوژی بیاورم و کاری کنم که تکنولوژی‌هایمان پاسخگوتر باشد و وسایلی که ما را از هم جدا کرده‌اند، دوباره ما را به هم بازگردانند. با انسانی‌کردن تکنولوژی این فرصت را داریم که دوباره درباره‌ی ارتباط‌مان با ماشین‌ها فکر کنیم؛ و در نهایت این‌که ما چگونه به‌عنوان انسان‌ها با یکدیگر ارتباط برقرار می‌کنیم. متشکرم.

۷- Emotion data point
۸- Valence
۹- Engagement
۱۰- Emotion enabling technology
۱۱- Emotion-enabled

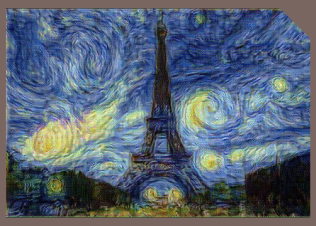
تـــــرنـد فـــصل

حجت مدرسی
Modaresi_hojat@hotmail.com

لوگوی جدید، نام جدید

گستردگی زمینه‌هایی که گوگل در حال کار روی آن‌ها است شاید شگفتی شما را هم برانگیخته باشد؛ از ماشین‌های بدون سرنشین تا تحقیقات روی ژنوم انسان‌ها! این چندان جای شگفتی ندارد، چون لری پیج و سرگی برین هم اخیراً به این نتیجه رسیده‌اند و تصمیم به تبدیل شرکت به چند شرکت کوچک‌تر، هرکدام با مدیرهای مستقل، گرفته‌اند. به همین دلیل، قرار است که از این پس شرکت گوگل به عنوان زیرمجموعه‌ی شرکتی با نام آلفابت به فعالیت خود ادامه دهد.
همچنین پس از گذشت مدت کوتاهی از این اتفاق، گوگل تصمیم گرفت که لوگوی خود را نیز تغییر دهد.
پس از فراگیر شدن طراحی متریال گوگل، این شرکت لوگوی خود را با در نظر گفتن ملاک‌های جدید در زمینه‌ی طراحی، تغییر داد. برای ساخت این لوگوی جدید، افراد و گروه‌های مختلفی از سراسر جهان به مدت یک هفته در نیویورک جمع شدند و طرح‌های مختلفی را ارائه کردند که در نهایت لوگوی فعلی گوگل از میان آن‌ها انتخاب شد.

کامپیوترها به‌سبک ون‌گوگ و پیکاسو نقاشی می‌کشند



محققان آلمانی طی تحقیقات خود به الگوریتمی دست یافته‌اند که می‌تواند سبک نقاشی افراد مشهوری مانند پیکاسو و ون‌گوگ را تقلید کند. این الگوریتم می‌تواند در مدت ۶۰ دقیقه هر عکسی را که به‌عنوان ورودی به آن داده می‌شود به‌سبک این دو نقاش مشهور دربیاورد. برای تبدیل عکس اولیه به نقاشی‌هایی با سبک این نقاش‌ها، این الگوریتم شبکه‌های عصبی کانولوشن ایجاد می‌کند. در واقع روش کاری آن به این صورت است که از الگوهای تشخیص اشیا استفاده کرده و هر شیء را با سبک نقاش مربوطه بازسازی می‌کند. نکته‌ی مهم در مورد این الگوریتم این است که در آن اصل عکس و سبک طرح مجزا از هم هستند و هر کدام به لایه‌های متفاوتی از این شبکه‌ی عصبی کانولوشن مربوط می‌شوند.

ارزیابی فهم عام[۱] سیستم‌های هوش مصنوعی با استفاده از آزمون مدارس[۲]

محققان موسسه‌ی هوش مصنوعی آلن[۳] در حال تلاش بر روی این موضوع هستند که بتوانند به سیستم‌های هوش مصنوعی، فهمی عام از جهان اطراف اضافه کنند. آن‌ها در تحقیقات اخیر خود به این نکته دست یافته‌اند که آزمون‌های مدارس می‌تواند معیار مناسبی برای ارزیابی فهم عام این نوع ماشین‌ها باشد.
برای بررسی این دست‌آورد، آن‌ها یک سیستم هوش مصنوعی با نام آریستو ساخته‌اند و سعی بر این دارند که به آن فهم عام بیاموزند. بهترین عملکرد این برنامه در آزمون‌ها، در پرسش‌های چهارگزینه‌ای است. این سیستم کامپیوتری سعی می‌کند با یافتن رابطه‌های منطقی و استدلال کردن، احتمال درستی هر گزینه را به دست بیاورد و در نهایت، بهترین گزینه را انتخاب کند.

۱ - Common Sense
۲ - http://www.technologyreview.com/news/541001/ai-software-goes-up-against-fourth-graders-on-science-tests/
۳ - Allen Institute for Artificial Intelligence
۴- Aristo

هوش مصنوعی با کامپیوترهای کوانتومی[۱]

گوگل و ناسا به‌همراه سازمان تحقیقات فضایی دانشگاه‌ها[۲] در حال تأسیس آزمایشگاهی برای انجام پردازش‌های هوش مصنوعی به‌کمک کامپیوترهای ناسا هستند. هدف از تشکیل این تیم پیاده‌سازی و آزمایش انواع روش‌های طراحی پردازنده‌های استنتاجی[۳] و بهینه‌سازی از طریق کوانتوم[۴] است. در این تحقیق از Vesuvius D-Wave Two، کامپیوتری qbit 512 که دومین نسل از کامپیوترهای کوانتومی تجاری است، استفاده شده است.

۲ - The Universities Space Research Association
۲ - iTech Post (07/30/2015) Vlad Tverdohleb
۳ - Inference processors
۴ - Quantum optimization

شرکت نرم‌افزاری داتیس آرین قشم

ارائه‌دهنده راهکار جامع بانکی

www.dotin.ir
info@dotin.ir