اگر برنامه‌نویس باشید، حتماً باید با مفاهیمی نظیر "زبان‌های مفسری" و "زبان‌های کامپایلری" آشنا باشید. درک درست از زبان‌های مفسری و کامپایلری، هر دو از مهمترین ابزارهایی هستند که برای توسعه‌ی نرم‌افزارها به کار تان می‌آیند. در این مقاله به بررسی و معرفی زبان‌‌های مفسری و کامپایلری می‌پردازیم و تعریف دقیق این دو نوع زبان را بررسی خواهیم کرد. سپس به توضیح نحوه کار آن‌ها و مزایا و معایب هر کدام می‌پردازیم. در دنیای برنامه نویسی "زبان‌های مفسری و کامپایلری" از اهمیت ویژه‌ای برخوردار هستند، چرا که با توجه به نوع پروژه‌ برنامه‌ نویسی، انتخاب درست بین هر یک از این زبان‌ها، باعث بهینه‌تر شدن کدهای شما می‌شود. با ما همراه باشید تا در این مقاله به بررسی و معرفی زبان‌های مفسری و کامپایلری بپردازیم.

زبان‌‌های مفسری

همانطور که می‌دانید، کامپیوتر درک درستی از زبان‌‌های برنامه نویسی سطح بالا ندارد؛ بنابراین به واسطی نیاز داریم که برنامه را به کد ماشین تبدیل کند. این واسط، همان زبان‌ برنامه نویسی مفسری (interpreter) و زبان‌ برنامه نویسی کامپایلری (compiler) است.

زبان مفسری چیست؟

زبان‌های مفسری یا Interpreted Languages، زبان‌هایی هستند که کدهای برنامه‌ نویسی آن‌ها، به صورت خط به خط توسط یک مفسر (Interpreter) اجرا می‌شود. به این صورت که مفسر، کد برنامه‌ نویسی را دریافت کرده و به ترتیب خطوط آن را تفسیر کرده تا به زبان قابل درک برای کامپیوتر تبدیل کند. به همین دلیل، زبان‌‌های مفسری نیازی به کامپایل کردن کد برنامه‌ نویسی ندارند و در عوض، اجرای کدها به صورتی که در فایل متنی آن‌ها ذخیره شده، صورت می‌گیرد. به عنوان مثال، Python و Ruby نمونه‌هایی از زبان‌های مفسری هستند. این زبان‌ها، برای اجرای کدهای برنامه‌نویسی خود نیازی به کامپایل کردن آن‌ها ندارند و توسط مفسرهای مخصوص به هر زبان، به صورت خط به خط تفسیر می‌شوند. استفاده از زبان‌‌های مفسری، مزایای خاص خود را دارد. به دلیل عدم نیاز به کامپایل کردن کدها، زبان‌‌های مفسری قابلیت توسعه و تست برنامه را افزایش می‌دهند. همچنین، به دلیل اینکه کدها بصورت متنی ذخیره می‌شوند، این زبان‌ها قابلیت خوانایی و نگهداری بالایی دارند. علاوه بر این، با اجرای خط به خط برنامه، می‌توان در هر لحظه از اجرای برنامه، وضعیت آن را بررسی کرد و از این ویژگی برای اشکال‌زدایی و تست برنامه استفاده کرد.

نحوه‌ی کارکرد زبان‌‌های مفسری

عملکرد زبان‌‌های مفسری رو مرحله را شامل می‌شود. در بخش اول، کل برنامه از قبل به زبان مفسر تحویل داده شده و در بخش دوم، برنامه توسط کاربر تایپ و سپس توسط مفسر اجرا می‌شود. در این نوع زبان‌ها، برنامه‌ها به صورت خط به خط خوانده خواهند شد و هر خط به تنهایی توسط مفسر ترجمه و اجرا می‌شود. در هر خط، مفسر ابتدا سینتکس (Syntax) را بررسی می‌کند تا مطمئن شود که خطایی در برنامه وجود ندارد و سپس خط را به صورت کامل اجرا می‌کند. در صورتی که خطایی در خط مورد نؤ وجود داشته باشد، مفسر به صورت خودکار خطا را نمایش می‌دهد. از آنجایی که در این نوع زبان‌ها هیچ فرایند کامپایلی صورت نمی‌گیرد، مفسر برای اجرای برنامه باید در هر بار اجرا به صورت خط به خط آن را تفسیر کند. این ویژگی باعث می‌شود که اجرای برنامه در زبان‌های مفسری به طور کلی کندتر و آهسته‌تر از زبان‌های کامپایلری باشد. همچنین، به دلیل نیاز به مفسر برای اجرای برنامه، مصرف حافظه و پردازش‌های اضافی مورد نیاز است.

مزایا و معایب زبان‌‌های مفسری

همانطور که پیشتر اشاره شد، زبان‌های مفسری برای اجرای کدهای خود از یک مفسر کمک می‌گیرند. برای تشخیص خطا در کد، مفسر ابتدا بررسی‌های لازم را انجام داده و پس از آن، کد را اجرا می‌کند. در ادامه به مزایا و معایب استفاده از زبان‌های مفسری پرداخته می‌شود:

مزایا زبان‌های مفسری

سهولت در نوشتن کد: در زبان‌های مفسری برای نوشتن کد نیاز به دانش کامپیوتری عمیقی ندارید؛ برای اجرای کد، فقط کافی است که یک مفسر مناسب برای آن زبان نصب شده باشد.

اجرای سریع کد: در زبان‌‌های مفسری، کد بلافاصله پس از تایپ اجرا خواهد شد. به همین دلیل، زمان اجرای کد در زبان‌‌های مفسری از زبان‌‌های کامپایلری کمتر است.

امکانات خوب دیباگینگ: در زبان‌های مفسری، به دلیل اجرای دستورات در لحظه و بلادرنگ (real-time)، می‌توان از امکانات دیباگینگ به صورت کارآمدتری استفاده کرد.

معایب زبان‌های مفسری

کندی در اجرای کد: زبان‌‌های مفسری در زمان اجرا به دلیل ترجمه خط به خط و اجرای کد مربوط به هر خط، با کندی مواجه می‌شوند. این مشکل به‌ویژه در برنامه‌هایی با حجم بالا و دستورات پیچیده برجسته‌تر بوده که اجرای آن‌ها باعث کاهش سرعت عملکرد می‌شود.

کد منبع قابل دیدن: در زبان‌‌های مفسری، کد منبع در هنگام اجرا قابل مشاهده است و می‌توان آن را به سادگی مورد حمله قرار داد.

نیاز به نصب مفسر: برای اجرای برنامه‌های نوشته شده با زبان‌‌های مفسری، نیاز است که مفسر آن زبان بر روی سیستم کاربر نصب شده باشد. این مسئله به عنوان یک نقطه ضعف در مورد استفاده از زبان‌‌های مفسری در نظر گرفته می‌شود.

امکان ایجاد خطا: به دلیل اجرای کد در لحظه و به دلیل اینکه کد‌های نوشته شده با زبان‌های مفسری به صورت خط به خط ترجمه و اجرا می‌شوند، در صورتی که کد نوشته شده به درستی کار نکند، احتمال ایجاد خطا افزایش می‌یابد.

مثال‌ هایی از زبان‌‌های مفسری

تعدادی از زبان‌‌های مفسری محبوب و پراستفاده را در زیر بررسی می‌کنیم:

پایتون (Python)

پایتون یکی از محبوب‌ترین زبان‌های مفسری در دنیا است که توسط Guido van Rossum در دهه ۱۹۸۰ توسعه یافت. این زبان در بسیاری از حوزه‌های مختلف مانند علم داده، ماشین لرنینگ، وب دولوپمنت و حتی بازی‌ سازی مورد استفاده قرار می‌گیرد. پایتون به دلیل سینتکس ساده و خوانا، انعطاف‌پذیری بالا، مجموعه کاملی از کتابخانه‌ها و فریم‌ورک‌های مختلف و جوامع بزرگ و پویا، به شدت محبوب است.

جاوااسکریپت (JavaScript)

زبان جاوا اسکریپت یا به اختصار "JS" یک زبان مفسری و بسیار رایج در بین توسعه دهندگان وب است. این زبان در داخل مرورگر وب اجرا می‌شود و به طور گسترده‌ای برای ایجاد وبسایت‌های پویا و بدون نیاز به بارگذاری مجدد صفحه، مورد استفاده قرار می‌گیرد. برای اجرای جاوا اسکریپت، نیازی به نصب یک مفسر مخصوص یا کامپایلر ندارید و می‌توانید کدهای JS را مستقیماً در برنامه‌های وب خود استفاده کنید.

روبی (Ruby)

روبی یک زبان برنامه‌نویسی شیءگرا و مفسری بوده که توسط یوکیهیرو ماتسوموتو (Yukihiro Matsumoto) توسعه داده شده است. روبی برای توسعه برنامه‌های وب، برنامه‌های سرور و نرم‌افزارهای دسکتاپ به کار می‌رود و به دلیل ساختار کد خوانا و قابل فهم، محبوبیت زیادی دارد.

پرل (Perl)

پرل یک زبان برنامه‌ نویسی مفسری است که برای پردازش متن و توسعه برنامه‌های موجود در سیستم عامل‌های یونیکس به کار می‌رود. پرل برای توسعه برنامه‌های وب و ابزارهای سرور نیز کاربرد دارد.

پی‌ اچ‌ پی (PHP)

پی‌ اچ‌ پی یک زبان برنامه‌ نویسی مفسری است که بیشتر برای توسعه برنامه‌های وب استفاده می‌شود. با استفاده از پی‌ اچ‌ پی، می‌توان صفحات وب پویا، فرم‌های تعاملی، سیستم‌های مدیریت محتوا و بسیاری از برنامه‌های وب دیگر را ایجاد کرد. PHP به سادگی با بسیاری از پایگاه‌ داده‌های رایج مانند MySQL و PostgreSQL ارتباط برقرار می‌کند و به عنوان یکی از بهترین زبان‌‌های برنامه‌ نویسی وب شناخته می‌شود. برخیدیگر از معروف‌ترین زبان‌های مفسری شامل بش (Bash)، پاورشل (PowerShell)، لوآ (Lua)، سوییفت (Swift) و Tcl هستند. انواع زبان‌های مفسری زبان‌‌های کامپایلری روش اجرا کدها، مهمترین وجه تمایز بین زبان‌های مفسری و زبان‌های کامپایلری است. برای درک این تفاوت، تعریف زبان کامپایلری و ابعاد مختلف آن می‌پردازیم: زبان کامپایلری چیست؟ زبان کامپایلری یا Compiled Language، زبانی است که کد منبع آن توسط یک کامپایلر به کد اجرایی ترجمه می‌شود. در زبان کامپایلری، کامپایلر با استفاده از الگوریتم‌های خاص خود، کد منبع را به صورت کامل به کد اجرایی تبدیل می‌کند. در ادامه این کد اجرایی می‌تواند در سیستم عامل وارد شده و اجرا شود. در زبان کامپایلری، قبل از اجرای برنامه، کامپایلر کد منبع را به کد ماشینی (machine code) ترجمه خواهد کرد. کد ماشینی برای پردازنده‌ها قابل فهم بوده و می‌تواند به صورت مستقیم اجرا شود. این کد اجرایی نیاز به نصب رانتایم (runtime) ندارد و عملکرد بهینه‌تری نسبت به زبان‌های مفسری دارد. با این حال، تغییرات در کد منبع نیازمند کامپایل مجدد و تولید مجدد کد اجرایی هستند.

نحوه‌ی کارکرد زبان‌‌های کامپایلری

نحوه کارکرد زبان‌‌های کامپایلری به این صورت است که کد منبع نوشته شده توسط برنامه‌نویس، به عنوان ورودی به کامپایلر داده می‌شود. در ادامه کامپایلر کد منبع را دریافت و بررسی کرده تا به زبان سطح پایین (Low-level programming language) و معمولاً به زبان ماشین، تبدیل کند. در این فرآیند، کامپایلر قوانین برنامه نویسی و خطاهای مرتبط با آن را زیر نظر می‌گیرد و در صورتی که خطایی وجود دارد، پیام خطای مناسب را نمایش می‌دهد. پس از تبدیل کد به زبان سطح پایین، این کد قابل اجرا توسط سیستم می‌شود.

مزایا و معایب زبان‌‌های کامپایلری

زبان‌‌های کامپایلری، علاوه بر دارا بودن مزایایی نظیر سرعت بالا و بهینه‌سازی کدها، با برخی نفاط ضعف همراه هستند که باید در نظر گرفته شوند. در این بخش، به بررسی مزایا و معایب زبان‌‌های کامپایلری پرداخته خواهد شد:

مزایا زبان‌های کامپایلری

سرعت اجرای برنامه

: زبان‌‌های کامپایلری به دلیل تبدیل کل برنامه به کد ماشینی در فرآیند کامپایل، از سرعت اجرای برنامه بسیار بالاتری نسبت به زبان‌های مفسری برخوردار هستند.

بهینه‌سازی برنامه

: در زبان‌‌های کامپایلری، کامپایلر به صورت خودکار قابلیت بهینه‌سازی کد را داراست، بنابراین کدی که توسط کامپایلر تولید می‌شود، بهینه‌تر بوده و سرعت اجرای برنامه را بیشتر می‌کند.

کشف خطاهای زمان کامپایل

: در زبان‌‌های کامپایلری، خطاهای برنامه در فرآیند کامپایل شناسایی و به کاربر اعلام می‌شود. به عبارت دیگر، با استفاده از زبان کامپایلری می‌توان خطاهای برنامه را در مرحله توسعه شناسایی کرد و آن‌ها را برطرف نمود.

پایداری بالا

: به دلیل ذخیره کد کامپایل شده به صورت فایل اجرایی، برنامه‌ها با پایداری بالاتری اجرا می‌شوند. این به این معناست که تغییرات در نرم افزار باعث بروز خطاهای بیشتر نمی‌شود

معایب زبان‌های کامپایلری

سختی در اجرای برنامه در پلتفرم‌های مختلف

: برای اجرای برنامه‌های کامپایل شده بر روی سیستم‌های مختلف، باید برنامه را برای هر سیستم، مجزا کامپایل کرد. این موضوع باعث می‌شود که اجرای برنامه بر روی پلتفرم‌های مختلف سخت باشد.

زمان کامپایل بیشتر

: زبان‌‌های کامپایلری بر خلاف زبان‌‌های مفسری، زمان بیشتری برای کامپایل برنامه‌ها نیاز دارند. این موضوع ممکن است باعث تاخیر در اجرای برنامه‌های بزرگ و پیچیده شود.

عدم قابلیت اجرای برنامه در حین کامپایل

: در زبان‌های کامپایلری، برنامه باید در ابتدا کامپایل شده و سپس اجرا شود. این موضوع باعث عدم امکان اجرای برنامه در حین کامپایل می‌شود که در برخی مواقع ایجاد مشکل می‌کند.

کد نویسی برای برنامه‌نویسان

: کد نویسی برای برنامه‌نویسان برنامه‌های کامپایل شده، ممکن است سخت‌تر و پیچیده‌تر باشد. همچنین این موضوع باعث افزایش هزینه‌ها و زمان توسعه برنامه خواهد شد.

برخی مشکلات امنیتی

: به دلیل آنکه برنامه‌های کامپایل شده به صورت کد باینری و اجرایی در دسترس هستند، ممکن است برخی مشکلات امنیتی وجود داشته باشد که در برنامه‌های مفسری به راحتی قابل حل هستند.

مثال‌ هایی از زبان‌‌های کامپایلری

تعدادی از زبان‌‌های کامپایلری محبوب و پراستفاده را در زیر بررسی می‌کنیم:

زبان برنامه نویسی سی (C)

زبان برنامه‌ نویسی سی یکی از معروف‌ترین زبان‌های کامپایلری است. این زبان در سال ۱۹۷۰ توسط دنیس ریچی برای توسعه سیستم‌عامل یونیکس طراحی شد و به دلیل سرعت و کارایی بالای خود، در توسعه نرم‌افزار و سیستم‌های عامل، همچنین قابلیت انتقال به سادگی بین پلتفرم‌های مختلف، به عنوان یکی از محبوب‌ترین زبان‌های برنامه‌نویسی شناخته شده است.

جاوا (Java)

جاوا یک زبان برنامه‌نویسی شی‌گرا و کامپایلری است که توسط جیمز گاسلینگ در سال ۱۹۹۵ طراحی شد. این زبان با استفاده از ماشین مجازی جاوا (JVM) که برای تفسیر کد جاوا طراحی شده، قابلیت اجرای برنامه‌ها را در پلتفرم‌های مختلف فراهم می‌کند.

سی پلاس پلاس ( ++ C)

یکی از مثال‌های دیگر زبان‌‌های کامپایلری، ++ C است. ++C یک زبان برنامه‌ نویسی شیءگرا و سطح بالا است که توسط Bjarne Stroustrup در دهه ۱۹۸۰ به عنوان یک توسعه از زبان C ایجاد شد. ++C توسط بسیاری از برنامه‌نویسان برای توسعه نرم‌افزارهای سیستمی، بازی‌های رایانه‌ای، نرم‌افزارهای گرافیکی و بسیاری دیگر از برنامه‌های کامپیوتری استفاده می‌شود. ++ C از روش کامپایلری برای تبدیل کد منبع به کد اجرایی استفاده می‌کند که در این فرایند، کامپایلر کد منبع را بررسی کرده و آن را به کد اجرایی ترجمه می‌کند.

پایتون (Python)

در واقع، پایتون یک زبان برنامه‌نویسی مفسری است، اما در برخی از پیاده‌سازی‌های آن می‌توان ترکیبی از مفسر و کامپایلر استفاده کرد. برای مثال، در پیاده‌سازی Cpython ابتدا کد پایتون توسط مفسر خوانده می‌شود و به بایت کد تبدیل می‌شود، سپس این بایت کد توسط ماشین مجازی پایتون (Python Virtual Machine) اجرا خواهد شد. در نتیجه، پایتون هم مفسری و هم کامپایلری به شمار می‌آید. برخی دیگر از معروف‌ترین زبان‌های کامپایلری شامل Swift، Rust، Go، Pascal، Fortran، Ada، و Assembly هستند.

مقایسه زبان‌‌های مفسری و کامپایلری

شباهت‌ها و تفاوت‌های زبان‌های مفسری و کامپایلری بسیار مهم و قابل توجه هستند. این دو دسته زبان برخی ویژگی‌های مشابهی دارند، اما در عین حال در بسیاری از مشخصات با یکدیگر تفاوت دارند. در این بخش، به مقایسه و توضیح شباهت‌ها و تفاوت‌های زبان‌های مفسری و زبان‌های کامپایلری می‌پردازیم:

شباهت‌ها

هر دو نوع زبان به کاربر اجازه می‌دهند تا برنامه‌ی خود را در یک زبان خاص بنویسند و آن را به زبانی قابل فهم برای کامپیوتر ترجمه کنند.

هر دو نوع زبان از ساختارهای کنترلی مشابهی برخوردار هستند که شامل حلقه‌ها، شرطی‌ها و توابع است.

در هر دو نوع زبان، برنامه‌های نوشته شده می‌توانند بر روی سیستم‌عامل‌های مختلف اجرا شوند.

هر دو نوع زبان برای تولید برنامه‌هایی با کارایی بالا و سرعت اجرای بالا طراحی شده‌اند.

هر دو نوع زبان می‌توانند برای ایجاد برنامه‌های بزرگ و پیچیده استفاده شوند.

هر دو نوع زبان در صورت نوشتن کد درست و بهینه، می‌توانند به نتایج مشابهی در اجرای برنامه‌ها دست پیدا کنند.

تفاوت‌ها

روش اجرا: زبان‌های مفسری کد را در زمان اجرا تفسیر و خط به خط آن را اجرا می‌کنند. در حالی که زبان‌های کامپایلری کد را در زمان کامپایل ترجمه و خروجی کامپایل شده را اجرا می‌کنند.

سرعت اجرا: بدون شک، زبان‌‌های کامپایلری در سرعت اجرای برنامه‌ها بهتر هستند. چرا که کامپایلر، کد را به صورت کامل به زبان ماشین ترجمه کرده و خروجی آن بهینه و سریع است. در مقابل، زبان‌‌های مفسری برای هر دستوری که باید اجرا شود، باید کدها را تفسیر و سپس آن را اجرا کنند که باعث کاهش سرعت اجرای برنامه می‌شود.

اشکال‌زدایی: اشکال‌زدایی برای زبان‌های کامپایلری ممکن است کمی سخت باشد، زیرا خطاهایی که در زمان کامپایل رخ می‌دهند، باید توسط کدگذاری برطرف شوند. در مقابل، زبان‌‌های مفسری به دلیل تفسیر دستورات، می‌توانند اشکال‌زدایی را راحت‌تر پیش ببرند.

پرتاب‌های استثناء (exception handling): برخلاف زبان‌‌های کامپایلری، زبان‌‌های مفسری به صورت پیش فرض پرتاب استثناء دارند. این به معنای این است که در صورت بروز خطا در هنگام اجرای برنامه، برنامه متوقف و خطایی به کاربر نمایش داده می‌شود.

جمع بندی

در این مقاله ابتدا به تعریف زبان‌‌های مفسری و کامپایلری پرداختیم و نحوه کارکرد هر کدام را بررسی کردیم. سپس به مزایا و معایب هر دو نوع زبان را بررسی و نمونه‌هایی از زبان‌‌های مفسری و کامپایلری را معرفی کردیم. در نهایت، برای انتخاب زبان مناسب برای پروژه‌های خود، باید به نیاز‌ها و شرایط پروژه توجه کرد. به عنوان مثال، انتخاب زبان برنامه‌ نویسی باید بر اساس نیازهای پروژه و محیط کاری باشد و هیچ زبانی به تنهایی برای تمام پروژه‌ها مناسب نخواهد بود. همچنین، در برخی موارد ممکن است ترکیب زبان‌‌های مفسری و کامپایلری عملکرد و کارایی بهتری داشته باشد. می‌توان گفت که استفاده از زبان‌‌های کامپایلری و مفسری به عنوان یک ابزار قدرتمند، در دسترس برنامه‌نویسان بوده و به آن‌ها اجازه می‌دهد کارهای پیچیده و زمان‌بر را بهتر و سریع‌تر به نتیجه برسانند.

مزایا و معایب هوش مصنوعی چیست؟

مزایای هوش مصنوعی

مزایای هوش مصنوعی را به‌صورت کلی می‌توان از این قرار دانست:

1. کاهش خطای انسانی

یکی از بزرگ‌ترین مزیت‌های هوش مصنوعی این است که می‌تواند به‌میزان چشمگیری خطاها را کاهش و دقت را افزایش دهد. تصمیماتی که هوش مصنوعی در هر مرحله اتخاذ می‌کند با توجه به اطلاعات جمع‌آوری‌شده‌ی قبلی و مجموعه‌ی خاصی از الگوریتم‌ها تعیین می‌شود. درصورتی‌که این موضوع به‌درستی برنامه‌ریزی شود، این خطاها را حتی می‌توان به صفر کاهش داد.

1. حذف ریسک

یکی دیگر از مزیت‌های بزرگ هوش مصنوعی این است که انسان‌ها می‌توانند با استفاده از ربات‌های هوش مصنوعی بر بسیاری از خطرات غلبه کنند. خواه خنثی‌کردن بمب باشد، خواه رفتن به فضا و کاوش در عمیق‌ترین بخش‌های اقیانوس‌ها، ماشین‌هایی با بدنه‌های فلزی مقاوم هستند و می‌توانند در جوهای غیرطبیعی زنده بمانند؛ علاوه‌براین، آن‌ها می‌توانند با مسئولیت بیشتر کار دقیقی ارائه کنند و به‌راحتی فرسوده نمی‌شوند.

1. هفت روز هفته ۲۴ ساعته در دسترس

مطالعات زیادی وجود دارد که نشان می‌دهد انسان‌ها تنها سه تا چهار ساعت در روز بازدهی دارند. همچنین انسان‌ها برای ایجاد تعادل میان زندگی کاری و شخصی خود به استراحت نیاز دارند؛ اما هوش مصنوعی می‌تواند بدون وقفه، بی‌پایان کار کند. آن‌ها خیلی سریع‌تر از انسان‌ها فکر می‌کنند و چندین کار را هم‌زمان با نتایج دقیق انجام می‌دهند. آن‌ها حتی می‌توانند کارهای تکراری خسته‌کننده را به‌راحتی با کمک الگوریتم‌های هوش مصنوعی برای مدت نامتناهی انجام دهند.

1. کمک دیجیتال

تقریباً همه‌ی سازمان‌های بزرگ این روزها از دستیارهای دیجیتال برای تعامل با مشتریان خود استفاده می‌کنند که به‌طور چشمگیری نیاز به منابع انسانی را به حداقل می‌رساند. شما به‌راحتی می‌توانید با چت‌بات‌ها چت کنید و از آن‌ها بپرسید دقیقاً به چه چیزی نیاز دارید. برخی از ربات‌های چت این روزها به‌قدری هوشمند شده‌اند که نمی‌توانید تشخیص دهید آیا با یک چت‌بات چت می‌کنید یا یک انسان.

1. اختراع‌های جدید

هوش مصنوعی تقریباً در هر حوزه‌ای به اختراع‌های جدید برای حل مشکلات پیچیده کمک کرده است؛ برای مثال، این اختراع‌های اخیر به پزشکان کمک کرده است تا با استفاده از فناوری‌های پیشرفته مبتنی بر هوش مصنوعی، مرحله‌های اولیه‌ی سرطان سینه را در زنان پیش‌بینی کنند.

1. تصمیمات بی‌طرفانه

انسان‌ها، خواه و ناخواه، با احساسات هدایت می‌شوند. هوش مصنوعی عاری از احساسات و در رویکرد خود بسیار کاربردی و منطقی است. مزیت بزرگ هوش مصنوعی این است که هیچ دیدگاه جانبدارانه‌ای ندارد که این موضوع تصمیم‌گیری دقیق‌تری را تضمین می‌کند.

معایب هوش مصنوعی

معمولاً همیشه حرف از مزایای هوش مصنوعی و کمکی است که می‌تواند به بشریت کند، اما بعد از بررسی مزایای هوش‌مصنوعی بهتر است با معایب این حوزه هم آشنا شویم تا درک‌مان از این حوزه کامل‌تر شود.

1. هزینه‌های بالا

توانایی ایجاد ماشینی که بتواند هوش انسان را شبیه‌سازی کند، کار کوچکی نیست. این کار به زمان و منابع زیادی نیاز دارد و می‌تواند هزینه‌ی زیادی را در بر داشته باشد. همچنین هوش مصنوعی باید روی جدیدترین سخت‌افزار و نرم‌افزارها کار کند تا به‌روز بماند و آخرین نیازها را برآورده کند؛ این موضوع درنتیجه آن را بسیار پرهزینه می‌کند.

1. نداشتن خلاقیت

یک عیب بزرگ هوش مصنوعی این است که نمی‌تواند یاد بگیرد خارج از چارچوب یا همان Out of Box فکر کند. هوش مصنوعی قادر است در طول زمان با داده‌هایی که از قبل دریافت کرده است و تجربیات گذشته‌ی خود یاد بگیرد، اما نمی‌تواند در رویکرد خود خلاق باشد. یک مثال کلاسیک، ربات Quill است که می‌تواند گزارش‌های درآمد فوربز (Forbes) را بنویسد. این گزارش‌ها فقط حاوی داده‌ها و حقایقی هستند که قبلاً به ربات ارائه شده است. اگرچه این موضوع چشمگیر است که یک ربات می‌تواند به‌تنهایی مقاله بنویسد، اما این مقالات از حس انسانی موجود در دیگر مقالات فوربز بی‌بهره است.

1. افزایش بیکاری

شاید یکی از بزرگ‌ترین معایب هوش مصنوعی این باشد که کم‌کم تعدادی از کارهای تکراری را با ربات‌ها جایگزین می‌کند. کاهش نیاز به دخالت انسان به مرگ بسیاری از فرصت‌های شغلی انجامیده است. یک مثال ساده چت‌بات است که یک مزیت بزرگ برای سازمان‌هاست، اما یک کابوس برای کارمندان است. مطالعه‌ی Mckinsey پیش‌بینی می‌کند که هوش مصنوعی تا سال ۲۰۳۰ جایگزین حداقل ۳۰ درصد نیروی کار انسانی خواهد شد.

1. انسان‌ها را تنبل می‌کند!

برنامه‌های کاربردی هوش مصنوعی اکثر کارهای خسته‌کننده و تکراری را خودکار می‌کنند؛ بنابراین از آنجا که برای انجام‌دادن کارها مجبور نیستیم چیزها را به خاطر بسپاریم یا معماها را حل کنیم، تمایل پیدا می‌کنیم کمتر و کمتر از مغزمان استفاده کنیم. این اعتیاد به هوش مصنوعی می‌تواند برای نسل‌های آینده مشکلاتی ایجاد کند.

1. حذف اخلاق

اخلاق ویژگی مهم انسانی است که گنجاندن آن در هوش مصنوعی دشوار است. پیشرفت سریع هوش مصنوعی نگرانی‌هایی را ایجاد کرده است که روزی هوش مصنوعی به‌طور غیرقابل‌کنترلی رشد خواهد کرد و درنهایت بشریت را از میان خواهد برد. از این لحظه به‌عنوان تَکینِگی هوش مصنوعی یا AI Singularity یاد می‌شود.

**منابع کتاب‌ها، مقالات گوگل و دانش قبلی**