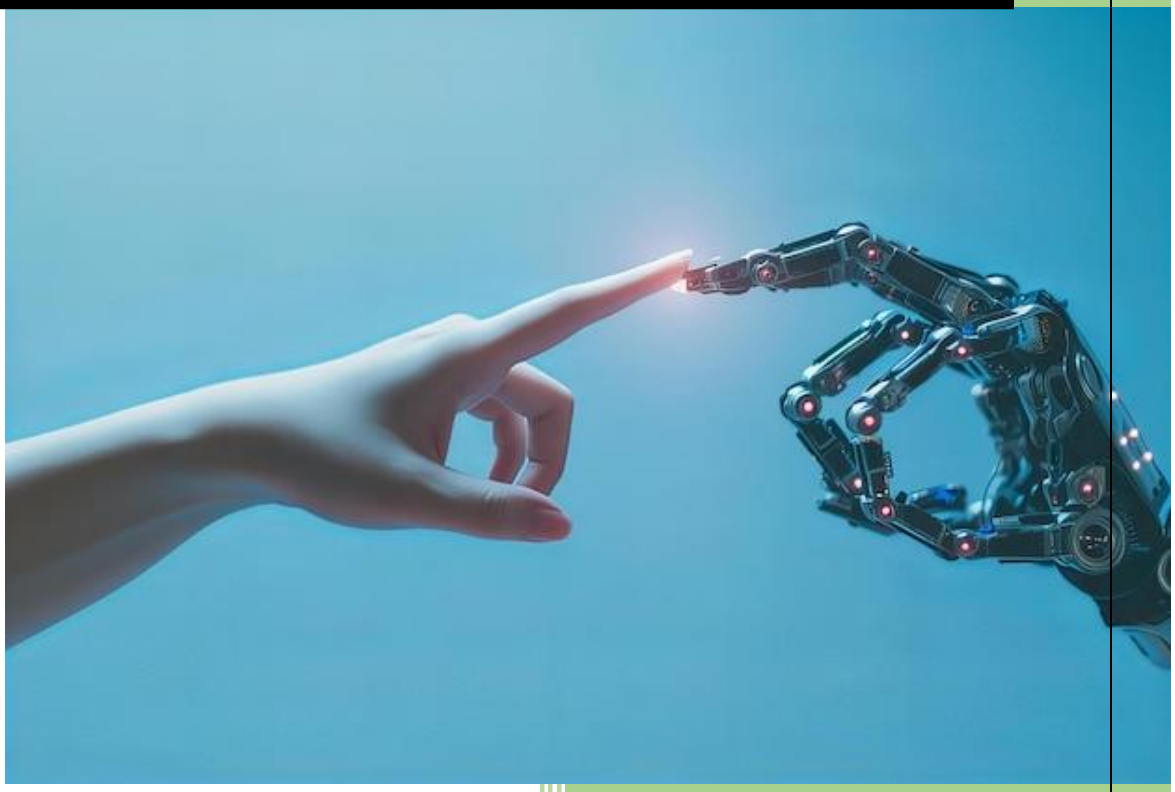
مروری بر ساختار و عملکرد زبان یوز



مقدمه

در دنیای امروز، پردازش زبان طبیعی (NLP) و تعامل بین انسان و ماشین‌ها به یکی از حوزه‌های حیاتی در علم کامپیوتر و هوش مصنوعی تبدیل شده است. با گسترش استفاده از چت‌بات‌ها، دستیارهای هوشمند و سیستم‌های پاسخ‌گویی خودکار، نیاز به زبان‌های تخصصی و ابزارهایی که به بهبود و تسهیل پردازش متون و تعاملات زبانی کمک کنند، بیش از پیش احساس می‌شود.

زبان یوز به عنوان یک زبان پردازش متنی نوین، با هدف ساده‌سازی و بهینه‌سازی فرایند تحلیل و پاسخ‌گویی به متون طراحی شده است. این زبان با تکیه بر اصول و مفاهیم ساده، اما قدرتمند، به کاربران اجازه می‌دهد تا با نوشتن کدها و الگوهای خاص، پاسخ‌هایی دقیق و متناسب با ورودی‌های متنی دریافت کنند. طراحی این زبان به گونه‌ای بوده است که نه تنها برای برنامه‌نویسان حرفه‌ای، بلکه برای کسانی که به تازگی وارد دنیای برنامه‌نویسی شده‌اند، قابل فهم و استفاده باشد.

در این کتاب، هدف ما معرفی کامل زبان یوز، از تاریخچه و انگیزه‌های پشت طراحی آن گرفته تا نحوه استفاده و پیاده‌سازی در پروژه‌های واقعی است. با ارائه مثال‌های عملی و توضیحات جامع، سعی داریم تا خوانندگان را با قدرت‌ها و قابلیت‌های این زبان آشنا کرده و آنها را برای استفاده مؤثر از آن در پروژه‌های خود آماده کنیم.

زبان یوز، با ساختاری انعطاف‌پذیر و امکانات گسترده‌ای که در اختیار کاربران قرار می‌دهد، می‌تواند به عنوان یک ابزار قدرتمند در دنیای پردازش زبان طبیعی به کار گرفته شود. ما امیدواریم که این کتاب بتواند راهنمایی مفید برای شما باشد و انگیزه‌ای برای توسعه پروژه های قدرتمند با زبان یوز ایجاد کند.

اصطلاحات رایج هوش مصنوعی

+ الگوریتم

الگوریتم در برنامه نویسی و هوش مصنوعی مجموعه‌ای از دستورات است که کامپیوتر می‌تواند آن‌ها را برای درک نحوه اجرا و تکمیل یک وظیفه دنبال کند و انجام دهد.

+ چت بات

چت‌بات یک سیستم هوش مصنوعی است که با کاربر از طریق کانال‌های صوتی یا متنی به تعامل می‌پردازد تا کاربران را در خصوص کارها راهنمایی کند.

+ یادگیری ماشین

یادگیری ماشین شاخه‌ای از هوش مصنوعی است که بر استفاده از الگوریتم‌ها و داده‌ها برای تقلید از نحوه یادگیری انسان‌ها تمرکز دارد. این به ماشین‌ها امکان می‌دهد تا بتوانند به خودشان آموزش بدهند و وظایف را به گونه‌ای موثرتر و بدون دخالت انسان‌ها از طریق به‌کارگیری الگوها و استنتاج‌ها اجرا کنند.

+ پردازش زبان طبیعی

پردازش زبان طبیعی که به اختصار آن را NLP خطاب می‌کنند، اصطلاحی فراگیر است که در خصوص قابلیت کامپیوترها برای اجرای کارکردها و عملکردهای مربوط به صحبت کردن و مکالمه مطرح می‌شود.

+ بیگ دیتا

بیگ دیتا به مجموعه بزرگی از داده ها گفته می شود که حجم و اطلاعات بسیار زیادی دارد .

+ طبقه بندی داده

در بحث یادگیری ماشین ، طبقه بندی یعنی یک مسئله مدل‌سازی که در آن مدل‌های یادگیری ماشین برچسب دسته را برای داده‌های ورودی پیش‌بینی می‌کند.

+ داده کاوی

داده کاوی به پیدا کردن الگو در داده‌ها برای پیش‌بینی خروجی گفته می‌شود. به بیان دیگر، داده کاوی فرایند و پردازشی است که در آن داده‌های خام به اطلاعات مفیدی تبدیل می‌شوند و می‌توان از آن اطلاعات برای انجام اقدامات لازم استفاده کرد.

فصل ۱

معرفی زبان یوز

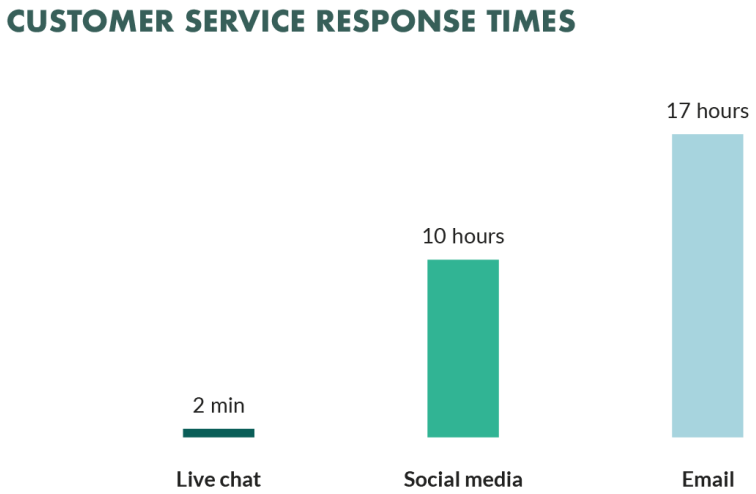
۱. زبان یوز چیست ؟

یوز یک زبان نشانه گذاری راست چین فارسی است که برای توسعه چت‌بات‌ها و سیستم‌های مکالمه‌ای طراحی شده است . یوز این امکان را به برنامه نویسان می دهد که با استفاده از ساختار هایی ساده و قابل فهم الگو هایی برای پاسخگویی ربات ها طراحی کنند .

یوز با نیاز سنجی های انجام شده کاملا نوآور بوجود آمده و از ویژگی هایی رونمایی می کند که تاکنون مشابه آن عرضه نشده است اما از سیستم زبان های منسوخ قدیمی نیز در ساختار خود استفاده کرده است . زبان هایی مانند chat script , AIML , RiveScript که سالها در صدر توجهات قرار داشتند اما به مرور با آمدن کتابخانه ها و پردازش های قدرتمند رنگ باختند لیکن یوز ایده های توسعه دهندگان آنها که حاصل سالها زحمات بود را گلچین و گردآوری کرده است .

۲. جایگاه یوز در کسب و کار های دیجیتال

در عصر دیجیتال میلیون ها فروشگاه اینترنتی و شرکت های خدماتی در بستر وب فعال هستند اما نمی توانند پاسخ مشتریان را به موقع بدهند و برای خدمات پاسخگویی بهتر مجبور به استخدام نیروی انسانی هستند . طبق نظرسنجی American Express در سال ۲۰۲۳ ، ۶۱ درصد مشتریان می گویند که اگر مجبور باشند بیش از ۵ دقیقه منتظر بمانند احتمالا خرید خود را لغو کنند و یا در نظر سنجی انجام شده توسط SuperCRM مدت زمان انتظار برای پاسخگویی فروشنده در شبکه های اجتماعی ۱۰ ساعت و پاسخ بصورت ایمیل ۱۷ ساعت است .



همه اینها باعث شده است که Gartnerدر گزارش خود در سال ۲۰۲۳ پیش بینی کند که ۷۰ درصد از تعاملات خدمات مشتری تا سال های آینده به فناوری های نوظهور مانند چت بات ها و دستیاران مجازی منتقل خواهد شد.

زبان یوز با ساختار ساده و منعطفی که دارد این امکان را برای کسب و کار ها ایجاد می کند که یک ربات همیشه آنلاین پاسخگو برای خود ایجاد کنند .

۳. سرعت پردازش

زبان یوز با ۲,۷۱ میلی ثانیه سرعت پردازش در داده های ۱ پارامتری (کتابخانه js) بیشترین سرعت را در مقایسه با سایر زبان های مشابه دارد :

Ai - language	Time (ms)
NLTK (Python)	۱۰.۲۳۴
AIML	۷.۴۵۶
Rive Script	۵.۱۲۳
Chat Script	۳.۷۸۹
yooz	۲.۷۱۰

فصل ۲

دستورات زبان یوز

ساختار یوز

در زبان یوز هر الگو درون پرانتز قرار میگیرد . علامت + نشانگر الگویی از سوی کاربر است و علامت – نشانگر الگوی پاسخی از سوی ربات :

```
(
   + <user pattern>
   - < response>
)
```

برای مثال داریم :

```
)
+ حالت چطور است؟
- من خوبم، مرسی که پرسیدی
(
```

یوز در الگو های دریافتی از سوی کاربر از قواعد ریجکس هم پیروی می کند . در ادامه به برخی از قوانین ریجکس می پردازیم :

مهارها - ^ و $

- **^The** – با هر رشته‌ای که با The آغاز شود تطبیق می‌یابد.
- **end$** – با هر رشته‌ای که به کلمه end خاتمه یابد تطبیق می‌یابد.
- **^The end$** – تطبیق دقیق رشته‌ای را تعریف می‌کند، یعنی رشته مورد جستجو باید با The آغاز و با end خاتمه یابد.
- **roar** – با هر رشته‌ای که کلمه roar در آن باشد تطبیق می‌یابد.

سورها – * + ? و {}

- **abc*** – با هر رشته‌ای که در ابتدایش کاراکترهای ab و در ادامه صفر یا چند c داشته باشد تطبیق می‌یابد.
- **abc+** – با هر رشته‌ای که در ابتدایش کاراکترهای ab و در ادامه یک یا چند کاراکتر c داشته باشد تطبیق می‌یابد.
- **abc?** – با هر رشته‌ای که در ابتدایش کاراکترهای ab و در ادامه صفر یا یک کاراکتر c داشته باشد تطبیق می‌یابد.
- **abc{2}** – با هر رشته‌ای که در ابتدایش کاراکترهای ab و در ادامه دقیقاً 2 کاراکتر c داشته باشد تطبیق می‌یابد.
- **abc{2,}** – با هر رشته‌ای که در ابتدایش کاراکترهای ab و در ادامه 2 یا بیشتر کاراکتر c داشته باشد تطبیق می‌یابد.
- **abc{2,5}** – با هر رشته‌ای که در ابتدایش کاراکترهای ab و در ادامه 2 تا 5 کاراکتر c داشته باشد تطبیق می‌یابد.
- **a(bc)*** – با هر رشته‌ای که در ابتدایش کاراکتر a و در ادامه صفر یا چند کپی از دنباله bc داشته باشد تطبیق می‌یابد.
- **a(bc){2,5}** – با هر رشته‌ای که در ابتدایش کاراکتر a و در ادامه 2 تا 5 کپی از دنباله bc داشته باشد تطبیق می‌یابد.

دسته‌های کاراکتر - \s \w \d و .

- **\d** – با یک کاراکتر منفرد که رقم باشد تطبیق می‌یابد.
- **\w** – با یک کاراکتر کلمه (کاراکتر حرفی/عددی به علاوه زیرخط) تطبیق می‌یابد.
- **\s** – با کاراکتر خالی تطبیق می‌یابد (شامل tab و line break نیز می‌شود).
- **.** – با هر کاراکتری تطبیق می‌یابد.

باید از عملگر (.) با احتیاط استفاده کنید، چون در اغلب موارد کلاس یا کلاس کاراکتر منفی آن (که در ادامه معرفی می‌کنیم) سریع‌تر و بسیار دقیق‌تر است. حالت نفی \w، \d و \s به ترتیب \W، \D و \S هستند. برای نمونه \D تطبیق معکوس را با توجه به آن چیزی که با \d به دست می‌آید ارائه می‌کند.

- **\D** – با یک کاراکتر غیر رقمی منفرد تطبیق می‌یابد.

برای این که به صورت عملی از آن استفاده کنید، باید کاراکترها را با استفاده از \ به صورت escape درآورید چون معنای خاصی دارند.

- **\$\d** – با رشته‌ای تطبیق می‌یابد که یک $ پیش از یک رقم دارد.

توجه کنید که می‌توانید کاراکترهای غیر قابل چاپ مانند \t، خطوط جدید \n، بازگشت‌های carriage یعنی \r را نیز تطبیق دهید.

عملگر OR - | یا []

- **a(b|c)** – با هر رشته‌ای که یک کاراکتر a و در ادامه b یا c داشته باشد تطبیق می‌یابد.
- **a[bc]** – همانند regex قبلی است.

فلگ‌ها

تا به این جا در مورد شیوه ساخت یک regex مطالبی آموختیم، اما یک مفهوم بنیادی به نام فلگ را فراموش کرده‌ایم.

Regex معمولاً به صورت /abc/ ارائه می‌شود که الگوی جستجو به وسیله دو کاراکتر اسلش متمایز شده است. در انتهای عبارت منظم می‌توان یک فلگ با مقادیر زیر تعیین کرد (امکان ترکیب کردن آن‌ها با هم نیز وجود دارد):

- **g (سراسری یا global)** – پس از نخستین تطبیق بازگشت نمی‌یابد و جستجوهای بعدی را از انتهای مورد مطابقت قبلی آغاز می‌کند.
- **m (چندخطی یا multi-line)** – زمانی که ^ و $ به جای کل رشته با ابتدا و انتهای یک خط مطابقت داشته باشند.
- **i (غیر حساس یا insensitive)** – موجب می‌شود که کل عبارت منظم از حالت حساس به حروف کوچک یا بزرگ خارج شود. برای نمونه /aBc/i می‌تواند با AbC تطبیق یابد.

گروه‌بندی و capture – ()

- **a(bc)** – پرانتزها یک گروه تشکیل می‌دهند که مقدار bc را به دست می‌دهد.
- **a(?:bc)*** – ما با استفاده از : ? گروه capturing را غیر فعال می‌کنیم.
- **a(?<foo>bc)** – با استفاده از ?<foo> یک نام برای گروه خود تعیین می‌کنیم.

این عملگر در مواردی که لازم است اطلاعات از رشته‌ها یا داده‌ها با استفاده از زبان برنامه‌نویسی خاصی استخراج شود کاملاً مفید خواهند بود. هر رخداد چندگانه به وسیله چند گروه به دست می‌آید و در یک آرایه کلاسیک عرضه می‌شود، یعنی می‌توان با استفاده از یک اندیس روی نتیجه تطبیق به هر رخداد مجزا دسترسی داشت.

عبارت‌های براکت – []

- **[abc]** – با هر رشته‌ای که یک a یا ab یا a c داشته باشد، تطبیق می‌یابد و معادل a|b|c است.
- **[a-c]** – همانند قبلی است.
- **[a-fA-F0-9]** – با رشته‌ای تطبیق می‌یابد که نماینده یک رقم هگزادسیمال منفرد است و حساسیت به حروف کوچک یا بزرگ ندارد.
- **%[0-9]** – با رشته‌ای تطبیق می‌یابد که پیش از علامت % یک کاراکتر از 0 تا 9 دارد.
- **[a-zA-Z^]** – با رشته‌ای تطبیق می‌یابد که حرفی از a تا z یا از A تا Z ندارد. در این حالت ^ به عنوان منفی عبارت استفاده می‌شود.

به خاطر داشته باشید که درون عبارت‌های براکتی، همه کاراکترهای خاص (شامل بک اسلش \) قدرت خاص خود را از دست می‌دهند. بدین ترتیب قاعده scape قابل استفاده نیست.

- دستور ستاره (*)

یکی از قابلیت‌های کلیدی زبان یوز، استفاده از پارامتر ستاره (*) است که به کاربران امکان می‌دهد بخش‌هایی از ورودی کاربر را به صورت متغیر در الگوهای خود تعریف کرده و از آن‌ها در پاسخ‌ها بهره‌برداری کنند. این قابلیت به ایجاد مکالمات پویا و انعطاف‌پذیر کمک شایانی می‌کند و امکان پاسخ‌گویی به ورودی‌های متنوع و پیچیده را فراهم می‌سازد.

دستور ستاره (*) به عنوان یک جای‌نگهدار عمل می‌کند که می‌تواند با هر بخشی از متن جایگزین شود. این دستور برای زمانی کاربرد دارد که بخواهید بخشی از ورودی کاربر را بدون نیاز به تعیین دقیق آن در الگو شناسایی کنید و سپس از همان بخش در پاسخ استفاده نمایید. برای مثال عبارت " من دیروز با ماشین به مشهد رفتم" جزء الگوی "من دیروز * به مشهد رفتم" محسوب می شود . از طرفی هر عبارتی درون * باشد در خود کاراکتر * ذخیره می گردد و قابلیت استفاده دارد . برای مثال در الگوی "اسم من * است" و پاسخ الگو " سلام *" باشد اگر کاربر عبارت "اسم من امیرحسین است" را وارد کند ربات عبارت "سلام امیرحسین" را برمیگرداند .

این امکان را وجود دارد که همزمان چند متغییر خالی * تعریف کرد و از همه ظرفیت های آن بهره برد . برای مثال داریم :

```
)
+ اسم من *۱ است .
- خوشبختم *۱ عزیز .
(
```

- پاسخ تصادفی :

گاهی برنامه نویس از ربات میخواهد که به ازای دریافت پارامتری جواب های مختلف و تصادفی بدهد برای این منظور در بین عبارات مد نظر از علامت ‘–‘ استفاده می شود .

پاسخ تصادفی به زبان یوز این امکان را می‌دهد که به ازای یک الگوی مشخص از ورودی کاربر، چندین پاسخ مختلف تعریف شود و بات به صورت تصادفی یکی از آن‌ها را انتخاب و ارائه کند. این ویژگی موجب می‌شود که مکالمات جذاب‌تر و غیرقابل پیش‌بینی‌تر باشند و حس تعامل انسانی بیشتری به کاربران القا شود و از پاسخ‌های تکراری و یکنواخت اجتناب کنند.

ساختار پاسخ تصادفی در یوز بدین شکل است :

```
(
   + <user pattern>
   - <response1> – <response2> - ... - < response>
)
```

برای مثال در نظر بگیرید که می‌خواهید برای سوال "حالت چطور است؟" چندین پاسخ مختلف داشته باشید. با استفاده از دستور پاسخ تصادفی، می‌توانید الگو را به شکل زیر تعریف کنید:

```
)
+ حالت چطور است؟
- من خوبم، مرسی که پرسیدی. _ من عالی‌ام، تو چطوری؟ _ خیلی خوبم، امیدوارم تو هم خوب باشی.
(
```

- خزش بیشتر برای جوابی کامل تر :

در زبان یوز، قابلیت "خزش بیشتر برای جوابی کامل‌تر" به بات این امکان را می‌دهد که پس از یافتن پاسخ اولیه، جستجوی خود را برای یافتن پاسخ‌های اضافی ادامه دهد. این ویژگی به خصوص در سناریوهایی مفید است که نیاز به ارائه اطلاعات تکمیلی یا ایجاد مکالمات پیچیده‌تر وجود دارد. در این بخش به توضیح کامل این قابلیت همراه با مثال می‌پردازیم.

برای پیاده‌سازی این قابلیت، از نشانه !> در انتهای پاسخ‌ها استفاده می‌شود. این نشانه به بات دستور می‌دهد که به جستجوی الگوهای اضافی ادامه دهد. برای مثال داریم :

```
#ایران : ایران کشوری با جمعیت بیش از ۸۰ میلیون نفر است .

)

+ ایران کجاست ؟

- کشوری زیبا در غرب آسیا !>

(

)

+ ایران کجاست ؟

- #ایران

(
```

اجرای این کد به شکل زیر است :

- دسته بندی داده ها :

یوز این قابلیت را دارد که برای فهم بیشتر جملات و کلمات داده ها در یوز داده ها بصورت طبقه بندی شده گردآوری شود . برای مثال برای فهم بیشتر ربات از همه ضمایر دسته ای بنام ضمایر در یوز باز می کنیم و ضمایر را داخل آن قرار می دهیم :

```
افعال {است، بود، شد، گشت، گردید، رفت، آمد}

ضمایر {من، تو، او، ما، شما، ایشان، آنها}
```

اکنون میخوایم یک الگو تعریف کنیم برای مثال که اگر از دسته کلمات ضمایر استفاده شد سپس از دسته کلمات فعل ربات متوجه شود که شخصی فعلی را انجام داد ! در تعریف الگو برای دسته ها از کاراکتر "&" استفاده میکنیم .

برای مثال در کد زیر داریم :

```
افعال {است، بود، شد، گشت، گردید، رفت، آمد}

ضمایر {من، تو، او، ما، شما، ایشان، آنها}

)

+ &ضمایر

- شما از ضمیر استفاده کردید.

(
```

- حذف کلمات قبل از پردازش :

بعضی کلمات مزاحم وجود دارد که پردازش ربات را به دردسر می اندازد ! اصطلاحا به این لغات در فرهنگ کامپیوتر stop words می گویند . برای افزایش راندمان در پردازش برنامه نویسان می توانند بعضی کلمات را از متن کاربر حذف کنند و سپس متن توسط ربات بررسی بشود . این کلمات می‌توانند شامل حروف اضافه، افعال کمکی، و ضمایر باشند. حذف این کلمات به بات اجازه می‌دهد تا بر روی کلمات و عبارات مهم‌تر تمرکز کند. برای اینکار در زبان یوز شما کافیست از الگوی – {} استفاده کنید . برای مثال ما میخواهیم حروف اضافه درون متن کاربر را حذف کنیم تا پردازش نشوند ! در اینصورت داریم :

```
- {از ، به ، با ، در ، برای }
```

اهمیت حذف کلمات توقف

کاهش حجم داده: حذف کلمات غیرضروری باعث کاهش حجم داده‌های ورودی و در نتیجه بهبود کارایی الگوریتم‌های پردازش زبان طبیعی می‌شود.

بهبود دقت: با حذف کلمات توقف، الگوریتم‌ها می‌توانند بر روی کلمات کلیدی و مهم متمرکز شوند که به درک بهتر متن کمک می‌کند.

افزایش سرعت: پردازش متون کوتاه‌تر و متمرکزتر به زمان کمتری نیاز دارد که منجر به پاسخ‌دهی سریع‌تر بات می‌شود.

- بخاطر سپردن اطلاعات :

یکی از ویژگی‌های مهم و جذاب زبان یوز توانایی به خاطر سپردن اطلاعات است که به بات‌ها اجازه می‌دهد مکالمات شخصی‌تر و پویاتری با کاربران داشته باشند. این قابلیت می‌تواند تعاملات طبیعی‌تری را فراهم کرده و بات را قادر به ایجاد پاسخ‌های بر اساس اطلاعات قبلی کاربران نماید.

بخاطر سپردن اطلاعات به معنای ذخیره موقت داده‌هایی است که در طول مکالمه با کاربر به دست می‌آید. این داده‌ها می‌توانند شامل اطلاعات شخصی کاربر، تنظیمات مورد علاقه، تاریخچه مکالمات، و سایر جزئیات باشند که در پاسخ‌گویی بهتر و دقیق‌تر به کاربر مورد استفاده قرار می‌گیرند.

برای فراخوانی اطلاعاتی که در بحث ربات یاد می گیرد از علامت مساوی استفاده می شود :

```
(
+ من نمیتوانم بخوابم.
- آیا شما بیمار شده اید ؟
=موضوع : خواب
)
(
+ درباره چه چیزی صحبت می کردیم ؟
- ما درباره =موضوع صحبت می کردیم
)
```

- پردازش های تو در تو :

برای مثال شما ربات خود را بطوری آموزش دادید که در مواجه با موضوعی سوالی از کاربر بپرسد ! اما کاربر وقتی جواب میدهد از نتایج کل دیتا برای پاسخ نگردد و صرفا از دیتا های آن بخش برای پاسخگویی استفاده کند ! اینجاست که قابلیت پردازش تو در تو می آید !

```
(
+ من نمیتوانم بخوابم.
- آیا شما بیمار شده اید ؟
=موضوع : خواب
(
+ بله
- متاسفم
)
(
+ خیر
- از علائم بسیاری از بیماری ها اختلال در خواب است .
)
)
```

- آموزش تعاریف کلی :

در زبان یوز، یکی از قابلیت‌های قدرتمند، امکان تعریف و استفاده از تعاریف کلی با استفاده از علامت # است. این قابلیت به شما اجازه می‌دهد تا مفاهیم، عبارات یا اطلاعات مشترک را یک‌بار تعریف کرده و در نقاط مختلف کد خود از آن‌ها استفاده مجدد کنید. این کار باعث افزایش خوانایی، نگهداری و قابلیت توسعه کد شما می‌شود.

تعاریف کلی به توسعه‌دهندگان این امکان را می‌دهند که مفاهیم یا عبارات رایج را با یک شناسه منحصر به فرد مشخص کرده و در سایر قسمت‌های کد از آن‌ها استفاده کنند. این روش نه تنها از تکرار غیرضروری جلوگیری می‌کند، بلکه امکان به‌روزرسانی آسان‌تر اطلاعات را نیز فراهم می‌آورد.

ساختار کلی برای تعریف و استفاده از تعاریف کلی به صورت زیر است:

```
#نام موضوع : تعریف موضوع .
```

برای مثال داریم :

```
#عراق : کشوری در غرب آسیا.
(
+ عراق کجاست؟
- #عراق
)
(
+ درباره عراق بگو
- #عراق
)
```

- درک متن با کلید واژه ها :

در زبان‌های پردازش متن، استفاده از کلیدواژه‌ها برای تحلیل و پاسخ به پیام‌های کاربر یکی از روش‌های مؤثر برای درک معنای اصلی متن است. یکی از تکنیک‌های پرکاربرد در این زمینه، استفاده از TF-IDF (Term Frequency-Inverse Document Frequency) است. این روش به زبان یوز کمک می‌کند تا کلیدواژه‌های مهم را در پیام‌های کاربر شناسایی کرده و پاسخ‌های مناسب را ارائه دهد.

مفهوم TF-IDF

TF-IDF یکی از تکنیک‌های اصلی در استخراج اطلاعات و درک معنای متون است. این روش بر پایه دو معیار مهم استوار است:

TF فرکانس واژه :این معیار نشان‌دهنده تعداد دفعات تکرار یک واژه در یک سند است. هرچه یک واژه در سند بیشتر تکرار شود، اهمیت آن واژه در آن سند بیشتر می‌شود.

IDF معکوس فرکانس سند :این معیار نشان‌دهنده اهمیت یک واژه در میان همه اسناد موجود است. واژه‌هایی که در تعداد کمی از اسناد ظاهر می‌شوند، اهمیت بیشتری دارند، در حالی که واژه‌هایی که در اکثر اسناد تکرار می‌شوند (مثل کلمات عمومی)، اهمیت کمتری دارند.

محاسبه TF-IDF با ضرب TF در IDF برای هر واژه در یک سند انجام می‌شود. نتیجه این محاسبه، میزان اهمیت نسبی هر واژه در سند مورد نظر را نشان می‌دهد.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

کاربرد TF-IDF در زبان یوز

در زبان یوز، TF-IDF به ما کمک می‌کند تا کلیدواژه‌های مهم و مرتبط با پیام کاربر را شناسایی کرده و به‌درستی پاسخ دهیم. این روش به‌ویژه زمانی که متن ورودی کاربر پیچیده‌تر است و شامل اطلاعات متنوعی می‌شود، اهمیت بیشتری پیدا می‌کند.

نحوه عملکرد در یوز

استخراج کلیدواژه‌ها: ابتدا پیام ورودی کاربر به کلمات تجزیه شده و تعداد تکرار هر واژه (TF) در پیام محاسبه می‌شود.

محاسبه اهمیت کلیدواژه‌ها: با توجه به مجموعه داده‌های موجود (مثلاً مکالمات یا اطلاعات قبلی)، میزان اهمیت هر واژه با استفاده از IDF محاسبه می‌شود.

اولویت‌بندی کلیدواژه‌ها: کلیدواژه‌هایی که TF-IDF بالاتری دارند، به عنوان کلیدواژه‌های مهم‌تر در نظر گرفته می‌شوند.

انتخاب پاسخ مناسب: بر اساس کلیدواژه‌های شناسایی‌شده، الگوریتم یوز پاسخ مناسبی را از میان پاسخ‌های ممکن انتخاب و ارائه می‌کند.

برای مثال :

```
)
+ { کشور ، ایران }
- #ایران
(
```

فصل ۳

دستورات در یک نگاه

اعضا {اشکان ، متین}

#عراق : کشوری در غرب آسیا.

} -از، به، با، در{

)

+ اسم شما چیست ؟ .

- اسم من یوز است.

)

)

+ حالت چطور است ؟

- من خوبم، مرسی که پرسیدی.

)

)

+ چطوری ؟.

- ممنون _ متشکرم _ تو چطوری ؟.

)

)

+ عراق کجاست ؟

- عراق

)

)

+ اسم من *۱ است.

- خوشبختم *۱ عزیز.

)

)

+ من هر سال *۱ بار شهر *۲ سفر می کنم.

- آیا هر *۱ بار از سفر خود لذت میبرید به شهر *۲ ؟.

)

)

} + عراق، کشور{

- عراق

)

)

+ کمی درباره خودت بگو.

- من خوبم !>

)

(

+ کمی درباره خودت بگو.

- تشکر

)

(

+ من &اعضا هستم

- شما از اعضای این باشگاه هستید

)

(

+ خوبی ؟

- ممنون!

= موضوع : احوالپرسی

)

(

+ در مورد چه چیزی صحبت می کردیم ؟

- ما در مورد =موضوع صحبت میکردیم

)

فصل ۴

اجرا دستورات یوز

کد های یوز :

بخش کد های یوز ، بخشی است که برنامه نویس کد های نوشته شده خود را در آنجا قرار می دهد .

تست کد یوز :

با چت باکس تست کد یوز می توانید کد های نوشته شده خود را امتحان کنید !

تمامی پاسخ های داده شده توسط ربات بر اساس داده های نوشته شده شما در بخش کد های یوز است .

کدی که پردازش شده :

در هر مکالمه کدی که باعث جواب مربوطه شد در این بخش نمایش داده می شود ! این باعث می شود برنامه نویس بدانند چه بخش از کد باعث جواب شده و ایراد یابی آسانتر می شود .