عنوان زیرپروژه:

استخراج ابعاد نیازمندی‌های تبدیل متن به گفتار در ایجاد پیکره‌های متناظر زبان فارسی

فهرست مطالب

عنوان — **شماره صفحه**

مقدمه ____ **۱۱**

فصل۱

معرفی سیستم‌های تبدیل متن به گفتار ____ **۱۳**

۱-۱. مقدمه ____ **۱۴**

۱-۲. تولید گفتار سطح بالا ____ **۱۶**

۱-۲-۱. پردازش متن ____ ۱۷

۱-۲-۲. تلفظ ____ ۱۸

۱-۲-۳. پروزودی ____ ۱۸

۱-۳. تولید گفتار سطح پایین یا تولید شکل موج گفتار ____ **۱۹**

۱-۳-۱. تولید گفتار مفصلی ____ ۱۹

۱-۳-۲. تولید گفتار فرمانت ____ ۲۱

۱-۳-۳. تولید گفتار اتصالی ____ ۲۴

۱-۴. نتیجه‌گیری ____ **۲۶**

فصل۲

سیستم‌های تشخیص و تبدیل متن به گفتار چندزبانه ____ **۲۷**

۲-۱. مقدمه ____ **۲۸**

۲-۲. سیستم های تشخیص و تبدیل متن به گفتار چند زبانه ____ **۲۸**

۲-۳. مرورگر صوت ____ **۲۹**

۲-۳-۲. تحلیل های صفحات وب برای مرورگر وب ____ ۳۱

۲-۳-۳. پردازش مرورگر وب ____ ۳۲

۲-۳-۴. برنامه کاربردی صوت ____ ۳۴

۲-۴. سیستم های دیالوگ ۳۴

فصل ۳

زبان علامت‌دار صوتی VXML ۳۷

۳-۱. مقدمه ۳۸

۳-۲. VoiceXML ۳۸

۳-۳. معماری VXML ۴۰

۳-۴. اهداف VXML ۴۱

۳-۵. محدوده VXML ۴۱

۳-۶. مفهوم ۴۱

۳-۶-۱. دیالوگ و زیر دیالوگ ۴۲

۳-۶-۱-۱. فرم ها ۴۳

۳-۷. کاربردهایی برای برنامه های VoiceXML ۴۵

۳-۸. پروتکل MRCP ۴۷

۳-۸-۱. انواع منابع و سرویس ها در MRCP ۴۷

فصل ۴

زبان علامتدار تولید گفتار SSML ۴۹

۴-۱. مقدمه ۵۰

۴-۲. زبان علامت دار تبدیل متن به گفتار ۵۱

۴-۲-۱. درون یک موتور تبدیل متن به گفتار ۵۱

۴-۳. عناصر در SSML ۵۴

فصل ۵

کاربرد سیستم‌های تولید گفتار در محصولات مایکروسافت _____ ۵۸

۵-۱. Microsoft Speech Server 2007 _____ ۶۰

۵-۱-۱. دید کلی از OCS 2007 Speech Server _____ ۶۰

۵-۱-۲. طرز کار Speech Server: _____ ۶۱

۵-۲. تشخیص گفتار و تبدیل متن به گفتار _____ ۶۲

۵-۳. اتصالات مشتری _____ ۶۳

5-4. زبان و گرامر در OCS 2007 _____ ۶۴

۵-۴-۱. Prompt Engine Markup Language (PEML) _____ ۶۶

۵-۴-۲. ساختار گرامر keyword _____ ۶۷

۵-۴-۳. دامنههای voicexml _____ ۶۸

۵-۵. کنترل تعاملی voicexml _____ ۶۸

۵-۶. پیاده سازی و ارزیابی یک راهنمای توریست چند وجهی و چند زبانه _____ ۷۰

۵-۶-۱. کارکرد سیستم _____ ۷۱

۵-۶-۲. معماری سیستم _____ ۷۳

۵-۶-۳. واسط کاربر سیستم _____ ۷۶

۵-۷. قابلیت تشخیص گفتار در ویندوز ویستا _____ ۷۸

۵-۸. نمونه های نرم افزارهای تشخیص گفتار _____ ۷۹

۵-۸-۱. Philips speech SDK _____ ۷۹

5-8-2. Commercial lingsoft speech recognizer _____ ۷۹

5-8-3. Nuance speech recognition software _____ ۸۰

۵-۸-۴. Nuance Vocalizer 3.0 _____ ۸۰

۵-۸-۵. Micro REC SD _____ ۸۱

۵-۸-۶. Microsoft Speech Server (MSS): _____ ۸۱

۵-۸-۷. Nuance V-Builder 2.0 _____ ۸۲

۵-۸-۸. IBM's Voice Toolkit _____ ۸۲

۵-۸-۹. Loquendo _____ ۸۲

۵-۸-۱۰. Neospeech _____ ۸۲

۵-۸-۱۱. Sky Greek ۸۲

۵-۹. چند نمونه نرم افزار تبدیل متن به گفتار ۸۲

۵-۹-۱. VoiceMX STUDIO 4 ۸۲

۵-۹-۲. ECTACO Voice Translator Russian -> Japanese 1.21.24 ۸۳

فصل ۶

سیستم های تبدیل متن به گفتار و فناوری جاوا ۸۴

۶-۱. مقدمه ۸۵

۶-۲. کتابخانه گفتار جاوا چیست؟ ۸۵

۶-۳. اهداف طراحی برای کتابخانه گفتار جاوا ۸۶

۶-۴. برنامه های جاوا با توانایی های گفتاری ۸۷

۶-۵. گفتار و کتابخانه های دیگر جاوا ۸۸

۶-۶. پیاده سازی ها ۸۸

۶-۷. نیازمندی ها ۸۹

۶-۸. موتورهای گفتار javax.speech ۹۱

۶-۸-۱. خصوصیات یک موتور گفتار ۹۲

۶-۸-۲. محلی کردن، انتخاب کردن و ساختن موتورها ۹۵

۶-۸-۲-۱. ساختن موتور به صورت پیش فرض ۹۵

۶-۸-۲-۲. ساختن موتور به صورت ساده ۹۶

۶-۸-۲-۳. انتخاب موتور به صورت پیشرفته ۹۷

۶-۹. حالت های موتور ۹۸

۶-۹-۱. سیستم وضعیت تخصیص ۱۰۱

۶-۹-۲. وضعیت های تخصیص و بلوکه کردن فراخوانی ۱۰۲

۶-۹-۳. سیستم وضعیت مکث-از سر گیری ۱۰۴

۶-۱۰. رویداد های گفتار ۱۰۵

۶-۱۱. بسته javax.speech.synthesis ۱۰۶

۶-۱۲. برنامه Hello World! ۱۰۶

۶-۱۳. زبان نشانه گذاری کتابخانه گفتار جاوا ۱۰۸

۶-۱۳-۱. اهداف JSML ۱۰۹

6-13-2. پردازش اسناد JSML ۱۰۹

6-13-3. عناصر JSML ۱۱۱

6-13-4. مثالی از JSML ۱۱۲

۶-۱۴. انتخاب پیاده سازی مناسب برای کتابخانه گفتار جاوا ۱۱۴

۶-۱۴-۱. پیاده سازی های تولید کننده گفتار ۱۱۴

۶-۱۴-۱-۱. پیاده سازی FreeTTS ۱۱۴

6-14-1-2. پیاده سازی Talking Java SDK ۱۱۵

6-14-1-3. پیاده سازی JSpeechLib ۱۱۶

6-14-1-4. پیاده سازی IBM Speech API ۱۱۷

6-14-1-5. پیاده سازی Lernout & Hauspie's TTS for Java Speech API ۱۱۷

6-14-1-6. پیاده سازی Conversa Web 3.0 ۱۱۷

6-14-1-7. پیاده سازی Elan Speech Cube ۱۱۷

6-14-1-8. پیاده سازی Festival ۱۱۸

۶-۱۴-۲. پیاده سازی های تشخیص دهنده گفتار ۱۱۸

۶-۱۴-۲-۱. پیاده سازی Sphinx 4 ۱۱۸

۶-۱۵. نتیجه گیری ۱۱۹

فصل ۷

استفاده از سیستم‌های تبدیل متن به گفتار در کاربردهای عملی ۱۲۱

۷-۱. مقدمه ۱۲۲

۷-۲. سیستم دیالوگ چند زبانه در محیط آموزشی ۱۲۲

۷-۳. پروژه GEMINI ۱۲۳

۷-۴. شبیه ساز کاربر بر پایه VoiceXML ۱۲۴

۷-۴-۱. توصیفا ت VoiceXML ۱۲۶

۷-۵. توسعه سیستم دیالوگ گفتگو بر پایه GALAXY/VoiceXML ۱۲۸

۷-۵-۱. معماری سیستم ۱۲۸

۷-۵-۲. مدیر دیالوگ ۱۲۹

۷-۶. مرورگر صوتی HearSay ۱۳۰

7-6-1. سیستم Hearsay ۱۳۰

7-6-2. معماری Hearsay ۱۳۰

۷-۶-۳. استفاده از Hearsay ۱۳۱

۷-۷. پرتال صدای دو زبانه عربی/انگلیسی ۱۳۳

۷-۷-۱. استانداردها و تکنولوژی ۱۳۳

۷-۷-۲. دوزبانی بودن و VoiceXML ۱۳۶

۷-۷-۳. عربی کردن VoiceXML ۱۳۷

۷-۷-۴. توسعه دادن نمونه های اولیه ۱۳۸

۷-۷-۴-۱. نمونه پرداخت جریمه ۱۳۸

7-7-4-2. برنامه ای برای بچه ها ۱۳۹

۷-۸. مرورگر وب دو زبانه انگلیسی/ چینی ۱۴۰

۷-۸-۱. بستر مورد استفاده ۱۴۰

۷-۸-۲. برنامه CU Weather ۱۴۰

۷-۸-۳. برنامه CU News ۱۴۳

فصل ۸

پایگاه دادگان گفتاری ۱۴۴

۸-۱. مقدمه ۱۴۵

۸-۲. پایگاه دادهی گفتاری TIMIT ۱۴۶

۸-۳. پایگاه دادهی گفتاری Tfarsdat ۱۴۶

۸-۴. پیکرهٔ متنی زبان فارسی ۱۴۷

۸-۵. پایگاه دادهی گفتاری TIDIGITS ۱۴۷

۸-۶. پایگاه دادهی گفتاری AURORA 2 ۱۴۷

۸-۷. دادگان مورد استفاده در سنتز گفتار ۱۴۸

۸-۷-۱. سنتز اتصالی ۱۴۸

۸-۷-۱-۱. سنتز با امکان انتخاب واحد ۱۴۸

۸-۷-۱-۲. سنتز Diphone ۱۴۹

۸-۷-۱-۳. سنتز در دامنهٔ تعریف مشخص ۱۴۹

جمع‌بندی ۱۵۰

منابع و مراجع ۱۵۲

فهرست اشکال

شکل۱-۱. مراحل تبدیل متن به گفتار ۱۴
شکل۱-۲. روش فرمانت در تبدیل متن به گفتار ۱۵
شکل۱-۳. تولید کننده گفتار فرمانت سری ۲۲
شکل۱-۴. تولید کننده گفتار فرمانت موازی ۲۳
شکل۱-۵. مدل PARCAS ۲۳
شکل۲-۱. مرورگر صوت ۳۰
شکل۲-۲. بستر گفتار ۳۱
شکل۲-۳. معماری مرورگر تلفنی ۳۳
شکل۳-۱. مدل معماری VXML ۴۱
شکل۴-۱. سیستم تبدیل متن به گفتار عمومی ۵۲
شکل۵-۱. نمای کلی چگونگی کار کردن Speech Server با بقیه سرورها ۶۲
شکل۵-۲. نمای کلی مفهومی از چگونگی کار کردن Speech Server با اتصالات مشتریها ۶۴
شکل۵-۳. معماری شماتیک MUST ۷۵
شکل۵-۴. نمایی از صفحه نمایش یک Pocket PC در حال اجرای MUST ۷۸
شکل۶-۱. دیاگرام وضعیت سیستم تخصیص وضعیت ۱۰۲
شکل۶-۲. وضعیت های مکث و از سر گیری یک موتور ۱۰۵
شکل۶-۳. وضعیت های تولید کننده گفتار ۱۰۸
شکل۶-۴. پردازش اسناد JSML ۱۱۰
شکل۶-۵. رابط کاربری Talking Java SDK ۱۱۶
شکل۷-۱. انتقال حالت ها در نمونه VXML ۱۲۸
شکل۷-۲. معماری سیستم دیالوگ Galaxy ۱۲۹
شکل۷-۳. اجزای اصلی واحد مدیریت دیالوگ بر پایه VXML ۱۲۹
شکل۷-۴. معماری HearSay ۱۳۱
شکل۷-۵. تصویری از سایت New York Times ۱۳۲
شکل۷-۶. پرتال دو زبانه ۱۳۴
شکل۷-۷. نمونه از کد VXML ۱۳۹

فهرست جداول

جدول۶-۱. خصوصیات انتخاب موتور پایه ای EngineModeDesc ۹۳

جدول۶-۲. خصوصیات انتخاب تولید کننده گفتار SynthesizerModeDesc ۹۴

جدول۶-۳. خصوصیات انتخاب تشخیص دهنده RecognizerModeDesc ۹۴

جدول۶-۴. ساختن موتور به صورت پیش فرض ۹۵

جدول۶-۵. ساختن موتور به صورت ساده ۹۶

جدول۶-۶. ساختن موتور به صورت ساده به زبان اسپانیایی ۹۷

جدول۶-۷. ساختن موتور به صورت پیشرفته ۹۸

جدول۶-۸. بلوکه کردن فراخوانی ۱۰۲

جدول۶-۹. رویداد های گفتاری: بسته javax.speech ۱۰۶

جدول۶-۱۰. رویدادهای گفتاری: بسته javax.speech.synthesis ۱۰۶

جدول۶-۱۱. بلوکه کردن فراخوانی ۱۰۷

جدول۶-۱۲. مثالی از یک پیام الکترونیکی ۱۱۱

جدول۶-۱۳. عناصر اصلی JSML ۱۱۲

جدول۶-۱۴. مثالی از JSML ۱۱۲

جدول۶-۱۵. مقایسه میان شش پیاده سازی مهم کتابخانه گفتار جاوا ۱۲۰

مقدمه

گفتار اولین وسیله برقراری ارتباط بین مردم است. تبدیل متن به گفتار یا تولید سیگنال‌های گفتار به‌طور اتوماتیک، چنددهه‌ای است که در حال توسعه است. پیشرفت اخیر در زمینه تبدیل متن به گفتار، موجب ایجاد تولیدکننده‌های گفتاری باهوشمندی بالا شده است، اما کیفیت صدا و طبیعی بودن گفتار همچنان بصورت یک مشکل عمده باقی مانده است. به هر حال کیفیت محصولات فعلی در حال حاضر به اندازه کافی خوب است و در کاربردهایی نظیر ارتباطات و کاربردهای چندرسانه‌ای از تبدیل متن به گفتار استفاده می‌شود.

سیستم‌هایی که ارتباط گفتاری انسان را با کامپیوتر برقرار می‌نمایند، معمولا به سیستم‌های تشخیص و تبدیل متن به گفتار چند زبانه احتیاج دارند که قابلیت سوئیچ کردن بین زبان‌های مختلف را داشته باشند، این قبیل سیستم‌ها باید توانایی تشخیص زبانی که توسط کاربر استفاده می شود را داشته باشند تا بتوانند نیازهای کاربر را با موتور تشخیص و تولید گفتار مربوطه تامین کنند.

به منظور استانداردسازی این گونه سیستم‌ها کنسرسیوم وب جهانی استانداردهایی را برای تشخیص گفتار، تولید گفتار از متن و تفسیر معنایی منتشر کرده است که در فصول بعد شرح داده می شوند. استفاده از این استانداردها مزایایی مختلفی دارد، مانند:

- کاهش هزینه توسعه سیستم‌های دیالوگ
- آسان نمودن تکنولوژی‌های مجتمع‌سازی
- به حداقل رساندن فعل و انفعالات بین سرویس‌گیرنده و سرویس‌دهنده

- دور کردن برنامه‌نویس از جزئیات سخت‌افزاری و سطح پائین
- افزایش دادن قابلیت انتقال سیستم

این مزایا باعث شده توسعه دهنده‌ها به سمت استفاده از این استانداردها در سیستم‌های خود پیش روند. این استانداردها از مرورگر صوتی پشتیبانی می‌کنند. مرورگر صوتی وب با استفاده از شبکه تلفن عمومی به تقاضاهای کاربران پاسخ می دهد. ابتدا تقاضای کاربر توسط موتور تشخیص گفتار شناسایی می شود و سپس مرور گر صوت صفحات مورد ارتباط را پیدا کرده و لینک‌های مربوطه را توسط موتور تبدیل متن به گفتار، به صوت تبدیل می کند و با استفاده از استانداردهایVXM به کاربر ارائه می‌نماید.

در این گزارش، ابتدا سیستم‌های تبدیل متن به گفتار معرفی خواهد شد. در این بررسی قسمت‌های اصلی این سیستم‌ها و روش‌های طراحی و پیاده‌سازی هر قسمت مورد نظر قرار خواهد گرفت. با معرفی این قسمت‌ها، نقش دادگان گفتاری در طراحی و پیاده‌سازی عموم آن‌ها مشخص می‌گردد.

در بخش‌های بعد، دو استاندارد زبان علامت‌دار صوتی (VXML) و زبان علامت‌دار تولید گفتار (SSML) که در موتورهای تبدیل متن به گفتار مورد استفاده قرار می‌گیرند، معرفی می‌شوند. ابداع این استانداردها امکان طراحی و کاربرد همه منظوره موتورهای تبدیل متن به گفتار را فراهم می‌سازد.

در فصول بعدی موارد کاربردی‌تر با مروری بر نرم‌افزارهای تولید گفتار مایکروسافت، امکانات و پشتیبانی موجود در فناوری جاوا و نیز مثال‌هایی از کاربردهای عملی موجود، مطرح خواهد شد. در فصل پایانی نیز مروری به پایگاه دادگان مطرح موجود، ارائه شده است. در این مرور ضمن نگاهی به طراحی آن‌ها، کاربردهای ممکن آن‌ها در تبدیل متن به گفتار روشن می‌گردد.

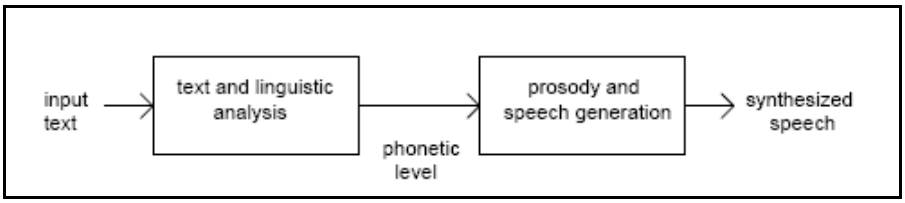
فصل1

معرفی سیستم‌های تبدیل متن به گفتار

۱-۱. مقدمه

تبدیل متن به گفتار از دو فاز عمده تشکیل شده است. فاز اول تحلیل متن که متن ورودی به یک سری علائم و فونم‌ها ونمایشات زبانی تبدیل می‌شود. فاز دوم تولید شکل موج گفتار می‌باشد که خروجی صوتی از این علائم و اطلاعات به دست می‌آید.

این دو فاز معمولاً با نام‌های تولید گفتار سطح بالا و تولید گفتار سطح پایین مشهورند. یک نمودار ساده از این فرایند در شکل۱-۱ آمده است.



شکل1-1. مراحل تبدیل متن به گفتار

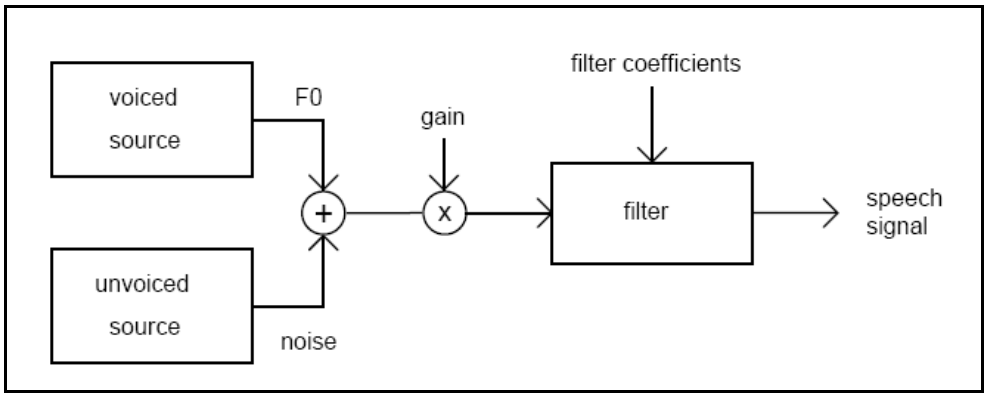
متن ورودی می‌تواند به طور مثال داده‌هایی از یک پردازشگر لغات یا کدهای اسکی استاندارد از یک پست الکترونیک، پیام کوتاه یا متن‌های اسکن شده از یک روزنامه باشد. رشته کاراکتری پیش پردازش و تحلیل می‌شود و به نمایشات فونتیکی که معمولاً رشته‌ای از فونم‌ها با بعضی اطلاعات اضافی نظیر لحن خواندن یا استرس وغیره می‌باشد، تبدیل می‌شود. گفتار در نهایت توسط تولیدکننده‌های گفتار سطح پایین و با استفاده از اطلاعات حاصل از تولید کننده‌های گفتار سطح بالا، تولید می‌شود.

راحت‌ترین راه برای تولید گفتار سطح پایین گفتار، استفاده از یکسری نمونه‌های از قبل ضبط شده از گفتار طبیعی می‌باشد که می‌تواند شامل کلمات ساده یا جملات باشد. این روش اتصالی کیفیت بسیار بالایی دارد و خیلی هم طبیعی است اما محدودیت در زمینه دایره لغات داشته و معمولاً شامل یک صدا است.

این روش برای بعضی سیستم‌های اطلاعاتی و هشدار دهنده بسیار مناسب است. اما به هرحال بسیار واضح است که نمی‌توان بانک اطلاعاتی مجتمع بر همه لغات و نام‌های رایج در دنیا داشت. حتی اگر این امر

ممکن شود، به این کار تبدیل متن به گفتار واقعی نمی‌گویند. زیرا تنها شامل کلمات از قبل ضبط شده است. بنابراین برای تولید گفتار غیر محدود، باید از اجزای کوچکتری از صوت گفتار، مثل سیلاب‌ها یا فونم‌ها یا حتی بخش‌های کوتاه‌تر استفاده شود.

روش دیگری که به طور گسترده برای تولید گفتار سطح پایین گفتار مورد استفاده قرار می گیرد، تولید گفتار به روش فرمانت است. این روش براساس مدل فیلتر کردن منبع می باشد و در شکل۱-۲ نشان داده شده است.



شکل1-2. روش فرمانت در تبدیل متن به گفتار

این روش فقط منبع صوتی و فرکانس‌های فرمانت را مدل می‌کند و ویژگی‌های فیزیکی قطعات صوتی را در نظر نمی‌گیرد . سیگنال تحریک می‌تواند یک سیگنال باصدا[1] با فرکانس اساسی f0 باشد یا یک نویز (بی‌صدا) باشد. یک تحریک مختلط از این دو روش می‌تواند برای صامت‌ها و بعضی اصوات حلقی مورد استفاده قرار گیرد. سپس تحریک در یک بهره ضرب می‌شود و با یک فیلتر صوتی، که این فیلتر از تشدیدکننده‌هایی شبیه به فرمانت‌های گفتار طبیعی ساخته می‌شود، فیلتر می‌شود.

در تئوری، صحیح‌ترین مدل برای تولید گفتار هوشمند مدل‌کردن سیستم تولید گفتار به طور مستقیم

[1] Voiced

است. این روش تولید گفتار مفصلی[1] جهاز صوتی خوانده می شود که شامل مدل‌هایی از جهاز صوتی و تارهای صوتی است.

تولید گفتار مفصلی معمولاً به وسیله مجموعه‌ای از توابع ناحیه‌ای که متشکل از بخش‌هایی با لوله‌های صوتی است، مدل می‌شود. تولید گفتار با این روش دارای کیفیت بالایی است ولی با توجه به پیچیدگی آن هنوز به نتایج خوبی نرسیده است. همه روش‌های تولید گفتار مزایا و مشکلات خاص خودشان را دارند و بسیار مشکل است که بخواهیم بهترین روش را انتخاب کنیم. با تولید گفتارهای اتصالی و فرمانت نتایج بسیار خوبی اخیراً به دست آمده است. تولید گفتار مفصلی دارای این پتانسیل هست که در آینده رشد کند.

در ادامه این فصل، ابتدا مروری بر تاریخچه تولید گفتار انجام می‌شود. سپس مراحل تولید گفتار در سطح بالا و روش‌های تولید گفتار سطح پایین گفتار، شرح داده می‌شود. در انتهای فصل، اشاره‌ای گذرا به برخی روش‌های موجود دیگر در تبدیل متن به گفتار می‌شود[۱].

۱-۲. تولید گفتار سطح بالا[2]

در اولین مرحله تولید گفتار، متن ورودی به نحوی بازنویسی می‌شود که تولید گفتارکننده مرحله دوم بتواند از روی آن خروجی صوتی بسازد. پیاده‌سازی شایسته این مرحله اساسی‌ترین بحث در بین سیستم‌های موجود می‌باشد. این مرحله سه بخش عمده دارد:

- پردازش و تحلیل اعداد، نویسه‌های خاص، مخففها و معادل‌های اختصاری عبارت‌های بزرگ
- تفکیک و تشخیص عبارت‌هایی که با تلفظ متفاوت، معانی متفاوت پیدا می‌کنند
- تجزیه و تحلیل پروزودی گفتار (وزن یا عروض)

[1] Articulate

[2] High level Synthesis

پس از پردازش مرحله نخست، اطلاعات به پردازشگرهای مرحله دوم داده خواهد شد که نوع پردازشگر مرحله دوم به بعضی مشخصه‌های متن بستگی دارد. مراحل تولید گفتار سطح بالا در ادامه شرح داده شده است.

۱-۲-۱. پردازش متن

نخستین وظیفه تمام سیستم‌های تبدیل متن به گفتار، تبدیل داده‌های ورودی به قالبی مناسب برای یک سیستم تولید گفتار است. در این مرحله نویسه‌های غیرالفبایی، اعداد، مخففها و عبارت‌های اختصاری باید به شکل الفبایی کامل، بسط داده شوند. این پردازش اغلب به‌وسیله جداول متناظر یک به یک انجام می‌شود. اما در برخی موارد، اطلاعات تکمیلی در مورد کلمات مجاور نیز مورد نیاز است. همین نکته کوچک سبب ایجاد یک پایگاه داده بزرگ و وضع قوانین پیچیده می‌شود. همچنین ممکن است دربر گیرنده بعضی کاراکترهای کنترلی باشد که باید بدون تغییر در این مرحله، به سیستم‌های مرحله بعد تحویل داده شوند. در زبان‌های غیرفارسی، مشکل اصلی بر سر فهم عداد و عبارت‌های عددی می‌تواند باشد. مانند تشخیص این‌که 2/5 به معنای «دو تقسیم بر پنج» است یا «دوم ماه می».

اما مشکل اصلی که در زبان فارسی با آن مواجهیم این است که برخلاف زبان‌های انگلیسی، فرانسه و...، حروف صدادار در این زبان بسیار اندکند (ا، و، ی) و اکثر کلمات با اعراب (َ، ِ، ُ) خوانده می‌شوند که اصولا نوشته نمی‌شوند. فارسی‌زبانان نیز برای تلفظ کلمات از پردازش ذهنی استفاده نمی‌کنند بلکه با مراجعه به حافظه خود، اعراب متن را به خاطر می‌آورند.

در عین حال استفاده از کلمات و عبارت‌های اختصاری در فارسی محدود به عباراتی نظیر (عج) و (ع) می‌باشد. البته تشخیص استفاده از عبارت (علیه السلام) یا (علیهم السلام) یا صورت‌های دیگر آن، بستگی به اسم قبل دارد و می‌تواند مشکل‌ساز باشد.

۱-۲-۲. تلفظ

یکی از مباحث مهم در تولید گفتار، تلفظ صحیح کلمات است. خصوصا در مورد برخی نرم‌افزارهای‌تلفنی که بیشتر اسامی معابر و مکان‌ها در آنها تلفظ می‌شود. یک راه این است که تا حد ممکن نام‌ها را (به همراه تلفظ) در یک جدول ذخیره کنیم که با توجه به مقدار بسیار زیاد اسامی موجود، این راه عملی نیست. بنابراین روش دیگری (در سال ۱۹۹۳ توسط Belhoula) ارائه شده که مناسب‌تر به نظر می‌رسد و آن اینکه کل سیستم را بر مبنای قواعدی برپا کنیم و یک جدول یا لغتنامه محتوی استثناها برای کلماتی که خواندن آنها توسط قواعد اصلی با شکست مواجه شده، داشته باشیم. این راه برای تلفظهای غیرعادی (غیر از اسامی خاص) نیز مناسب است.

در این روش یک کلمه را به اجزاء تشکیل‌دهنده آن تفکیک می‌کنیم (هجا) مانند پیشوند، پسوند، ریشه. تحقیقی که در سال ۱۹۸۷ توسط Allen انجام شده نشان می‌دهد با حدود دوازده هزار هجا می‌توان چیزی در حدود ۹۵% از زبان انگلیسی را پوشش داد.

۱-۲-۳. پروزودی

برای تلفظ صحیح صدا باید نکاتی مثل زیر و بمی صدا در حین گفتار، کشیدگی یا دوره صوت و استرس روی کلمات نیز توجه کرد. در یک سیستم تلفظ ایده‌آل باید به خصوصیاتی نظیر جنس صدا، سن، هیجانات و احساسات نیز اهمیت داد. اما توجه به تمام این نکات که به وزن صوت برمی‌گردد،پیاده‌سازی سیستم را بسیار پیچیده می‌کند.

نکات مربوط به وزن گفتار را می‌توان به سه رده تقسیم نمود: سطح هجاها، سطح کلمات و سطح جمله یا عبارات. زیر و بمی صدا یا فرکانس اصلی صوت در یک گفتار واقعی به پارامترهای زیادی بستگی دارد.

صعودی یا نزولی بودن تون صدا در بیان جمله می‌تواند از معنای جمله استخراج شود. مثلا در جملات

پرسشی انتهای جمله معمولا با افزایش تون صدا بیان می‌شود درحالیکه در جملات عادی در انتهای جمله تون صدا کاهش می‌یابد. همچنین بالا رفتن تون صدا در انتهای جمله می‌تواند نشانه این باشد که جمله دیگری در ادامه خواهد آمد. نهایتا تون صدا به خصوصیات شخصی گوینده و حالت او نیز بستگی دارد.

۱-۳. تولید گفتار سطح پایین یا تولید شکل موج گفتار

گفتار تولید گفتار شده می تواند با روش‌های متفاوت تولید شود. همه این روش‌ها دارای معایب و محاسنی هستند که در این بخش شرح داده شده است. روش‌های تبدیل متن به گفتار به سه گروه عمده تقسیم می شوند:

- تولید گفتار مفصلی (جهاز صوتی): که سعی در مدل سازی سیستم تولید گفتار انسانی بطور مستقیم دارد.
- تولید گفتار فرمانت: که قطب‌های فرکانسی سیگنال گفتار یا تابع تبدیل علائم صوتی (نشانه-های صوتی) را مدل می‌کند و بر اساس مدل فیلتر کردن منبع است.
- تولید گفتار اتصالی: که از نمونه‌های از قبل ضبط شده که دارای طول‌های متفاوتی‌اند و از گفتار طبیعی بدست آمده‌اند، استفاده می‌کند.

در سیستم‌های امروزی فرمانت وروش‌های اتصالی رایج‌ترند.

۱-۳-۱. تولید گفتار مفصلی

تولید گفتار مفصلی سعی در مدل‌سازی اعضای صوتی بدن بطور کامل دارد. بنابراین بطور بالقوه بهترین روش برای تبدیل متن به گفتار با کیفیت بالا می باشد. اما از آنجا که پیاده‌سازی آن مشکل است نسبت به

روش‌های دیگر مورد توجه کمتری قرار گرفته است وهنوز موفقیت‌های چشمگیری در این زمینه حاصل نشده است.

تولید گفتار مفصلی بطور عمده به مدل‌سازی اجزای دستگاه صوتی انسان وتارهای صوتی او می‌پردازد. اجزای دستگاه صوتی معمولا بوسیله گروهی از توابع ناحیه‌ای بین دهانه حنجره ودهان مدل می‌شوند. اولین مدل اجزای دستگاه صوتی متشکل از جدولی از توابع ناحیه‌ای علائم صوتی، از حنجره تا لب‌ها، برای هر بخش آوایی می‌باشد.

در تولید گفتار باقاعده برای مثال پارامترهای کنترلی می‌تواند گشادی (میزان باز شدن) دهان باشد. فشار لب‌ها و بلندی نوک زبان ومحل برخورد زبان با هربخش دهان را نیز می‌توان در نظرگرفت.

پارامترهای تحریک می تواند گشادگی حنجره، مقاومت تارهای صوتی وفشار ریه باشد.

هنگامی که صحبت می‌کنیم، ماهیچه‌های تارهای صوتی موجب می‌شوند که اجزای دستگاه صوتی جابه‌جا شوند وشکل علائم صوتی با صداهای مختلف ایجاد شود.

مدل اجزای دستگاه صوتی معمولا از تحلیل اشعه x گفتار طبیعی بدست می‌آید. به‌هرحال این اطلاع فقط دوبعدی است، در حالی که علائم صوتی واقعی سه بعدی اند.

بنابراین تولید گفتار مفصلی مبتنی بر قانون بسیار سخت برای بهینه کردن است زیرا اطلاع کافی از حرکت دستگاه صوتی در حین صحبت نداریم.

مشکل دیگر تولید گفتار مفصلی این است که اطلاع اشعه X، جرم (سنگینی وبمی) ودرجه آزادی اجزای دستگاه صوتی را مشخص نمی‌کند. مزیت تولید گفتار مفصلی این است که مدل نشانه‌های صوتی اجازه مدل-سازی صحیح تغییرات را با توجه به تغییرات ناحیه‌ای ناگهانی می‌دهد. در حالی که تولید گفتار فرمانت فقط رفتار طیفی را مدل می‌کند.

تولید گفتار مفصلی بسیار کم در سیستم‌های کنونی مورد استفاده قرار می گیرد. اما از آنجا که روش-

های تحلیل به‌سرعت در حال گسترش است، می‌تواند بعنوان یک روش بالقوه، در آینده مورد استفاده قرار گیرد.

۱-۳-۲. تولید گفتار فرمانت

این روش مبتنی بر مدل فیلتر کردن منبع گفتار است و در دهه‌های گذشته زیاد مورد استفاده قرار گرفته است.

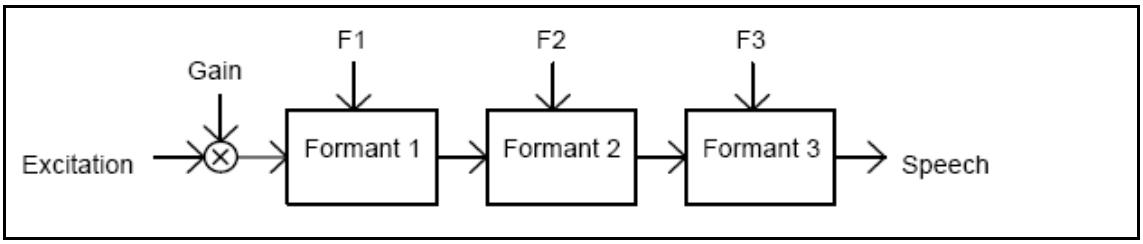
تولید گفتار فرمانت دو ساختار موازی و سری دارد و گاهی ترکیبی از هر دو مورد استفاده قرار می‌گیرد. این روش تعداد نامحدودی از صداها را تدارک می‌بیند و از روش‌های اتصالی انعطاف‌پذیرتر است.

حداقل ۳ فرمانت برای تولید گفتار هوشمند نیاز است. برای کیفیت بالاتر به بیش از ۵ فرمانت نیاز است. هر فرمانت معمولا با یک وسیله ارتعاشی دو قطبی مدل می شود. این مدل فرکانس فرمانت و پهنای باند آن را مشخص می کند.

تولید گفتار فرمانت مبتنی بر قاعده، براساس مجموعه‌ای از قوانین است که برای مشخص کردن پارامترهای ضروری به منظور تولید گفتار یک نطق از تولیدکننده گفتار فرمانت استفاده می کند. پارامترهای ورودی می‌تواند موارد زیر باشد:

- فرکانس اساسی صوت
- خارج قسمت باز تحریک صوت
- درجه صدا در تحریک
- فرکانس‌های فرمانت (f3،f2،f1) ودامنه ها(A3،A2،A1)
- فرکانس یک اسباب ارتعاش که دارای فرکانس پایین اضافی باشد
- شدت بالاو پایین بودن ناحیه فرکانسی

شکل ۱-۳، یک تولیدکننده گفتار فرمانت سری متشکل از اسباب ارتعاش ودارای باند گذر که بطور سری به هم متصل شده‌اند را نمایش می‌دهد.



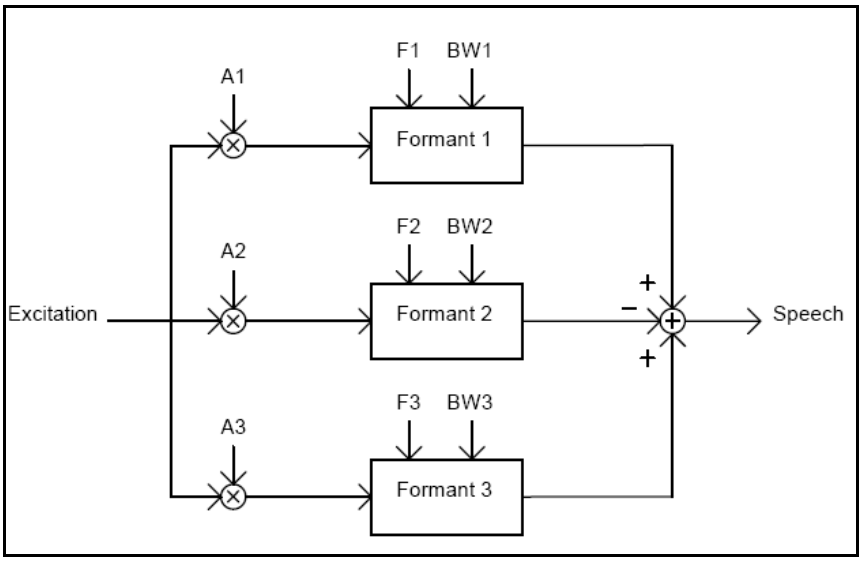
شکل1-3. تولید کننده گفتار فرمانت سري

ساختار سری فقط به فرکانس‌های فرمانت به‌عنوان اطلاعات کنترلی نیاز دارد. مزیت عمده ساختار سری این است که دامنه‌های فرمانت نسبی برای حروف صدادار به کنترل کننده‌های فردی نیاز ندارد.

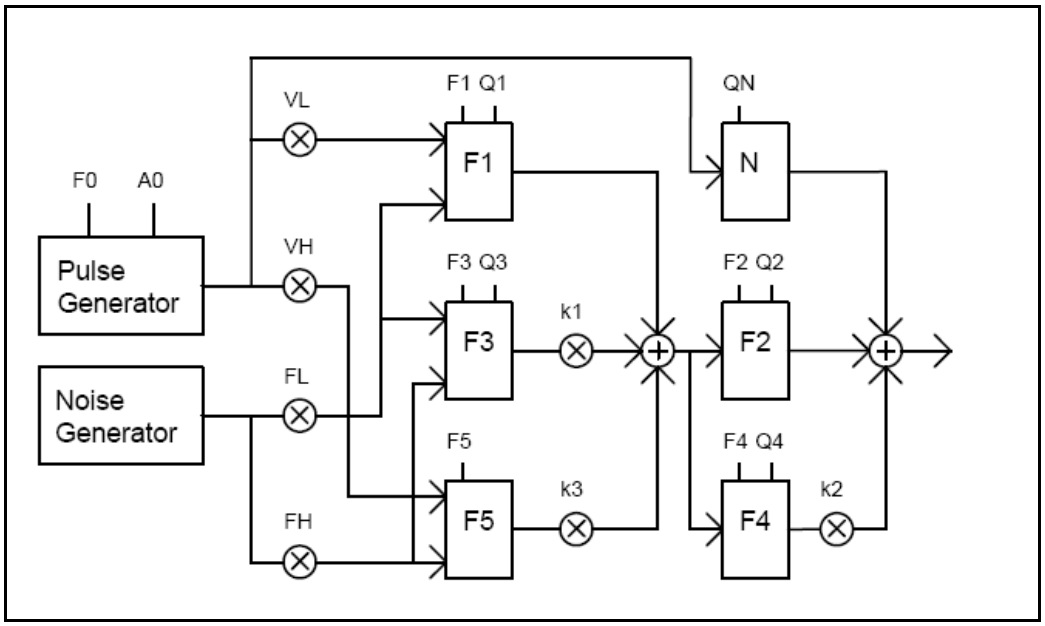
ساختار سری برای صداهای غیر وابسته به بینی مناسب‌تر است.این ساختار به اطلاعات کنترلی کمتری نسبت به ساختار موازی نیاز دارد و پیاده‌سازی ساده‌تری دارد. یک تولیدکننده گفتار فرمانت موازی از اسباب ارتعاشی موازی ساخته شده است. گاهی اوقات وسیله‌های ارتعاشی اضافی برای حروف وابسته به بینی لازم است. سیگنال تحریک به همه فرمانت‌ها به‌طور همزمان داده می‌شود و خروجی آن‌ها با هم جمع می‌شود.

ساختار موازی کنترل پهنای باند و بهره را برای هر فرمانت بطور جداگانه قادر می‌کند و بنابراین به اطلاعات کنترلی بیشتر نیاز دارد.

ساختار موازی (شکل۱-۴) برای حروف وابسته به بینی و همچنین صامت بهتر است. اما بعضی حروف صدادار با تولیدکننده گفتار فرمانت موازی بخوبی سری قابل مدل نیست. مقایسه میان این دو ساختار بسیار بحث برانگیزاست.



شکل1-4. تولید کننده گفتار فرمانت موازي



شکل1-5. مدل PARCAS

ضرایب K1،K2 وK3 ثابت‌اند و برای برقراری تعادل دامنه‌های فرمانت انتخاب شده‌اند. مدل PARCAS

(شکل۱-۵) مجموماً از ۱۶ پارامتر کنترلی استفاده می‌کند که عبارتند از:

- F0 وA0: دامنه وفرکانس اصلی جزء صوتی
- FN وQN: فرکانس فرمانت و پهنای باند
- VL وVH: دامنه جزء صوتی ؛بالاوپایین
- FL و FH: دامنه جزء غیر صوتی (Unvoiced)
- QN: مقادیر فرمانت مربوط به بینی در 250h

سیگنال تحریک در تولید گفتار فرمانت شامل بعضی انواع منبع صوتی یا نویز سفید است. اولین سیگنال‌های منبع صوتی مورد استفاده از نوع دندانه‌ای ساده بودند. در سال ۱۹۸۱ Deniss Klatt منبع صوتی پیچده‌تری معرفی کرد.

صحت ودقت منبع انتخاب شده مهم است. مخصوصاً وقتی که کنترل خوب ویژگی‌های گفتاری مد نظر است.

فیلترهای فرمانت فقط تشدیدهای علائم صوتی را نشان می دهد. بنابراین تدارکات اضافی برای تأثیرات مربوط به شکل موج حنجره و ویژگی‌های انعکاسی مربوط به دهان لازم است.

معمولاً شکل موج حنجره‌ای[1] با فیلتر 12db/octave- و ویژگی‌های انعکاسی با فیلتر 6db/octave- تخمین زده می‌شود.

۱-۳-۳. تولید گفتار اتصالی[2]

اتصال نطق‌های طبیعی از قبل ضبط شده ساده‌ترین راه برای تولید گفتار هوشمندانه گفتار طبیعی

[1] glottal waveform

[2] Concatenative Synthesis

است. تولید کننده‌های گفتار اتصالی معمولاً به یک گوینده و یک صدا نیاز دارند و نسبت به سایر روش‌ها حافظه بیشتری اشغال می کنند.

یکی از جنبه‌های مهم در تولید گفتار اتصالی، انتخاب واحدهای آوایی با طول مناسب است. درست‌ترین انتخاب عبارت از برقراری تعادل میان واحدهای آوایی کوتاه و بلند است. در صورت انتخاب واحدی با طول بلندتر، گفتار طبیعی‌تر خواهد بود، اما میزان بخش‌های مورد نیاز و حافظه افزایش می‌یابد. با انتخاب واحد آوایی کوتاه‌تر، حافظه کمتری نیاز است، اما جمع‌آوری نمونه‌ها و فرآیند برچسب‌گذاری سخت‌تر و پیچیده‌تر خواهد بود. در سیستم‌های امروزی کلمه، سیلاب، نیم سیلاب و فونم، به عنوان واحد آوایی مورد استفاده قرار می‌گیرند.

کلمه، طبیعی‌ترین بخش برای متن‌های نوشته شده و سیستم‌های پیغامی با دایره لغت خیلی محدود است. ولی از آنجا که صدها هزار لغت متفاوت در هر زبانی وجود دارد، لغت یک واحد انتخابی مناسب برای سیستم گفتار به متن نامحدود نیست.

تعداد سیلاب‌های مختلف در هر زبانی کمتر از تعداد کلمه‌هاست. برای مثال در انگلیسی حدود ۱۰۰۰۰ سیلاب هست. اما از آنجا که تلفظ دقیق در سیلاب‌ها وجود ندارد، (بر خلاف کلمه‌ها که تلفظ‌ها مشخص است)، استفاده از آن‌ها منطقی نیست و راهی برای کنترل عروضی جمله نیست. راه معقول استفاده از فونم‌ها یا نیم‌سیلاب‌ها یا ترکیبی از این‌هاست.

نیم‌سیلاب‌ها بخش ابتدایی و انتهایی سیلاب‌ها را نشان می‌دهند وبیشتر تغییرات از ابتدا تا انتها را نگه-می‌دارند. به همین علت است که حدود ۱۰۰۰ تا از آن‌ها برای ساختن ۱۰۰۰۰ سیلاب انگلیسی کافی است.

استفاده از نیم‌سیلاب به جای فونم به نقاط اتصال کمتری نیاز دارد. در این حالت حافظه مورد نیاز باز هم زیاد ولی قابل قبول است.

تعیین دقیق تعداد نیم‌سیلاب‌ها ممکن نیست. بنابراین با این روش نمی‌توان همه لغات را تولید گفتار

کرد. فونم‌ها رایج‌ترین ونرمال‌ترین بخش‌های مورد استفاده در تبدیل متن به گفتارند. تعداد آن‌ها بین ۴۰ تا ۵۰ عدد است. استفاده از فونم‌ها بیشترین انعطاف را در تبدیل متن به گفتار ایجاد می‌کند.

۱-۴. نتیجه‌گیری

در فصل جاری روش‌های معمول تولید گفتار مورد بررسی قرار گرفت. چنان که اشاره شد، تولید گفتار دارای دو مرحله تولید در سطح بالا و تولید در سطح پایین است. در مرحله اول، متن به فرمی مناسب برای ایجاد صوت از روی آن، تبدیل می‌شود و در مرحله دوم، تولید صوت صورت می‌پذیرد که این خود می‌تواند به روش اتصالی، روش فرمانت یا روش مفصلی محقق شود. برای پیاده‌سازی هر یک از دو سطح تولید گفتار، مجموعه دادگان مناسب باید مورد استفاده قرار گیرد.

در فصل بعد به کاربردهای مختلف تولید گفتار و مسائلی چون چند زبانگی در تبدیل متن به گفتار، پرداخته می‌شود.

فصل2

سیستم‌های تشخیص و تبدیل متن به گفتار چندزبانه

۲-۱. مقدمه

در سال‌های اخیر یک دگرگونی برای معرفی و پخش شدن سریع تکنولوژی اینترنت رخ داد .از آنجا که به دلایل مختلفی همچون محل جغرافیایی ، فقدان تخصص و تجربه ، فقدان تجهیزات کامپیوتری و غیره تمامی مشتریان دارای دسترسی آسان به اینترنت نیستند، استفاده از سیستم‌های دیالوگ و صوت تحت وب که ارتباط با شبکه جهانی وب را از طریق صوت میسر می سازد یک امر ضروری است.

در طرف دیگر سهم زیادی از جمعیت به شبکه تلفنی سوئیچ عمومی PSTN[1] یا شبکه بدون سیم یا هر دو دسترسی دارند . بنابراین اقداماتی که مزیت اینترنت را برای اجماع با اینترنت و تکنولوژی های تشخیص صدا در یک سرویس که دسترسی آسان به اینترنت را از طریق رابط صدا (برای مثال تلفن) امکان پذیر می سازد مورد درخواست است این دسترسی به اینترنت از هر مکان و هر زمان با استفاده از شماره وسیله (تلفن، موبایل، کامپیوتر با استفاده از VOIP[2] و غیره) امکان پذیر می کند.

در این فصل به بررسی سیستم های دیالوگ و مرورگرهای صوت و عملکرد آن ها می پردازیم.

۲-۲. سیستم های تشخیص و تبدیل متن به گفتار چند زبانه

سیستم‌هایی که ارتباط گفتاری انسان را با کامپیوتر برقرار می نمایند، معمولا به سیستم های تشخیص و تبدیل متن به گفتار چند زبانه احتیاج دارند که قابلیت سوئیچ کردن بین زبانهای مختلف را داشته باشند.

مزیت استفاده از صدا نسبت به ورودی های دیگری مثل صفحه کلید این است که ارتباط از طریق صدا

[1] Public Switched Telephone Network

[2] Voice over Internet Protocol

دستاورد طبیعی تری در انجام تکنولوژی های تجارت الکترونیکی است. با پرتال صدا یک دستیابی بدون استفاده از دست را به اطلاعات فراهم می کنیم اگر چه سرعت آن در مقایسه با اپراتور انسانی کم است اما سرعت پاسخگو یی قابل قبولی دارد . پرتال صدا همچنین برای کاربران مختلف همچون گروه های سنی یا مشتریانی با نیازهای مخصوص امکانات فراهم می کند .

در سال های اخیر تحقیقات زیادی در مورد استاندارد سازی یک سیستم تشخیص صدا وجود داشته و باعث نزدیک کردن تکنولوژی به روشی که انسان ها با هم ارتباط برقرار می کنند شده است. ایده استفاده از شبکه تلفن سوئیچ عمومی (خطوط زمینی) و شبکه های بی سیم (موبایل) برای دستیابی به اطلاعاتی که به صورت خواندنی بر روی وب موجودند به یک مورد مطلوب و کاربردی تبدیل شده است .

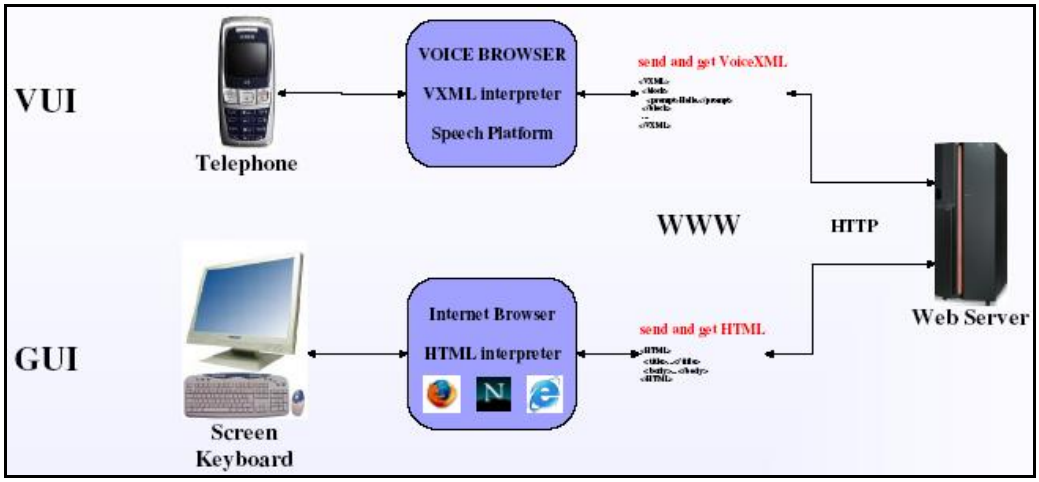
پرتال های صدا که با پرتال های وب برابری می کنند دستیابی به اطلاعات وب را از طریق یک رابط صدا اجازه می دهند[۳].

۲-۳. مرورگر صوت

مرورگر صوت مشابه جستجوی اینترنت کار می کند . به جای استفاده از صفحه کلید ، ماوس و صفحه نمایش ، اطلاعات اینترنت را از طریق تلفن بدست می آوریم . همچون یک مرورگر اینترنت مثل Internet Explorer که کدهای HTML[1] را به صورت گرافیکی نمایش می دهد (graphic user interface – GUI) رابط کاربر گرافیکی یک مرورگر صدا کدهای منبع VoiceXML را به صورت صوتی (VUI[2] رابط کاربر صوتی) نمایش می دهد[۹]. نمای کلی یک مرورگر صوت در شکل۲-۱ نشان داده شده است.

[1] Hyper Text Markup Language

[2] Voice User Interface



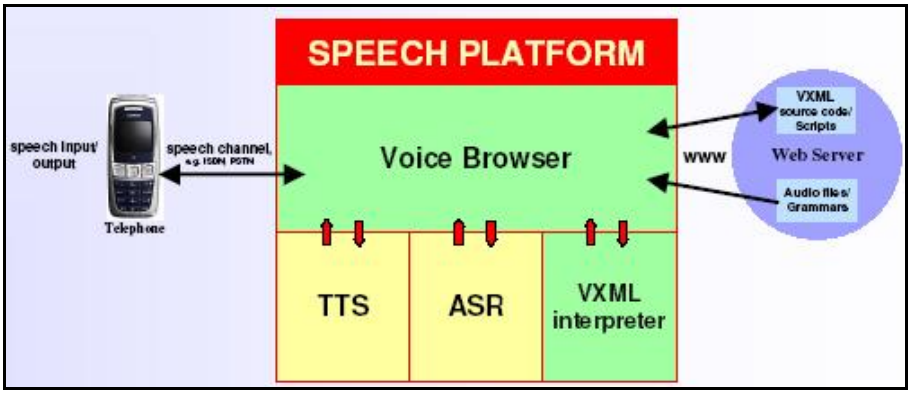
شکل2-1. مرورگر صوت

یک مرورگر صوت یک قسمتی از بستر گفتار[1] است که ورودی و خروجی ها و عناصر دیگر گفتار را مثل موتور TTS[2] و ASR[3] و مفسر VXML را مدیریت می کند .

متد کار سیستم : یک تقاضا که توسط تماس گیرنده ارائه شده توسط تشخیص دهنده اتوماتیک گفتار تشخیص داده می شود . مرورگر صوت تقاضا را با گرامرها یا صفحات VXML روی سرور وب تحلیل می کند و برای یک پاسخ مناسب جستجو می کند که یک صفحه VXML است که توسط سرور وب فرستاده می شود . کد VXML توسط مفسر کد می شود و با موتور تبدیل متن به گفتار، تولید گفتار می شود . پاسخ از طریق شبکه تلفن مربوط به تماس گیرنده می رسد .

بستر گفتار یک دروازه میان وب و شبکه تلفن (برای مثال ISDN[4] و PSTN) است. شکل۲-۲ نمایی از بستر گفتار را نشان می‌دهد.

[1] Speech platform
[2] Text to Speech Synthesis
[3] Automatic Speech Recognition
[4] Integrated Services Digital Network



شکل2-2. بستر گفتار

با وجود شباهت هایی که بین HTML و VXML وجود دارد ، همچنین تعدادی تفاوت اساسی بین آن ها وجود دارد ، برای مثال وسیله ای که برای جستجو استفاده می شود و مکانی که تکنولوژی مرورگر قرار می گیرد. در سرویس دیداری وسیله معمولا یک کامپیوتر شخصی است ، که شامل اجزای تکنولوژی مرورگر است . در سرویس صوتی ، وسیله تلفن یا تلفن بی سیم است که شامل اجزای تکنولوژی مرورگر نیست بلکه زمانی که یک مشتری به سرویس تلفنی سازمان دستیابی پیدا می کند ، یک مرورگر که روی بستر VXML، اجرا می شود ، فراهم می شود.بستر VXML یک مرورگر برای سرویس تلفنی است[۱۰] .

۲-۳-۲. تحلیل های صفحات وب برای مرورگر وب

رابط های کاربر صوتی (VUIS[1]) به طور سریع در حال تکمیل شدن برای شامل شدن مفاد بر پایه وب هستند . انگیزه اصلی ، تامین کردن یک وسیله موجود گسترده برای تولید برنامه های صوتی دارای فعل و انفعال جدید است .

[1] Voice user interface system

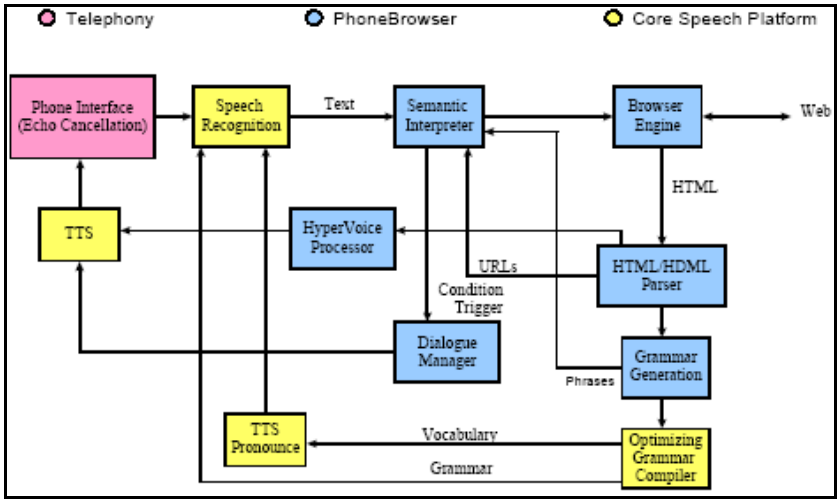
مرورگری صوت اسناد وب تنها در دو سال اخیر از آزمایشگاه ها به برنامه های تجاری وارد شده اند . قبل از آن اغلب برنامه های صوت شبکه از زبان های اسکریپت نویسی اختصاصی برای تعریف ارتباط متقابل دیالوگ کاربر استفاده می کردند .

یکی از زبان‌های استفاده از استانداردهایی مثل VoiceXML این بود که تعداد زیادی از سایت های وب موجود دارای برنامه هایی بر پایه HTML هستند . و این نیازمند صرف هزینه جدیدی برای دوباره نویسی این سایت ها دارد . بنابراین استفاده از مفاد HTML موجود به عنوان پایه ای برای تعدادی از برنامه های صوتی مفید است .

مرورگر تلفن مستقل از گوینده است که سرور پروکسی را فعال می کند و توصیف صوتی از اسناد وب را روی هر تلفن به طور اتوماتیک تامین می کند و به صورت اتوماتیک عنوان های لینک شده را با فرمان صوتی فعال می کند .

۲-۳-۳. پردازش مرورگر وب

معماری مرورگر تلفن در شکل۲-۳ نشان داده شده است . پردازش با یک تماس تلفنی توسط کاربر شروع می شود . عبارتی که کاربر ادا کرده باعث به وجود آمدن اولین اتصال به وب سایت مربوط با آن عبارت می شود و اولین صفحه وب را بارگذاری می کند .



شکل2-3. معماري مرورگر تلفني

فرمان های کنترل مرورگر مثل "go back" و "go home" هم علاوه بر ترکیبات کلمات در عنوان ها برای تولید یک رنج وسیع از زیر مجموعه های آن عنوان توسط گرامر محاسبه می شوند . پردازش اجازه می دهد تا کاربر هر کلمه و عبارتی را از عنوان (حتی ممکن است آن عنوان کامل ادا نشود) را بگوید تا لینک مربوطه را فعال کند .

بعد این فرهنگ لغات بدست آمده مربوط گرامر توسط موتور متن به گفتار پردازش می شوند که نسخه فونتیکی را برای هر کلمه گرامر فراهم می کند . گرامر و فرهنگ لغات سپس در تشخیص دهنده گفتار بارگذاری می شوند ، تا برای عملیات بعدی که تشخیص پاسخ کاربر است آماده شوند.

همزمان اسناد وب برای کاربر توضیح داده می شوند و کاربر ممکن است که یک لینک یا یک فرمان کنترلی مرورگر را برای کنترل ادا کند . اگر فرمان کاربر مبهم باشد مدیر دیالوگ یک لیست از تفاسیر ممکن جمع آوری می کند و از کاربر می خواهد که فرمان مربوطه را انتخاب کند.[۱۱]

زمانی که کاربر یک عنوان را بیان می کند توسط تشخیص دهنده گفتار بررسی می شود و صفحه مربوطه بارگذاری می شود.

۲-۳-۴. برنامه کاربردی صوت

برنامه‌ای که تماس گیرنده تنها از طریق تلفن با آن ارتباط دارد برنامه صوتی نامیده می شود . این برنامه شامل کد منبع VXML ، گرامرها ، اسکریپت ها ، فایل های صوتی است و روی سرور وب ذخیره می شود[۹].

۲-۴. سیستم های دیالوگ

در سال های اخیر تکنولوژی‌های جدیدی برای آسان کردن ارتباط کاربران با کامپیوترها و محیط های اطراف ارائه شده است . یکی از آن ها " سیستم های دیالوگ " هستند.

سیستم های دیالوگ برنامه های کامپیوتری برای برای فعل و انفعال با کاربر برای انجام دادن یک کار به خصوص است . این سیستم ها امروزه برای تامین کردن اطلاعات درباره هواپیما یا زمان بندی قطار ، مسیریابی تماس یا اطلاعات آکادمیکی به کار می رود .

در سیستم های دیالوگ گفتگو فعل و انفعال سیستم _ کاربر با استفاده از گفتار به عنوان تنها وسیله ارتباطی انجام می شود .

این سیستم ها معمولا برای تامین سرویس های تلفنی اتوماتیک استفاده می شوند .در سیستم های دیالوگ چند حالته ارتباط انسان و کامپیوتر بر انواع مختلف کانال های ارتباطی مثل گفتار ، متن ، گرافیک ، ژست بدن و دست بنا نهاده شده است . اطلاعات که توسط کاربر تولید می شود توسط انواع مختلف وسیله همچون میکروفون ، صفحه کلید ، دوربین ها برای بینایی مصنوعی یا صفحات نمایش حساس به لمس گرفته می شود .

برای مثال ممکن است که به کاربران پیشنهاد داده شود که در محیط های نویزی به جای تلفظ کلمات داده را از طریق صفحه کلید تایپ کند. به علاوه این ویژگی به کاربران فعل و انفعال با سیستم را با به کار بردن کیفیت های ارتباط که بهتر با نیاز های آن ها منطبق می شود اجازه می دهد . (برای مثال کاربرانی که مشکلات بینایی دارند می توانند از گفتار استفاده کنند در حالی که افرادی که مشکلات گفتاری و بیان دارند از صفحه کلید استفاده می کنند)[۱۲].

VoiceXML به عنوان یک زبان علامت دار استاندارد برای نمایش دادن دیالوگ های بین کامپیوتر و انسان است.استاندارد VoiceXML یک specification را برای رابط میان لایه وابسته به برنامه و یک مرورگر وب تعریف می کند .لایه وابسته به برنامه در برنامه VoiceXML بسیار سبک است و توسط یک مجموعه از اسناد XML نمایش داده می شود.

یک مرورگر وب یک موتور مستقل از برنامه است که دیالوگ ها را بر طبق اسناد VoiceXML داده شده پیاده سازی می کند[۱۳].

برای عملیات تشخیص گفتار نیز یک استاندارد توسط کنسرسیوم وب جهانی به نام گرامر تشخیص گفتار ارائه شده است که در مرور گر های صوتی برای درک و تشخیص کلمات بیان شده استفاده می شود. همچنین استاندارد تولید گفتار متن در گام تبدیل متن به گفتار استفاده می شود . کنسرسیوم وب جهانی همچنین برای درک و فهم معانی گفتار تولید شده از مفسر معنایی تشخیص گفتار استفاده می کند .

در دو فصل آتی به معرفی استانداردهای زبان علامت‌دار صوتی (VXML) و زبان علامت‌دار تولید گفتار (SSML) می پردازیم و چگونگی استفاده از آن ها را خواهیم دید.

فصل 3

زبان علامت‌دار صوتی

VXML

۳-۱. مقدمه

VXML از سال ۱۹۹۵ به عنوان یک زبان مبتنی بر XML مطرح شد. یک زبان علامتگذاری بر پایه وب برای به نمایش در آوردن دیالوگ‌های بین کامپیوتر و انسان با به کار بردن تجهیزات ورودی و خروجی صوتی که جهت به دست آوردن محتوای اینترنت و دسترسی به اطلاعات از طریق صوت و تلفن طراحی شده است.

VXML برای ایجاد دیالوگ‌های صوتی که بر اساس گفتار ترکیبی، صدای دیجیتالی، تشخیص صدا، ورودی کلید با DTFM[1]، ضبط صدای ورودی و تلفنی می باشند، طراحی شده است. هدف اصلی از ارائه آن رساندن توسعه نرم افزارهای بر پایه وب به نرم افزارهای پاسخگوی صوتی می باشد[۱۴]. برای تشخیص گفتار از گرامر تشخیص گفتار و برای تولید گفتار متن از استاندارد تولید گفتار متن در اسناد VXML استفاده می‌شود.

در این فصل به بررسی استاندارد VXML می پردازیم و مزایای استفاده از آن را بررسی می کنیم

۳-۲. VoiceXML

اکثر وزارتخانه ها و اداره های عمومی سرویس های بر پایه وب مشتری ها را ترجیح می دهند ، مزایای استفاده از این سرویس ها بسیار محسوس است . به هر حال تعداد زیادی از مشتریان استفاده از تلفن را ترجیح می دهند. Call center ها یک راه حل مناسب برای این منظور هستند ولی هزینه عامل های انسانی در آن ها بسیار زیاد است. به نظر می رسد که یک راه حل بهتر برای مجتمع کردن کانال های سرویس فوق و عامل‌های انسانی، VXML باشد.

سیستم‌های دیالوگ یک دستاورد مناسب برای این سرویس‌ها هستند که می توانند کارآمدتر از سیستم‌های جاری با کمبودهای اجرایی باشند. به‌علاوه، صوت قابلیت استفاده از وب را افزایش می دهد ،

---

[1] Dual-Tone Multi-Frequency

مخصوصا برای افرادی با معلولیت های مختلف (دیداری ، فیزیکی) . مزیت های دیگر مد زبان در ادامه آمده است :

- استفاده از آنها ساده و راحت است .
- انواع مختلف ارتباط را ساپرت می کنند (منوها ، پر کردن فرم ها و فرمان ها)
- می توانند به سوالات پیچیده که پاسخ آن ها ممکن است که نیاز به پردازش اطلاعات از منابع مختلف داشته باشد جواب دهند.
- با انواع دیگر حالت ها غیر صوت می توانند کامل و ترکیب شوند .
- برای انواع مختلف کاربران (زبان های مختلف ، مهارت ها ، سن ها و حساسیت های فرهنگی مختلف) قابلیت تطبیق دارند.

تشخیص گفتار در دامنه با ابعاد زیاد دارای مشکلات اجرایی است . به این دلیل ، مد گفتار به طور اساسی در سیستم های با دامنه محدود استفاده می شود . اگرچه توسعه پذیری این سیستم های محدود بسیار گران و تطبیق پذیری آن ها برای دامنه های جدید بسیار سخت است .

تعریف زبان استاندارد VoiceXML (یک زبان علامت گذاری بر پایه XML به طور ضمنی توسعه برنامه های تلفنی بر پایه اینترنت را اختصاصی می کند) هزینه توسعه سیستم های دیالوگ را کاهش می دهد و تکنولوژی های مجتمع سازی را آسان می سازد .[۱۵]

در زمان بحث راجع به پرتال های صدا ممکن است که در استانداردهای پیشنهاد شده که مشابه VoiceXML هستند دچار گمراهی شویم . تفاوت گذاشتن بین VoiceXML و دو استاندارد دیگر که در حال رقابت با VoiceXML هستند SALT و XHTML+Voice profile خیلی مهم است . VoiceXML اساسا برای پذیرفتن دیالوگ های صدا که شامل تشخیص صدا ، تبدیل متن به گفتار و ضبط و پخش صدا می باشد ، DTMF یا ورودی لمسی و تلفن طراحی شده .

دیگر استاندارد هایی که بسیار نزدیک به VoiceXML هستند استانداردهایی هستند که توسط W3C تحت کالبد رابط گفتار توسعه پیدا کرده اند[۳].

۳-۳. معماری VXML

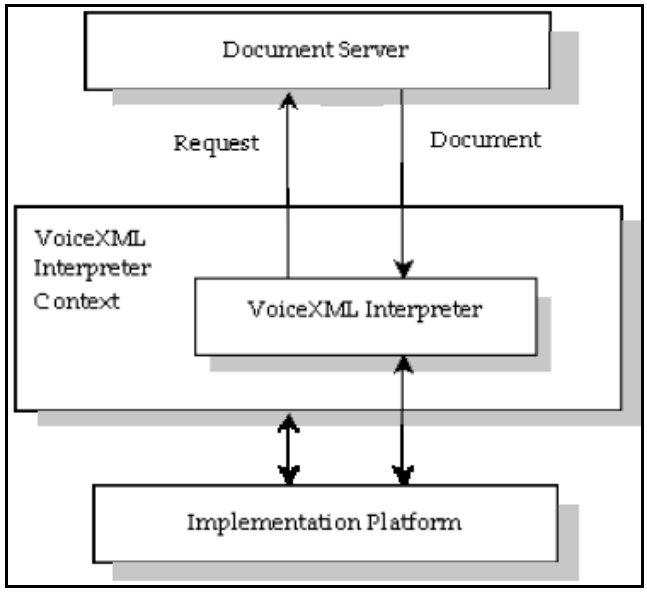
معماری VXML در شکل ۳-۱ نشان داده شده و شامل اجزای زیر است :

سرور اسناد : یک Document Server برای مثال یک سرور وب که وظیفه آن پردازش تقاضاها از برنامه مشتری است .

مفسر VXML : که اسناد VXML را پردازش می کند و دیالوگ ها را هدایت و اداره می کند.

مفسر زمینه VXML : وظیفه آن بدست آوردن اسناد VXML ، تشخیص و پاسخ به تماس هاست.

بستر پیاده سازی : که توسط مفسر زمینه VXML و مفسر VXML که اعمالی را در پاسخ به رفتار کاربر (برای مثال دریافت کاراکتر های ورودی و قطع تماس) و همچنین رخدادهای سیستم (برای مثال انقضا وقت) تولید می کند[۱۶].



شکل3-1. مدل معماری VXML

۳-۴. اهداف VXML

VXML زبانی است که فعل و انفعالات بین سرویس گیرنده و سرویس دهنده را به حداقل می رساند. برنامه-نویس را از جزئیات سخت افزاری و سطح پائین دور می کند. فعل و انفعال کاربر را از منطق سرویس جدا می کند. قابلیت انتقال در سخت افزارهای مختلف را می دهد. پیاده‌سازی دیالوگ های ساده و پیچیده فراهم می باشد[۱۷].

۳-۵. محدوده VXML

زبانی است که فعل و انفعال متقابل انسان و ماشین را با سیستم های پاسخگویی صوتی توضیح می دهد که شامل موارد زیر است:

- تبدیل متن به صدا به عنوان خروجی
- پخش فایل های صوتی به عنوان خروجی
- تشخیص صدای ورودی
- تشخیص کدهای DTMF به عنوان ورودی
- ضبط صدای ورودی
- کنترل جریان دیالوگ ها
- پشتیبانی از ویژگی های تلفنی مانند انتقال مکالمه و قطع مکالمه[۱۶].

۳-۶. مفهوم

از انتشار voicexml 1.0 به وسیله انجمن voicexml در جولای ۲۰۰۰ توسعه دهنده ها بسیاری از

مرورگرهای voicexml را پیاده سازی کردند . voicexml 1.0 توسعه دهنده های برنامه های کاربردی را قادر به تولید و گسترش دادن هزاران برنامه کاربردی گفتار مکالمه ای می کنند.

در می ۱۹۹۹ کنسرسیوم وب جهانی W3C به گروه مرورگر صدا (Voice Browser Working Group) امتیاز آماده کردن و تجدید نظر کردن زبان های علامت دار که مرورگرهای صدا را فعال می ساختند، داد[۱۸].

یک سند VoiceXML یک ماشین حالت مکالمه متناهی را فرم می دهد ، کاربر در یک زمان در یک حالت مکالمه و دیالوگ است . عنصر سطح بالا <vxml> نام دارد که اساسا در بر گیرنده دیالوگ هاست .یک سند VXML شامل انواع مختلف عناصر است.

اجرای سند به طور پیش فرض با دیالوگ اول شروع خواهد شد ، هر دیالوگ ، دیالوگ بعدی را برای انتقال مشخص می کند. انتقال ها توسط URI[1] ها که سند و دیالوگ بعدی را برای استفاده نشان می دهد مشخص می شوند .اگر یک URI به یک سند اشاره نداشته باشد سند جاری به عنوان سند فرض می شود و اگر به یک دیالوگ اشاره نکند اولین دیالوگ سند در نظر گرفته می شود .اجرا زمانی که دیالوگ ، یک دیالوگ جایگزین را مشخص نکند و یا با استفده از یک عنصر از مکالمه خارج شود خاتمه پیدا می کند[۱۶].

۳-۶-۱. دیالوگ و زیر دیالوگ

دو نوع دیالوگ در VXML وجود دارد فرم ها و منوها. فرم یک فعل و انفعال که مقادیری را برای مجموعه ای از متغیرها جمع آوری می کند ، می باشد. منو لیستی از انتخاب ها را به کاربر ارائه می کند و بر اساس انتخاب کاربر به دیالوگ دیگری می رود .

[1] Uniform Resource Identifier

زیر دیالوگ مانند یک فراخوانی تابع می باشد که در آن مکانیزم درگیر شدن در یک فعل و انفعال دیگر و سپس بازگشت به فرم اصلی قرار می گیرد.

۳-۶-۱-۱. فرم ها

فرم ها جزء کلیدی صفحات VoiceXML می باشند که شامل :

- مجموعه ای از قلم ها که در حلقه اصلی الگوریتم تفسیر فرم قرار دارند . این قلم ها به دو دسته تقسیم می شوند : دسته اول قلم های فیلد که متغیرهای قلم فیلد را تعریف می کنند و دسته دوم قلم های کنترل که به جمع آوری فیلد های فرم کمک می کند .
- اعلان متغیرهایی غیر از قلم های فرم
- توابع رخداد ها
- عمل "Filled" و مجموعه ای از رویه ها که هنگامی اجرا می شوند که متغیرها مقدار خاصی پیدا کنند.

در اینجا یک مثال کوتاه از زبان VoiceXML می بینیم :

```
<?xml version=”1.0”?>
<vxml version=”2.0”>
<form>
<block>Hello World!</block>
</form>
</vxml>
```

VXML یک استاندارد مستقل از زبان است.

این مثال یک فرم دارد که شامل یک بلوک است که تولید گفتار می شود و "Hello World" را به کاربر می‌گوید .از آنجایی که این فرم یک فرم جانشین بعدی پیدا نمی کند، مکالمه خاتمه پیدا می کنـد . در ایـن مثال از عناصر <vxml> ، <form> و <block> استفاده کرده ایم[۱۶].

W3C علاوه بر VoiceXML، سه استاندارد برای تفسیر معنایی SISR[1] ، گرامر تشخیص گفتار SRGS[2] ، تبدیل متن به گفتار SSML[3] منتشر کرده است، که در فصول بعد به توضیح آن ها خواهیم پرداخت.

در شکل زیریک مثال از چگونگی کارکرد چهار زبان با هم برای توضیح دادن رابطهای گفتار مکالمـه را نشان می دهد:

```
<field name="recipient">
<prompt>
Welcome to the
<emphasis level = "strong">
electronic payment system
</emphasis>
<break time = "500ms"/>
Whom do you want to pay?
</prompt>
<grammar>
<rule id= "recipient">
<one-of>
<item> Ajax </item>
<item>
```

[1] Semantic Interpretation for Speech Recognition

[2] Speech Recognition Grammar Specification

[3] Speech Synthesis Markup Language

<tag> 'Drugstore' </tag> **Ajax**
</item>
<item> Stanley </item>
<item>
<tag> 'Supermarket' </tag> **Stanley**
</item>
</one-of>
</rule>
</grammar>
</field>

فونت معمولی کد های voicexml و فونت ایتالیک تبدیل متن به گفتار و بولد گرامر گفتار و متن هایی که زیر آنها خط کشیده شده تگ های تفسیر معنایی را نشان می دهند.

تگ <prompt> شامل متنی می شود که توسط موتور تبدیل متن به گفتار به گفتار تبدیل می شود . تگ گرامر یک گرامر را از کلمات و عباراتی که توسط موتور تشخیص گفتار استفاده می شود را برای تبدیل گفتار به متن تعریف می کند .

تگ های <prompt> و گرامر <grammar> با همدیگر یک <field> که در آن کاربر یک مقداری را باگفتار قرار می دهد تعریف می کند[۱۸].

۳-۷. کاربردهایی برای برنامه های VoiceXML

انسان ها با سیستم های پاسخگویی صوتی دارای فعل و انفعال تجاری مانوس و در کار با آن ها وارد هستند این سیستم ها معمولا در دستیابی صوتی mail ، کنفرانس های تلفنی وجود دارند .

VoiceXML یک زبان است که سیستم های IVR محدود و خصوصی تجاری را به یک معماری برنامه-

ریزی بزرگ تبدیل می کند و همچنین مزایای تکنولوژی های وب را با تامین کردن دیالوگ های برنامه ریزی شده، که مشابه با فرم های HTML هستند برای یک کاربر تلفنی فراهم می کند[۱۹]. برخی از کاربردهای مختلف آن عبارتند از:

- ابزار یادگیری زبان : استفاده از یک gateway برای آموزش زبان به افراد بدون نیاز به معلم
- دیکشنری لغات: یک دیکشنری تلفنی ، کاربر می تواند کلمه ای را که معنی آن را می خواهد بگوید و یا هجی کند و در جواب معنی کلمه را بشنود .
- Caller ID: هنگامی که یک تماس گرفته می شود، کاربر می تواند تلفن را بردارد و از طریق VoiceXML اسم تماس گیرنده به او داده می شود و کاربر می تواند تلفن را جواب دهد و یا اینکه آن را به voice mail منتقل می کنیم.
- Call Waiting: وقتی که شخص A با شخص B صحبت می کند و شخص C با شخص A تماس می گیرد اسکریپت های VoiceXML در تماس وقفه می اندازند و به شخص A اطلاع می دهند که C تماس گرفته است.
- Call Center: یک خط تامین مشتری که اجازه می دهد که کاربران با سیستم اتوماتیک VoiceXML به محاوره بپردازند. زمانی کاربر برای صحبت با یک اپراتور انسانی منتظر می ماند در این مدت می تواند با استفاده از VoiceXML یکسری اطلاعات بگیرد و یا اینکه پیغام بگذارد.
- Banking: معاملات بانکی از طریق صوت مثل چک کردن اطلاعات حساب شخصی و انتقال پول از یک حساب به حساب دیگر.
- اطلاعات درسی برای دانشجویان دانشگاه: دانشجویان می توانند شماره و نام درس را انتخاب کنند و یک توصیفی از درس مربوطه بگیرند .

- مشاور پزشکی: کاربر لیست علائم بیماری خودش را وارد می کند وسیستم برای تشخیص بیماری پردازش انجام می دهد و کاربر را به آنچه که باید انجام راهنمایی می کند .[۱۶]

از آنجایی که پروتکل MRCP پیاده سازی بستر های IVR را با استفاده از مرورگرهای VXML امکانپذیر می سازد و در این گزارش زیاد به آن اشاره شده است، در ادامه آن را معرفی می‌کنیم.

۳-۸. پروتکل MRCP

این پروتکل به نحوی طراحی شده است که به ابزار client اجازه کنترل منابع پردازش مدیای تحت شبکه را می‌دهد. بعضی از این منابع پردازش مدیا شامل موتورهای بازشناسی صوت، موتورهای تولید گفتار صوت، موتورهای تائید صحبت کننده و موتورهای تشخیص صحبت کننده می شوند.

این پروتکل پیاده سازی بسترهای IVR را با استفاده از مرورگرهای VXML امکانپذیر می سازد. از طرف دیگر قابلیت های پردازش صوت مستقل در دو طرف در سرورهای پردازش صوت خاص را نیز حفظ می‌کند .

۳-۸-۱. انواع منابع و سرویس ها در MRCP

منبع مدیا یک قسمت از سرور پردازش صوت است که می توند از طریق پروتکل MRCP کنترل شود. سرور MRCP مجموعه یک یا چند منبع مدیا بر روی یک سرور است. سرور تنظیم می شود تا مجموعه ای مشخص از منابع و سرویس ها را به client پیشنهاد دهد . این تابع سه نوع می باشند :

- منابع ارسال: این ها منابعی هستند که قادر به تولید جریان های بلادرنگ هستند مثل تولید کننده گفتار های گفتار که جریان های صوت بیان شده را تولید می کنند.
- منابع دریافت: این ها منابعی هستند که داده های جریانی را دریافت و پردازش می کنند مثل تشخیص دهنده های گفتار .
- منابع دو کاره: منابعی که هم ارسال و هم دریافت داده ها را انجام می دهند[۳۰].

در فصل بعد زبان علامت‌دار تولید گفتار را بررسی خواهیم کرد و عناصر مختلف آن را شرح می دهیم.

فصل 4

زبان علامت‌دار تولید گفتار

SSML

۴-۱. مقدمه

سیستم Text to Speech سیستمی است که یک دنباله از کلمات رابه عنوان ورودی می گیرد و در خروجی آن را به گفتار تبدیل می کند. هدف کاربردی از یک موتور TTS[1] تولید کردن گفتار به عنوان خروجی از روی متن یا یک سند XML است. TTS برای بسیاری از برنامه های کاربردی که از صدا به عنوان خروجی استفاده می کنند کاربرد دارد.

زبان علامت گذاری تبدیل متن به گفتار یک روش استاندارد جدید برای تولید مواردیست که باید توسط سیستم تبدیل متن به گفتار بیان شود. SSML[2] یک استاندارد جدید تولید صدا توسط سیستم تبدیل متن به گفتار است که توسط W3C ارائه شده است. یک زبان علامت گذاری بر پایه XML که برای کنترل فرآیند تبدیل متن به گفتار در زمینه های متفاوت برنامه های کاربردی هوشمند شده است.

TTS تنها برای کاربردهایی که افراد با نداشتن امکانات بصری برای خواندن متن و ارتباط با کامپیوتر مشکل دارند نیست، بلکه همچنین برای بسیاری از کاربردها که از خروجی صدا استفاده می کنند کاربرد دارد. برای مثال اعلام خطر(مثلا در ایستگاه راه آهن)، سرویس‌های تلفنی متقابل، سرویس‌های اطلاعاتی (پیش بینی آب و هوا، شرایط ترافیک، اخبار، ...)، بانک ، پست الکترونیک ، خواندن فاکس ، بازی‌ها و غیره.

TTS به منظور تغییر پارامترهایی برای بهبود دادن و بهینه سازی صدا کنترل می شود، SSML در کنترل جزئیاتی مانند این که چطور یک متن، یک پاراگراف، یک جمله و یا یک کلمه ادا شود، کمک می-کند[۲۲].

VoiceXML که باعث پیشرفت برنامه های کاربردی گفتار بر پایه وب است، برای بیان پیام‌ها SSML

[1] Text - to - Speech

[2] Speech Synthesis Markup Language

در برنامه های گفتاری از SSML استفاده می کند .

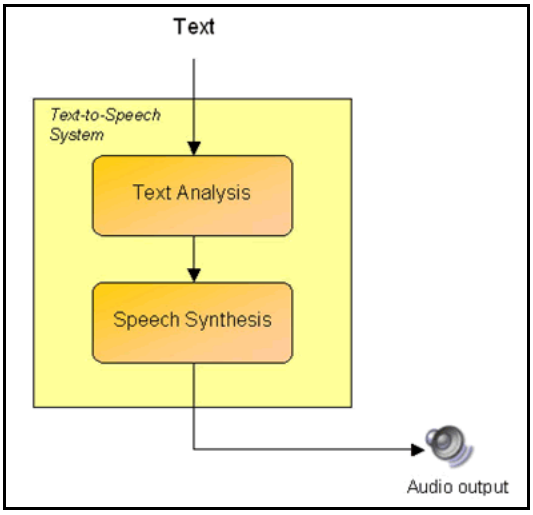
SSML برای فرمان ها در Synchronized Multimedia Integration Language(SNIL) و در پروتکل فعل و انفعال متقابل MRCP، تعیین شده است[۲۲].

۴-۲. زبان علامت دار تبدیل متن به گفتار

SSML یک زبان علامت گذاری برای تولید فرمان ها با استفاده از ترکیب موارد ضبط شده، تبدیل متن به گفتار و موزیک است. کنترل کردن خصوصیات صدای تولید گفتار شده مثل جنس مخاطب (مرد یا زن بودن)، سن، سرعت، شدت، پیچ و تاکید امکان پذیر می باشد . همچنین کنترل خصوصیات تلفظ یک کلمه برای موتور تبدیل متن به گفتار امکان پذیر می باشد[۳].

۴-۲-۱. درون یک موتور تبدیل متن به گفتار

اگر چه سیستم های مختلف ممکن است که الگوریتم های متفاوتی را برای دستیابی به سیگنال خروجی صحیح استفاده کنند، پردازش تولید گفتار عمومی چنان‌که پیشتر اشاره شد، مراحلی همچون شکل۴-۱ دارد.



شکل4-1. سیستم تبدیل متن به گفتار عمومی

گام تحلیل متن: هدف آن تبدیل یک متن ورودی به جزئیات بیشتر و توضیحاتی است که چه چیز باید تلفظ شود، برای تعیین دقیق خصوصیات صداها که باید تولید گفتار شوند باید ابهام را از بین ببریم، نقطه گذاریها را تفسیر کنیم و علائم و اختصارها را حل کنیم. زمانی‌که مشخص شد که چه چیز باید تلفظ شود مرحله دوم شروع می شود.

گام تبدیل متن به گفتار: سیگنال گفتار واقعی را تولید می کند، در این مرحله تکنیک های متفاوتی ممکن است به کار برده شود، مهمترین راه حل در بدست آوردن گفتاری با کیفیت بالا، تکنیک تولید گفتار الحاق واحدها ست، که بر پایه یک پایگاه داده از نمونه های گفتاری انسانی است که برای انطباق متن ورودی قسمتهای مناسب از آن استخراج و ترکیب می شود .

در مرحله اول برای بدست آوردن توضیحات دقیق از تلفظ متن ورودی، کارهای زیر توسط موتور TTS انجام می شود:

- **نرمال سازی متن**: با توجه به فضاهای خالی و نقاط ، متن ورودی باید به جملات و کلمات تقسیم

شود. اعداد، علامات، اختصارها باید به فرم نرمال تبدیل شوند. برای مثال جمله " Dr. John Smith lives at Jefferson dr. 94، Paris، Texas " در مرحله نرمال سازی به جمله " Doctor John Smith lives at Jefferson drive ninety-four {pause} Paris {pause} Texas {pause}" تبدیل خواهد شد .

- **تلفظ کلمه**: هر کلمه به نمایش فونتیکی آن تبدیل می شود، قوانین و فرهنگ لغات برای این هدف طراحی شده اند. در بعضی از زبانها تبدیل حرف به صدا یک فرایند مستقیم است، در حالی که در دیگر زبان ها این فرایند می تواند غیر قابل پیش گویی باشد مثل زبان انگلیسی.
- **ساختار جمله**: تلفظ و ریتم و در بعضی موارد استرس و فونتیک وابسته به نقش کلمات در جمله است.به منظور تشخیص مکان صحیح مکث و ریتم و آهنگ صدا معمولا در موتور تبدیل متن به گفتار قوانین پروزودیک پیاده سازی می شوند .
- **اصلاح سطح جمله در تبدیل فونتیکی**: فونتیک کلمات باید در جمله به هم متصل شوند و این نیازمند یکسری اصلاحات در تبدیل فونتیکی با توجه به تغییرات گفتار پیوسته است .
- **محاسبه پارامترهای پروزودیک**: تحلیل یک متن در دنباله ای از فونم ها وقتی توضیحات پروزودیک داده می شود کامل می شود، ریتمی که به دنباله ای از فونم ها نسبت داده می شود توسط مقادیر عددی یا توسط برچسب های پروزودیک مشخص می شود. تعیین مقادیر پروزودیک می تواند بر اساس قوانین باشد، یا بوسیله سیستم یادگیری اتوماتیک اجرا شود.

SSML تمام مراحل تحلیل متن را به منظور تامین یک روش استاندارد برای کنترل جنبه های گفتار مثل تلفظ، توسعه علامات اختصاری، اندازه صدا، پیچ، محدوده[1]، تاکید و غیرهدایت می کند[۲۲].

[1] Range

۳-۴. عناصر در SSML

SSML مانند دیگر زبانهای بر پایه XML، ترکیبی از عناصر است. تمام عناصر می توانند دارای صفت باشند. SSML دارای یک عنصر <speak> است که شامل متنی است که باید بیان شود و شامل دو صفت زبان (زبان متنی است که باید گفته شود) است.

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
  <voice name="Dave">
   Hello، world; my name is Dave.
  </voice>
</speak>
```

در این مثال صوتی به نام Dave باید جمله"Hello، world; my name is Dave " را تلفظ کند . عنصر <meta> و <meta data> توضیحاتی راجع به سند ارائه می دهند(نویسنده، تاریخ و ...).

به عنوان مثال در گرامر زیر برای تبدیل علامات مختصر که در پیام های SMS به کار می روند از Lexicon مربوط به آن استفاده شده است:

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
   <lexicon uri="file://localhost/share/SMSLexicon.lex"/>
     <!-- This is the text of a SMS to be read using a
        lexicon file. -->
     I love U :-)
</speak>
```

عنصر اصلی برای بیان ساختار متن <p> است که پاراگراف را نشان می دهد و<s> که بیانگر یک جمله است.

یکی دیگر از عناصر <phoneme> است که با صفت <ph> تلفظ را برای متن نشان دهد. انتخاب الفبای فونتیکی متفاوت برای استفاده با صفت phoneme alphabet وجود دارد (الفبای IPA، یک الفبای ضروری است ولی موتورهای متفاوت ممکن است دیگر الفباها را پشتیبانی کنند)[۲۲].

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
   Tipical italian dishes are
   <phoneme alphabet="ipa" ph="&#112;&#712;&#105;&#720;&#116;&#794;&#115;&#601;">
    pizza
   </phoneme>
   <-- "pizza" will be pronunced as: p  i ː t͡sə -->
   and
   <phoneme alphabet="ipa" ph="&#115;&#112;&#601;&#103;&#712;&#101;&#116;&#812;&#105;">
    spaghetti.
   </phoneme>
   <-- "spaghetti" will be pronunced as: spəg  etˌi -->
</speak>
```

برای بسط دادن علائم اختصار عنصر <sub> استفاده می شود، و با صفت alias مقداری را که باید به جای آن علامت مختصر تلفظ شود، نشان می دهیم.

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
   The <sub alias="Speech Synthesis Markup Language">SSML</sub>
   <sub alias="version">v.</sub> 1.0 is a new standard of
   <sub alias="World Wide Web Consortium">W3C</sub>.
</speak>
```

به منظور کنترل پروزودی و مدل از عنصر <voice> و صفتهای آن XML: lang و جنسیت و گوناگونی و سن و نام استفاده می کنیم، یا عناصر <emphasis> و <break> به منظور افزایش و کاهش تاکید در جمله و قرار دادن و حذف کردن مکث.

```
<?xml version="1.0"?>
```

```
<speak version="1.0" xml:lang="en">

  <voice gender="male" variant="1">
   Elsinore. A platform before the Castle.
  </voice>

  <voice gender="male" variant="2">
   <emphasis>Who's there?</emphasis>
  </voice>

  <voice gender="male" variant="3">
   Nay، answer me: stand، and unfold yourself.
  </voice>

  <voice gender="male" variant="2">
   <emphasis level="strong"> Long live the king! </emphasis>
  </voice>

  <voice gender="male" variant="3">
   Bernardo?
  </voice>

  <voice gender="male" variant="2">
   He.
  </voice>

  <voice gender="male" variant="3">
   You come most carefully upon your hour.
  </voice>
```

<voice gender="male" variant="2">
'Tis now struck twelve. Get thee to bed، Francisco.
</voice>

<voice gender="male" variant="3">
For this relief much thanks: 'tis bitter cold،
And I am sick at heart.
</voice>

<voice gender="male" variant="2">
Have you had quiet guard?
</voice>

<voice gender="male" variant="3">
Not a mouse stirring.
</voice>

<voice gender="male" variant="2">
Well، good night.
<break/> If you do meet Horatio and Marcellus،
The rivals of my watch، bid them make haste.
</voice>

<voice gender="male" variant="3">
I think I hear them. Stand، ho! Who is there?
</voice>
</speak>

با عنصر <prosody> و صفتهای pitch و contour وrange و rate و duration و volume می توانیم تلفظ را کنترل کنیم.

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-GB">
   Using prosody element it is possible
   <prosody rate ="x-fast" >
    read fast a sentence
   </prosody>
   or
   <prosody pitch ="+60Hz">
    manipulate the voice pith
   </prosody>
   and other important parameters.
</speak>
```

SSML همچنین کنترل هایی را برای گام تبدیل متن به گفتار فراهم می کند .با عنصر <voice> می توان یک صوت مشخص را انتخاب کرد. عنصر <audio> یک فایل صوتی را در متن تولید گفتار شده قرار می دهد. در دو فصل آینده ابتدا به کاربردهای موتورهای تولید گفتار در محصولات مایکروسافت و سپس به بررسی امکانات تولید گفتار موجود در بستر جاوا پرداخته خواهد شد.

فصل 5

کاربرد سیستم‌های تولید گفتار در

محصولات مایکروسافت

۵-۱. Microsoft Speech Server 2007

OCS 2007 Speech Server یا Office Communication Server 2007، یک جزء اختیاری از OCS است که بعد از Live Communication Server آمده است. جزء Speech Server، بعد از Microsoft Speech Server 2004 (که یک محصول stand-alone است) آمده است.

جزء Speech Server می تواند بطور جداگانه از OCS از طریق واسط نصب، نصب شود. ابتدا ایجاد برنامه‌های کاربردی تلفنی را با استفاده از Speech Server بطور جداگانه توضیح داده می‌شود و سپس ادغام این دو محصول شرح داده می‌شود.

۵-۱-۱. دید کلی از OCS 2007 Speech Server

Speech Server یک بستر IVR است که با Visual Studio 2005 مجتمع می شود. Speech Server ابزارهایی برای توسعه برنامه هاب کاربردی که روی تلفن اجرا می شوند یا برنامه های کاربردی تلفنی، (telephony application)، فراهم می کند. برای مثال، برنامه های کاربردی تلفنی به کاربر اجازه می دهند که حساب بانکی خود را ازطریق تلفن چک کند یا یک تماس خودکار از مطب پزشک فرد، نوبت بعدی او را یادآوری کند. Speech Server، برای یک برنامه کاربردی تلفنی است همانطور که Web Server مثل IIS یا Internet Information Server) برای یک برنامه کاربردی وب است.

برنامه های کاربردی Speech Server، می توانند قابلیت های زیر را داشته باشند:

- Speech recognition: که به کاربران اجازه می دهد تا به prompt های برنامه پاسخ دهند.
- قابلیت‌های Touch-tone یا DTMF: که به کاربران اجازه می دهد تا به prompt های برنامه از طریق صفحه کلید پاسخ دهند.
- Text-to-Speech یا TTS: که به برنامه ها اجازه می دهد تا متن نوشته شده را برای کاربران

بخواند یا صحبت کند. یکی از خصوصیات Speech Server، پشتیبانی ملی برای VoIP است. Speech Server می تواند تماس های VoIP را بدون هرگونه سخت افزار یا نرم افزار اضافی، بپذیرد. Speech Server ، سه نوع پروژه را پشتیبانی می کند:

۱) SALT یا Speech Application Language Tags: که یک زبان استاندارد w3c برای speech روی وب و تلفن است و روی صفحات ASP.net اجرا می شود.
۲) VoiceXML: همان کار SALT را انجام می دهد و روی صفحات ASP.net اجرا می‌شود. می توان برنامه IVR بر پایه VoiceXML را بدون تغییرات زیاد در کدهای آن، به Speech Server تبدیل کرد.
۳) Voice Response Workflow: اگر الگو[1]ی وب را نخواهیم دنبال کنیم، انتخاب دیگر، همان Voice Response Workflow می باشد. این نوع پروژه، بر اساس Windows Workflow Foundation یا WF است. بر خلاف SALT و VoiceXML، Voice Response Workflow به کاربر اجازه می دهد تا جریان تماس‌های برنامه‌اش را ببینید.

برای تولید یک برنامه جدید IVR ، Voice Response Workflow اکیدا نسبت به SALT و VoiceXML توصیه می شود، ار آنجایی که برنامه های IVR معمولا یک Workflow (جریان کار) را دنبال می کنند. برای ادغام یک صفحه SALT یا VoiceXML درون یک برنامه Voice Response Workflow، می توان این کار را با استفاده از کنترل‌های مفسر SALT و VoiceXML انجام داد.

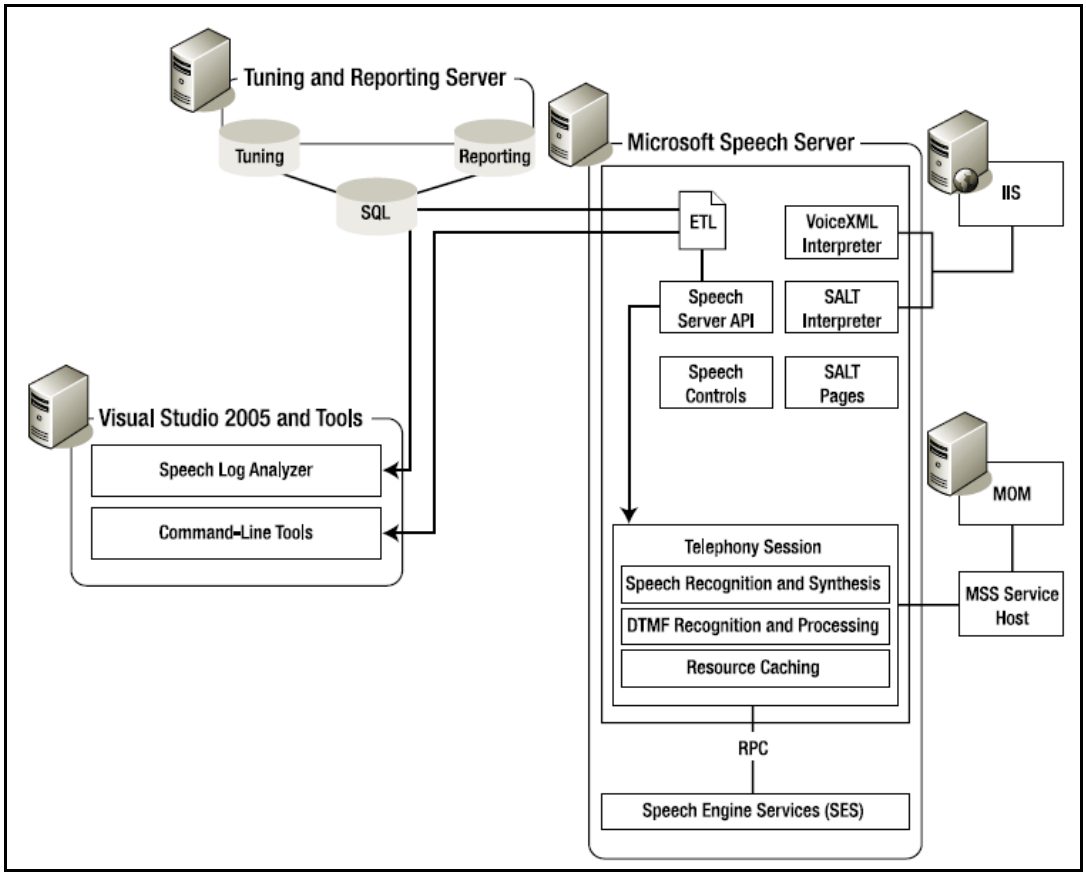
۵-۱-۲. طرز کار Speech Server:

Speech Server دو جزء اساسی دارد:

- Speech Engine Services
- ASP.net

[1] Paradigm

در حالیکه IIS بطور تکنیکی یک قسمت از Speech Server نیست، ولی نقش اساسی در معماری آن ایفا می‌کند. شکل۵-۱، نمایی کلی و مفهومی از این که Speech Server چگونه با بقیه سرورها و با Visual Studio کار می‌کند، را نشان می دهد.



شکل5-1. نمای کلی چگونگی کار کردن Speech Server با بقیه سرورها

۵-۲. تشخیص گفتار و تبدیل متن به گفتار

Speech Engine Services یا SES دو جزء اصلی دارد:

- Speech Recognizer Engine
- Speech Synthesis Engine

Speech Recognizer Engine، یک سند XML به SML یا Semantic Markup Language تولید می‌کند.

این سند SML شامل کلمات یا اصطلاحاتی است که Speech Recognizer Engine می تواند بفهمد و یک مقدار عددی (معیار اطمینان) برای مطمئن بودن از اینکه کاربر این کلمه یا اصطلاح را ، بر اساس گرامر تعریف شده توسط برنامه کاربردی گفته است.

Speech Synthesis Engine، که بعنوان TTS engine نیز نامیده می شود، صدا را بر اساس یک سند XML به زبان SSML یا Speech Synthesis Markup Language تولید می کند. سند SSML شامل متنی است که Speech Synthesis Engine خواهد گفت. قبل از اینکه Speech Synthesis Engine صحبت کند، prompt engine را بررسی می کند. اگر prompt engine شامل کلمه یا اصطلاحی که Speech Synthesis Engine باید بگوید، باشد، prompt از پیش ضبط شده associated را به جای استفاده از موتور TTS، بیان می کند.

۵-۳. اتصالات مشتری[1]

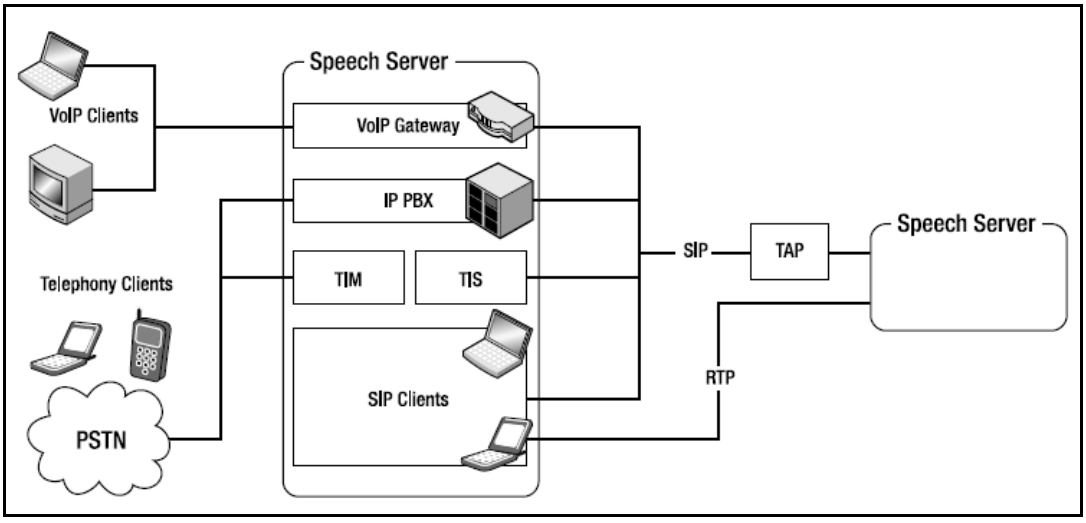
همانطور که قبلا گفته شد، Speech Server، VoIP را برای تماس های تلفنی روی اینترنت، پشتیبانی می‌کند. پروتکلی که زیر آن خوابیده، پشتیبانی ملی VoIP برای Speech Server فراهم می‌کند، SIP یا Session Initiation Protocol می‌باشد. برای انتقالات، ابتدا یک مشتری، مثل Office Communicator، یک سرور SIP (اینجا همان OCS 2007)است که داده قطعی سرور SIP را می‌دهد (مثل موقعیتش، آدرس IP، و پروتکلی که پشتیبانی می کند). وقتی یک کاربر می خواهد یک تماس به یک کاربر ثبت شده برقرار کند، یک پیغام INVITE، به کاربر ثبت شده فرستاده می شود، بر اساس داده ای که مشتری به سرور SIP در هنگام ثبت شدن داده است. همین که مشتری، INVITE را پذیرفت، دو تا مشتری می توانند ارسال و دریافت (ارتباط) داشته باشند. دلیل استفاده از INVITE این است که مشتری می تواند در چندین موقعیت ثبت شود، مثل یک دستگاه موبایل یا یک کامپیوتر. موقعیت اول که INVITE را می پذیرد، پیغام های بعدی را ارسال می کند. اگر

[1] Client Connections

شبکه مورد نظر، VoIP را پشتیبانی کند،باید آماده‌سازی[1]های اضافه ای (مثل حالتی که از Speech Server 2004 استفاده می شد) بکار ببرد. یک کارت تلفن و نرم افزار Telephone Interface Manager یا TIM. Speech Server، یک مطابقت backward از طریق Telephony Interface Service یا TIS، که بعنوان یک واسط بین TIM شما و Telephony Application Proxy یا TAP سرویس می دهد، فراهم می کند.

TAP خیلی شبیه یک SIP redirect Server، که درخواست ها را از/به TIS و سایر SIP Peer ها، تفسیر می‌کند و آنها را به سرور مناسب مسیریابی می کند، عمل می کند.

شکل۵-۲، یک نمای کلی مفهومی از چگونگی کار کردن Speech Server با اتصالات مشتری‌ها نشان می دهد.



شکل5-2. نمای کلی مفهومی از چگونگی کار کردن Speech Server با اتصالات مشتری‌ها

۵-۴. زبان و گرامر در OCS 2007

وقتی نصب اجزا در ocs پایان می یابد ، باید language pack نصب شود . packهای زبان روی CD نصب موجود هستند در دایرکتوری language pack در حال حاضر زبان های زیر را پشتیبانی می کند:

[1] setup

- English (us)
- English (uk)
- German
- French
- Spanish

حداقل us pack باید نصب شود اگر برنامه مورد نظر چندین زبان را لازم است پشتیبانی کند، باید بقیه packها نیز نصب شود.

یک برنامه که گفتار را تشخیص می دهد، نیاز دارد بداند کدام کلمات و اصطلاحات را می تواند قبول کند. برای این کار به گرامر احتیاج است. ساختن گرامر، شامل کامپایل کردن لیست های پاسخ های پیش بینی شده کاربر به اعلان های خاص است.

ایجاد گرامر، لزوما نباید اولین مرحله در ایجاد یک برنامه IVR باشد، اما مطمئنا لازم است که سریعا در پروسه توسعه برنامه، انجام شود. گرامر در یک فرمت w3c-compliant، که SRGS نامیده می‌شود، ذخیره می‌شود. یا در یک فرمت XML که Grammar XML یا GRXML خوانده می شود.

فرمت کامپایل نشده، با یک پسوند فایل grxml. ذخیره می شود و فرمت کامپایل شده با پسوند cfg. ذخیره می‌شود. وقتی گفتار از گرامر تشخیص داده می شود، یک نتیجه در فرمت XML که SML یا semantic markup language خوانده می شود تولید می کند.

هنگام توسعه یک برنامه IVR ، ساختن گرامر، وقت‌گیرترین قسمت است. این‌کار بیشتر از نوشتن خود برنامه وقت می برد. خوشبختانه Speech Server، یک زوج ابزار GUI برای آسان کردن این پروسه و صرفه‌جویی در وقت پیشنهاد می کند. این دو ابزار عبارتند از:

- Conversational Grammar Builder
- Grammer Editor

هر دوی این ابزارها، گرامر keyword را پشتیبانی می‌کنند، اما فقط Grammar Buider ، گرامر مکالمه را پشتیبانی می کند.

Speech Server همچنین چندین برنامه کمکی دیگر برای کمک به ساخت گرامر، شامل semantic script builder ، Lexicon Editor و Pronunciation فراهم می‌کند.

Conversational Grammar Builder به Speech Server اجازه می دهد تا زبان طبیعی را پشتیبانی کند که به عنوان گرامر (HMIHY) یا How may I help you و Conversational Grammar Builder شناخته می‌شود. زبان طبیعی به کاربران اجازه می دهد تا پاسخ سوال را، بدون اینکه محدودیتی در چگونگی بیان آن داشته باشند، بدهند.

شما می توانید گرامر را با استفاده از Grammar Editor ایجاد کنید و با استفاده از XML editor، گرامـر را باز کنید تا به GRXML مستقیما دسترسی داشته باشید.

<grammar> باید بالاترین Container در سند grxml. باشد و باید شامل صفت های زیر باشد:

- language: علامت اختصاری زبانی که می خواهید گرامر را پشتیبانی کند.
- version: شماره نسخه SRGS که در حال حاضر 1.0 است.
- root: root rule گرامر شما
- mode: نوع برنامه ای که گرامر پشتیبانی می کند. می تواند voice یا dtmf باشد.

مثال:

“1.0” version <grammar XML : lang = “en-US”
Root = “Dependent” mode = “voice” >

اگر برنامه‌ای چندین زبان را پشتیبانی کند، لازم است تا یک پایگاه داده اعلان برای هر زبان اضافه شود.

۵-۴-۱. Prompt Engine Markup Language (PEML)

PEML به شما اجازه می دهد تا مشخص کنید که چگونه موتور اعلان باید به اعلان‌هـای از پـیش ضـبط

شده رسیدگی کند.

برای PEML عنصر <PEML> موجود است.

در صورت استفاده از TTS برای اعلان‌ها، می‌توان صدای پیش‌فرض را برای زن یا مرد تعیین نمود. همچنین می‌توان صدا را با استفاده از SSML سفارشی کرد و بلندی صدا، زیروبمی، سرعت و تلفظ یک اعلان را با استفاده از SSML اصلاح کرده و یا تغییر داد.

اجزای SSML که اعلان های صحبت موتور TTS را کنترل می کنند عبارتند از :

<SSML:speak>
Paragraph
Sentence
Say-as
Phoneme
Sub
Emphasis
Breal

از شیء Run speech نیز می توان برای مدیریت توابع اعلان استفاده نمود. به محض این که یک تماس پاسخ داده می شود، Runspeech بطور خودکار آغاز به کار می کند.

۲-۴-۵. ساختار گرامر keyword

گرامرهای Keyword، کلمات ساده یا اصطلاحات کوتاه هستند که برنامه آنها را تشخیص می‌دهد. با Conversational Grammar Builder به راحتی می توان آنها را ساخت. برای ساختن گرامر DTMF نیز می توان از Conversational Grammar Builder استفاده کرد.

۵-۴-۳. دامنه‌های voicexml

دامنه تعیین می‌کند که چگونه کارها انجام شوند و به متغیرها رسیدگی شود. voicexml چهار دامنه زیر را دارد:

- Session: یک session به محض اینکه تماس کاربر برقرار شود، قبل از اینکه اولین برنامه بارگذاری شود، شروع می شود و تا زمان قطع ارتباط کاربر ادامه دارد. هر چیزی که در این دامنه تعریف می شود در طول تمام session قابل دسترس است.
- Application: یک برنامه، یک یا چند سند voicexml است که در مدت یک session، شامل root یکسان هستند. یک کاربر ممکن است با چندین برنامه در تعامل باشد. وقتی یک کاربر به برنامه دیگری گذر می کند، برنامه قبلی از بارگذاری برداشته می شود، قبل از اینکه برنامه جدید بارگذاری شود. هر چیزی که در این دامنه تعریف می‌شود، فقط برای برنامه جاری کاربر قابل دسترس است.
- **Document**: یک سند شامل یک یا چند دیالوگ است. در هر نقطه کاربر می تواند فقط در یک دیالوگ باشد. session پایان می پذیرد وقتی که دیالوگ دیگر هیچ دیالوگ دیگری را به عنوان بعدی نداشته باشد تا ادامه دهد. هر چیزی که در اسکوپ سند تعریف می شود، برای هر دیالوگی در همان سند قابل دسترس است.
- **Dialog**: هر چیزی که در دامنه دیالوگ تعریف می شود، فقط در همان دیالوگ قابل دسترس است.

۵-۵. کنترل تعاملی voicexml

الگوریتم تفسیر فرم (FIA) ، تعاملات بین تماس گیرنده و فرم ها و منو های برنامه voicexml را کنترل می‌کند. FIA تصمیم می گیرد که کدام عنصر در هر زمان لازم است دیده شود و اجرا شود. FIA دو فاز دارد که

تعیین می کند کدام جزء نیاز دارد اجرا شود.

- **فاز Initialization**: این فاز وقتی اتفاق می افتد که یک فرم بار گذاری شده یا از بارگذاری در آمده است. این فاز تمام متغیر های در سطح فرم را، به مقادیر پیش فرض از پیش تعریف شده یا نشده، مقدار دهی اولیه می کند. FIA قسمتی از دامنه سند است، بنابراین فاز initialization یک-بار که یک سند بارگذاری می شود، رخ می دهد.
- **فاز حلقه اصلی**: بعد از پایان فاز initialization فاز حلقه اصلی آغاز می شود. درون فرم، حلقه تکراری اجرا می‌شود. این حلقه عنصرهایی را که قرار است مشاهده شوند، گرامری که می خواهد فعال شود، و اینکه چه کاری باید انجام شود، با استفاده از اطلاعاتی که از کاربر جمع آوری شده است، انتخاب می کند. حلقه اصلی به سه فاز شکسته می شود:

۱) **فاز انتخاب**[1]: قلم[2] بعدی پر نشده فرم را انتخاب می کند. سپس به صفت حالت قلم پر نشده نگاه می کند، و اگر آنرا درست ارزیابی کند، آن عنصر فرم را انتخاب می کند و پروسه جمع آوری را شروع می کند.

۲) **فاز گردآوری**[3]: این فاز، قلم انتخاب شده فرم را از فاز انتخاب مشاهده می کند. براساس قلم فرم، اعلان ها را اجرا می کند، گرامر ها را فعال می کند و ورودی را از کاربر جمع آوری می کند.

۳) **فاز پردازش**[4]: این فاز ورودی را با اجرای عناصر پر شده برای قلم، پردازش می کند. فاز انتخاب سپس تکرار می شود، و قلم بعدی پر نشده فرم انتخاب می شود.

[1] Select Phase
[2] Item
[3] Collect Phase
[4] Process Phase

۵-۶. پیاده سازی و ارزیابی یک راهنمای توریست چند وجهی[1] و چند زبانه

MUST (Multimodal، Multilingual Small Mobile Terminals)

در این قسمت، مروری بر پروژه Eurescom MUST ارائه می شود. این پروژه در فوریه ۲۰۰۱ شروع شد و تا پایان ۲۰۰۲ ادامه داشت. براساس تکنولوژی ها و بسترهای موجود، یک سیستم چند وجهی (راهنمای توریست MUST برای پاریس) پیاده سازی شده است. این سیستم، گفتار و اشاره[2] را برای ورودی می پذیرد، و خروجی آن، گفتار، متن و گرافیک برای خروجی است. بعلاوه، یک سیستم پرسش و پاسخ (Q&A) چند زبانه با آن مجتمع شده است تا درخواست‌های رسیده را بدست گیرد و پاسخگو باشد. این گفتار روی پیاده‌سازی سیستم متمرکز است. سیسستم زمان واقعی، برای ارزیابی‌های انجام شده توسط کارشناسان قابلیت استفاده[3] به‌کار می رفته است. نتایج این ارزیابی نیز توضیح داده شده اند.

برای telecom operators وservice provider ها، این یک اصل اساسی است که وسعت احتمالی استفاده از شبکه های MUST آینده را برانگیزند. کاربرد وسیع، این پیش فرض را ایجاد می‌کند که سرویس ها حداقل دو نیاز را پوشش می دهند: اول این که مشتریان باید این احساس را داشته باشند که این سرویس کارایی بهتر یا قابلیت بیشتر نسبت به انواع موجود ارائه می دهند، و دوم این که سرویس باید یک واسط طبیعی و راحت داشته باشد. بویژه نیاز آخر با قابلیت های تعاملی گوشی های موبایل کوچک و سبک برای پوشش دادن، سخت است. ترمینال‌هایی که گفتار و قلم را در سمت ورودی و متن و گرافیک و صوت را سمت خروجی در یک فرم کوچک، ترکیب می‌کنند. بستر لازم برای طراحی واسط ها چند وجهی برای ترکیب چندین مد ورودی و خروجی در یک نشست، به نظر می رسد که مشکلات انسانی و تکنولوژی جدیدی به همراه داشته باشد.

دپارتمان های تحقیق سه آزمایشگاه Telecom operator با دو انستیتوی آکادمیک در پروژه MUST

[1] Multimodal
[2] Pointing
[3] Usability

EURESCOM با هم کار کرده اند. هدف های اصلی MUST عبارتند از :

- کسب تجربه با استفاده از مجتمع‌سازی تکنولوژی‌های گفتار و زبان موجود درون یک واسط چند وجهی تجربی، به یک سیستم زمان واقعی به منظور یافتن درک بهتر از موضوعاتی که برای سرویس های چند زبانه و چند مد آینده در شبکه های موبایل قابل دسترسی از ترمینال های کوچک می توانند مهم باشند.
- بکار گیری از این سیستم برای اتصال تجربیات فاکتور انسانی با کاربران غیر متخصص طبیعی برای ارزیابی تعاملات چند وجهی.

تعاملات چند وجهی طی چندین سال مورد مطالعه قرار گرفته اند. پروژه MUST روی یک مطالعه کاربر با سیستم زمان واقعی متمرکز است که آن را تبدیل به یک سرویس واقعی کند. بعلاوه یک قسمت بزرگ از تحقیقات موجود بر اساس تجربیاتی است که موضوعاتی مثل اولویت برای وجوه بخصوص برای ترمیم خطا و مقایسات ترکیبات چندین وجه (شامل تعامل تک وجهی) را آدرس دهی می کنند.

در MUST، ما روی جمع آوری اطلاعات درباره رفتار کاربران آموزش ندیده که با یک سیستم چند وجهی با دقت طراحی شده، تعامل دارند، که بطور مجازی غیر ممکن است که بدون اجتماع گفتار و قلم برای ورودی استفاده کنند.

در این قسمت ابتدا کارایی سرویس سیستم که بعنوان back-bone پروژه MUST می باشد ارائه می‌شود. سپس معماری و واسط کاربر را تشریح خواهد شد. در آخر نتایج یک ارزیابی کارشناسی از اواین نسخه کاربردی سیستم را ارائه می‌شود.

۵-۶-۱. کارکرد سیستم

تعاملات چند وجهی، در چندین فرم می آیند که کارکرد های مختلفی را برای کاربر امکان پذیر می کند. اصطلاح فوق توسط w3c بکار رفته که برای تعاملات چند وجهی بسیار پیچیده، بکار می رود که تمام دستگاه های موجود ورودی بطور همزمان فعال هستند و action هایشان بطور همزمان تفسیر می شوند.

ما می توانیم Telecom operator را با اطلاعاتی در مورد این نوع تعاملات پیاده سازی و روی کاربرانی که از این روش جدید تعاملی استقبال می کنند، پیاده کنیم.

فقط بعضی از سرویس هایی که می‌توان برای شبکه های اینترنتی موبایل توسعه داد، خود بطور طبیعی براساس استفاده از ورودی های گفتار و متن همزمان می باشد .

یکی از سرویس هایی که به اشاره کردن (با قلم) و صحبت کردن همزمان مربوط می شود، وقتی است که یک کاربر نیاز داشته باشد تا در مورد موضوعی یا شئی روی یک نقشه صحبت کند.

راهنماهای توریستی که به جزئیات نقشه های قسمت های کوچک یک شهر می پردازد، یک مثال از این نوع سرویس ها می باشند. شهر پاریس برای سیستم MUST انتخاب شد. بنابراین، راهنمای MUST برای پاریس، به شکل قسمت‌های کوچک اطراف شهر یا Poin of interests یا (POI's)، مثل برج ایفل، the arc de triumph و غیره سازماندهی شد. PIOS ها نقطه ورودی اصلی برای گشت و گذار بودند. نقشه ها نه تنها نقشه خیابان، بلکه ارائه تصویری از ساختمان‌های مهم را ارائه می دادند. وقتی کاربر یکی از این POI ها را انتخاب می-کرد، یک نقشه کلی از اطراف آن شیء روی صفحه ترمینال نمایش داده می شود. بسیاری از قسمت‌های نقشه، شامل اشیاء اضافی بود که ممکن بود برای بیننده جالب باشد. با اشاره به این اشیاء روی صفحه، آنها موضوع مکالمه می شوند، و کاربر می تواند سوالاتی در مورد این اشیاء بپرسد. مثلا : "What is this bulding?" و "What are the opening hours" کاربر همچنین می تواند سوالات عمومی تری درباره قسمتی از شهر که نمایش داده شده، بپرسد، مثل : " what restaurants are in this neighbour hood?" این سوال آیکون هایی برا رستوران اضافه می کند تا نمایش داده شوند. این مسئله می‌تواند به موضوع مکالمه تبدیل شود اگر به آن اشاره شود و سوالاتی پرسیده شود. برای مثال برای نوع غذا، دامنه قیمت و ساعت های باز بودن رستوران. این اطلاعات توسط سیستم به صورت متن، گرافیک (نقشه ها و تصاویر هتل ها و رستوران‌ها)، و تبدیل متن به گفتار انجام می شود.

برای اپراتورهای شبکه موبایل، یک قسمت اساسی دسترسی به سرویس ها از مشتریان رومینگ می‌آید. به

خوبی روشن شده است که مردم ترجیح می دهند زبان ملی خودشان را بکار ببرند. بخصوص وقتی از تشخیص دهنده‌های گفتار استفاده می شود، چون این ابزارها برای گفتارهای غیر ملی، کاهش کارایی دارند. بنابر این سرویس‌های اطلاعاتی که در شبکه های موبایل استفاده می شوند، باید چند زبانه باشند. بنابر این باید به مشتریان اجازه دهند تا هر کس زبان ملی خودش را بکار ببرد. سیستم MUST برای زبان‌های انگلیسی، فرانسوی، پرتقالی و نروژی توسعه داده شده است.

کاربران اجازه خواهند داشت که سوالاتی درباره POI's ها برای پاسخ هایی که در پایگاه داده سرویس وجود ندارد، نیز بپرسند. البته معمولا کاربران کمی هستند که دوست دارند چنین سوالاتی بپرسند. مثل :

Who is architect of this building?

What other buildings has he designed in paris?

برای پاسخگویی به این سوالات، یک دسترسی به سیستم پرسش و پاسخ چند زبانه که توسط France telecom R&D توسعه یافته است، ایجاد می شود و کاربران می توانند پاسخ اینگونه سوالاتشان را از روی اینترنت بیابند.

۵-۶-۲. معماری سیستم

معماری کلی سیستم MUST در شکل۵-۳ نمایش داده شده است. قسمت سرور معماری شامل تعدادی ماژول های خاص است که اطلاعات را بین همدیگر رد و بدل می کنند. سرور توسط کاربر از طریق یک مشتری کم‌حجم[1] که روی ترمینال یک موبایل اجرا می شود، قابل دسترسی است. سرورهای application، بر اساس Portugal Telecom Inovacao وTelenor R&D Voice Server، تفاوت دارند. سرویس صوتی یک واسط به تلفن ISDN و PSTN و نیز منابع صوتی پیشرفته مثل یک ASR (Automatic speech recognition) و TTS (Text-to-speech) را فراهم می‌کند.

[1] Thin client

امکانات ASR مثل confidence scores و لیست‌های N-best پشتیبانی شده‌اند. موتور TTS برای تبدیل زمان واقعی خروجی گفتار بکار می‌رود. موتور های TTS مختلفی برای زبان‌های مختلف در MUST بکار می‌روند.

France Telecom و Telenor از موتور های تبدیل متن به گفتار built– home استفاده می کنند. در حالی که portugal Telecom از Real speak و L & H استفاده می کنند.

سیستم پرسش و پاسخ (Q/A) چند زبانه مجموعه ای از تجزیه معنایی/گرامری و تکنیک های پردازش زبان طبیعی استاتیکی برای جستجوی وب برای اسناد مرتبط استفاده می کند. این جستجو بر اساس یک سوال که به زبان طبیعی پرسیده شده است، کار می کند و سعی می کند مستقیماً یک پاسخ کوتاه از این اسناد استخراج کرده و پاسخ دهد، سایز این پاسخ (بر اساس تعداد کاراکتر ها)، نمی تواند بصورت پیشرفته پیش‌بینی شود، اما این انتظار هست که اغلب پاسخ‌هایی که به اندازه کافی کوتاه باشند، برای پرکردن متنی که برای ارائه اطلاعاتی که الان در پایگاه داده موجود هست، توسط TTS فراهم شود.

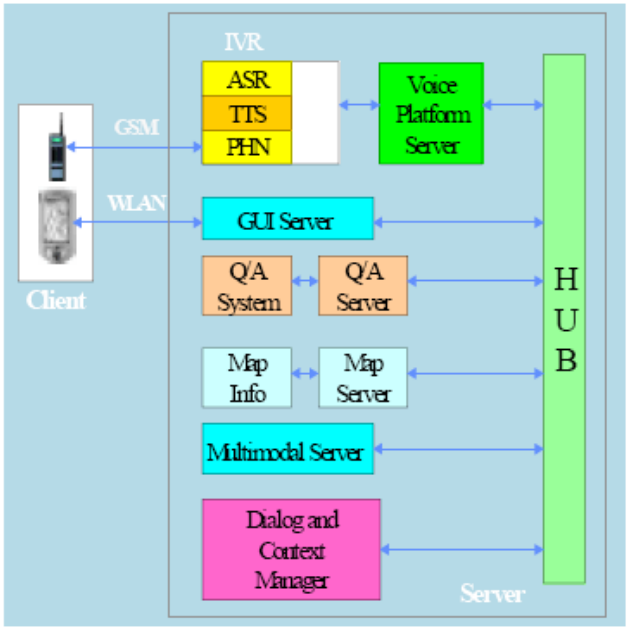
GALAXY Communicator Software Infrastructure، که یک نسخه مرجع دامنه عمومی از DARPA Communicator است و توسط MITRE نگهداری می شود، بعنوان چارچوب[1] انتقالی از سیستم در نظر گرفته شده است.

امکان موجود دیگر HUB است (شکل۵-۳). خصوصیات اصلی این چارچوب عبارتند از: ماژولار بودن، طبیعت توزیع شده، یکپارچه بودن اجتماع ماژول ها، و انعطاف پذیری به‌صورتinter – module data exchange (ارتباط همزمان و غیرهمزمان مستقیما از طریق HUB بین ماژول ها).

GALAXY به اجزای موجود (مثل ASR و TTS و غیره) اجازه می دهد تا به همدیگر با روش‌های مختلف پیوند بخورند. با فراهم کردن امکانات extensive برای عبور دادن پیغام ها بین اجزا از میان HUB مرکزی، یک جزء می تواند براحتی یک کارکرد را طلب کند که توسط سایر اجزا فراهم شده است، بدون دانستن اینکه کدام

[1] Framework

جزء آنرا فراهم می کند یا کجا اجرا می شود.



شکل5-3. معماری شماتیک MUST

پردازش در HUB می تواند با استفاده از یک اسکریپت کنترل شود یا می تواند بصورت یک تسهیل‌کننده در یک سیستم بر اساس عامل[1] کار کند. در MUST، کنترل پیغام دهی HUB، بر پایه اسکریپت است. ماژول ها در java و C یا C++ تحت لینوکس و ویندوز NT نوشته شده اند. به منظور نگهداری فرمت پیغام هایی که بین ماژول های ساده و منعطف، رد و بدل می شوند، تصمیم گرفته شده که از یک زبان بر پایه XML که MXML یا MUST XML Mark Up Language استفاده می شود. MXML برای ارائه اغلب محتوا های چند وجهی که بین ماژول‌ها ردوبدل می شوند بکار می رود. پارامترهای مورد نیاز برای آماده‌سازی، همزمان کردن و قطع اتصال ماژول‌ها از صفت‌های کلید–مقدار[2] در پیغام های GALAXY استفاده می‌کنند. قسمت مشتری سیستم، روی یک COMPAQ iPAQ Pocket PC روی Microsoft CE با اتصال WLAN اجرا می شود. قسمت گفتار توسط

[1] Agent
[2] Key-value

یک تلفن موبایل انجام می شود. کاربر به راه حل این دو قسمت توجهی نخواهد داشت، زیرا تلفن مخفی می شود و واسط ظاهر خواهد شد و فقط میکروفن و گوشی با یک اتصال سیم برای کاربر قابل مشاهده است.

گفتار بیان شده، به تشخیص دهنده گفتار با استفاده از ماژول تلفنی فرستاده می شود. ورودی های متن و قلم از GUI Client از طریق اتصال TCP/IP به سرور GUI انتقال می یابد. ورودی ها از تشخیص دهنده گفتار و سرور GUI در سرور چند وجهی[1] مجتمع می شوند و به Dialogue / Context Manger یا DM فرستاده می شوند. DM نتایج را تفسیر می کند و بر طبق آن عمل می کند، برای مثال اتصال به Map server و واکشی اطلاعات که برای کاربر ارائه می شود. اطلاعات سپس به سرور GUI و سرور صوتی از طریق سرور Multimodal ارسال می‌گردد که سرور Multimodal، کار استخراج داده های آدرس داده شده را به خروجی انجام می دهد. این خروجی گفتار و گرافیک است.

۵-۶-۳. واسط کاربر سیستم

یک خصوصیت مهم برای واسط کاربر، کارکرد "Tap while Talk" است وقتی قلم اندکی قبل در همان زمان و یا اندکی بعد از گفتار به‌کار گرفته می‌شود، دو عمل ورودی به یک ورودی مجتمع می شوند. اگر زمان بین Tapping و گفتار بیشتر از یک مقدار از قبل تنظیم شده باشد، این اعمال بصورت اعمال جداگانه و پشت سر هم و مستقل از هم در نظر گرفته می شوند.

تشخیص دهنده گفتار، باید همیشه باز باشد تا بتواند ورودی را بگیرد، در ظاهر این کار پردازش سیگنال وتشخیص گفتار را پیچیده می کند. اگر چه، این سخت است که یک گزینه انتخابی[2] برای یک ASR فعال پیوسته بدون تغییر در استراتژی تعامل تصور کنیم.

---

[1] Multimodal
[2] Alternative

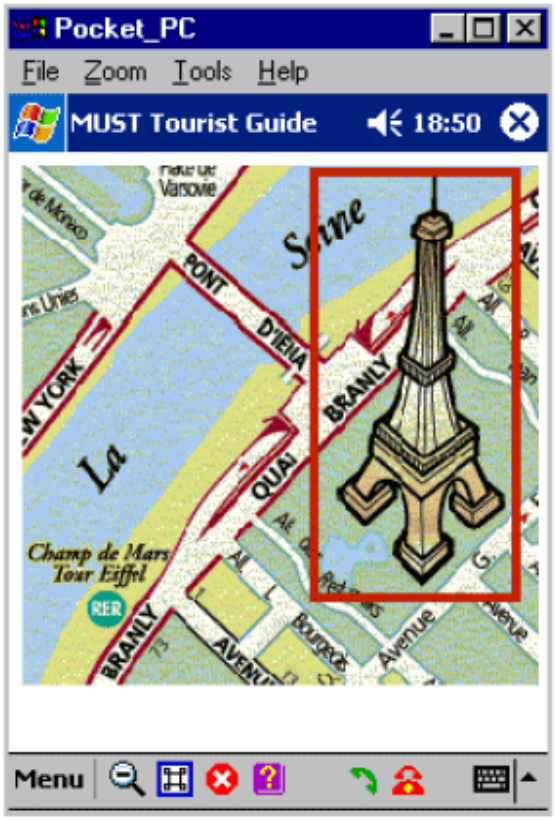
کاربران می توانند به عملیات پشت سر هم، با باقی گذاشتن مقدار کافی زمان بین گفتار و حرکت قلم، رجوع کنند.

اطلاعات خروجی بطور اهم، به شکل متن (مثل :"the entrance fee is 3 euro") وگرافیک (نقشه‌های تصاویر هتل ها و رستوران ها) می باشند. خروجی متنی بصورت جعبه متن، روی صفحه نمایش داده می‌شوند.

برای کمک به کاربر برای آنکه حالت سیستم را نگهداری و پیگیری کند، سیستم همیشه به یک ورودی پاسخ خواهد داد. در بسیاری مواقع، پاسخ گرافیکی است. برای مثال وقتی یک POI انتخاب می شود، سیستم با نمایش نقشه متناظر با آن پاسخ می دهد. اگر سیستم یک ابهام را تشخیص دهد (مثل :اگر ورودی صدا تشخیص داده شده باشد، اما ASR نتواند ورودی را با اطمینان به اندازه کافی بالا، تشخیص دهد)، یک اعلان ایجاد می کند که می گوید گفته را خوب نفهمیده است.

قسمت گرافیکی واسط کاربر شامل دو نوع نقشه است: یک نقشه کلی که همه POI ها را نشان می دهد و یک نقشه با جزئیات با یک POI در مرکز.

Dialogue/Context Manager، طوری طراحی شده که تعامل، بدون تمرکز روی دیالوگ اجرا شود. بنابراین اولین عملی که یک کاربر باید انجام دهد این است یک POI را انتخاب کند. روی شیء انتخاب شده، بطور خودکار تمرکز خواهد شد. وقتی یک کاربر یک POI را انتخاب کرد، تسهیلات مختلفی مثل هتل ها و رستوران‌ها بصورت اشیاء روی نقشه نشان داده می شوند. مثلاً با استفاده از گفتار مثل : What hotel are there in this neighborhood ? ، یا استفاده از Tapping کردن روی یکی از دکمه های کمکی که در قسمت پایین صفحه نمایش داده می شود. شکل۵-۴ نمایی از اجرای MUST را نشان می‌دهد.



شکل5-4. نمایی از صفحه نمایش یک Pocket PC در حال اجرای MUST

۵-۷. قابلیت تشخیص گفتار در ویندوز ویستا

ویندوز ویستا اولین سیستم عامل مایکروسافت است که تشخیص گفتار built-in دارد. با سیستم عامل های قبلی مثل ویندوز XP، لازم بود قابلیت‌های تشخیص گفتار را بعنوان یک واحد جدا نصب نمود.

این به این معنی است که اغلب برنامه‌هایی که تحت ویندوز ویستا اجرا می شوند، گفتار را بدون دسترسی مستقیم به Speech API پشتیبانی می کنند.

تشخیص‌دهنده گفتار، بطور واقعی متن یک برنامه را می خواند، مثل متن روی یک دکمه یا منو، و به کاربر اجازه می‌دهد تا به آنها به سادگی با گفتن نامشان دسترسی داشته باشد.

ویندوز ویستا ۱۳ زبان را پشتیبانی می کند. این زبان‌ها انگلیسی با لهجه آمریکایی و بریتانیایی، فرانسه،

چینی، ژاپنی، دانمارکی، هلندی، فنلاندی، آلمانی، ایتالیایی، کره ای، نروژی، اسپانیایی و پرتقالی هستند.

زبان تشخیص دهنده گفتار، و صداها، بستگی به نسخه زبان سیستم عامل دارد. برای مثال نسخه انگلیسی ویندوز ویستا، گفتار چینی را تشخیص نمی‌دهد و حتی نمی تواند صدای چینی را بشنود. اگر از ویندوز ویستا Ultimate یا Enterprice استفاده شود، می توان بسته زبان‌های دیگر را از طریق windows update بدست آورد. یک‌بار که این بسته‌های زبان نصب شود، کاربر، از طریق پانل کنترل می تواند، انتخاب کند که کدام صدا یا تشخیص گفتار استفاده شود.

Start meno **à** Accessories **à** ease of Access **à** speech recognition **à** options **à** Advanced speech recognition options

ویندوز ویستا دو صدای جدید دارد : Microsoft Anna و Microsoft Lili که Anna صدای جدید انگلیسی و Lili صدای جدید چینی است. این صدا ها صدا های ضبط شده واقعی هستند.

۸-۵. نمونه های نرم افزارهای تشخیص گفتار

در این قسمت، چند نمونه از نرم افزارهای تجاری که قابلیت تشخیص گفتار دارند و به‌عنوان سیستم‌های IVR در دنیا کاربرد دارند، توضیح داده شده است[۲۱].

۱-۸-۵. Philips speech SDK

در ۱۸ زبان پردازش انجام می دهد و قابل برنامه نویسی از طریق یک c/c++API می‌باشد[1].

۲-۸-۵. Commercial lingsoft speech recognizer

ارزانتر و دقیق تر از Philips است اما به برنامه نویسی بیشتری نیاز دارد و بصورت فایل‌های dll برای

[1] http://www.speechrecognition.philips.com/index.php?id=183

ویندوز2000 و NT4 و لینوکس وجود دارد. این سیستم یک فرهنگ لغت قابل تعریف توسط کاربر[1] دارد که حدود ۱۸۰۰ کلمه با ۹۴% تا ۹۸% دقت تشخیص گفتار دارد[2].

۵-۸-۳. Nuance speech recognition software

بهترین محصول در بازار است و معماری آن برای برنامه‌های تلفنی بهینه است. ۲۷ زبان را پشتیبانی می‌کند و خصوصیاتی مثل keyword spotting from phrases را داراست (که به کاربر اجازه می دهد که تمام جملات را به زبان محاوره بگوید). Nuance 8.0، دارای واسط نرم افزاری برای c/c++ و Java می‌باشد. البته Finnish را پشتیبانی نمی‌کند. بنابراین برای Finnish می‌توان از Philips speech SDK یا از Lingsoft Recognizer استفاده نمود.

در Nuance قابلیت های تشخیص زبان طبیعی، اجازه انعطاف پذیری پاسخ تماس گیرنده را می‌دهد، تعاملات را ساده و کامل انجام می دهد. با Nuance verifier، یک سیستم شناسایی هویت صوتی که به سادگی کار می کند و دقتی حدود ۹۹.۹% دارد، تعاملات سفارش کالا را می تواند با دقت مدیریت کند. علاوه بر این Nuance 8.0 ، voiceXML را پشتیبانی می کند بنابراین توسعه مجدد سیستم بسیار راحت است.

۵-۸-۴. Nuance Vocalizer 3.0

۱۸ موتور TTS را به زبان های مختلف پشتیبانی می‌کند. برای شبکه های فروش مناسب است. شبکه فروش، یکی از کاربرد های IVR که می توان به آن اشاره کرد.

بطور مثال، یک سازنده لوازم الکترونیک، از یک سیستم مبتنی بر گفتار خودکار برای فروش خود استفاده می‌کند. فروشنده، با یک سرویس اتوماتیک مشتری تماس می گیرد و اطلاعاتی راجع به محصولات مختلف خود می دهد و سفارشاتی برای آنها قرار می دهد.

---

[1] User-defined

2 http://speech.lingsoft.fi/fi/lssr.html

واسط باید بتواند مشخصه های گفتار منحصر به فرد را از یک مشتری بفهمد و به کاربر اجازه می دهد که از زبان طبیعی خود برای سفارش دادن کالاها استفاده کند. علاوه بر این، لازم است که سیستم شبکه فروش، voiceXML را پشتیبانی کند بطوریکه در آینده راحت تر بتواند سیستم مبتنی بر تلفن را با سرویس‌های وب سازندگان مجتمع سازد. همچنان به یک تشخیص دهنده گفتار نیاز است تا از دقت سفارش مشتری مطمئن شود. نرم افزار Nuance 8.0، تشخیص گفتار های مرتبط با واسط ها کاربر مبتنی بر تلفن دارد، که به تماس-گیرندگان اجازه می دهد تا آزادانه و به زبان طبیعی صحبت کنند و همچنان voiceXML را پشتیبانی می کند.

Nuance vocalizer 3.0 ، نرم افزار تولید گفتار متن به گفتار[1] Nuance است که اطلاعات را با صدای طبیعی شنیده شده از کاربر، دریافت می‌کند و تشخیص می‌دهد.

این به همراه Nuance Verifier و Nuance Voice Platform، توسعه و پیاده سازی شبکه های فروش مبتنی بر تلفن را برای سازندگان لوازم الکترونیکی فراهم می کند.

۵-۸-۵. Micro REC SD

یک محصول تشخیص گفتار است که برای میکروکنترلرها استفاده می‌شود. محصولات MicroREC برای محصولات high-volume ، low cost و تعاملی مثل لوازم خانگی هوشمند یا اسباب بازی های الکترونیکی طراحی شده اند.

۶-۸-۵. Microsoft Speech Server (MSS):

برای دامنه وسیعی از دستگاه ها مثل Pocket PC ها ،Table PC ها ، smart phone ها و landline phone ها طراحی شده است.

[1] TTS (Text To Speech)

۵-۸-۷. Nuance V-Builder 2.0

یک محیط مجتمع برای توسعه و تست برنامه‌های صوتی فراهم می‌کند.

۵-۸-۸. IBM's Voice Toolkit

یک Toolkit برای توسعه برنامه های صوتی به مرحله بعدی (آینده)، با استفاده از ابزار های call-flow گرافیکی و ابزار های فهم زبان طبیعی (NLU) می باشد.

۵-۸-۹. Loquendo

تکنولوژی ها TTS ، ASR (تشخیص گفتار اتوماتیک) و Speaker verification دارد که ۱۷ زبان با ۳۹ صدا را پشتیبانی می کند.

۵-۸-۱۰. Neospeech

تکنولژی TTS برای بازارهای موبایل، enterprise و education دارد. موتور TTS آن، برای انگلیسی US و زبانهای آسیایی اصلی را پشتیبانی می کند.

۵-۸-۱۱. Sky Greek

تعدادی برنامه های IVR دارد. بستر IVR آن voicexml است و شامل اجزای مبتنی بر وب ساده برای بکارگیری (easy-to-use) است که برای طراحی، مدیریت و گزارش گیری برنامه های VIP بکار می رود.

۵-۹. چند نمونه نرم افزار تبدیل متن به گفتار

۵-۹-۱. VoiceMX STUDIO 4

VoiceMX STUDIO 4، یک نرم افزار تولید کننده گفتار متن به گفتار است که تمام انواع متن را با استفاده از یک موتور گفتار با کیفیت بالا، به گفتار تبدیل می کند. ورودی متنی این نرم افزار، می تواند نوشته های متنی، اسناد، ایمیل ها، e-book ها، صفحات وب و .. باشد. این نرم افزار روی سیستم های عامل های زیر

قابل اجراست:

Windows 98، Windows Me، Windows NT، Windows 2000، Windows XP

۲-۹-۵. ECTACO Voice Translator Russian -> Japanese 1.21.24

این نرم افزار مترجم صوتی روسی به ژاپنی، دارای ۳۲۰۰ کلمه و اصطلاح در ۱۵ دسته است و یک موتور TTS درونی دارد که خروجی آن، شامل متن و گفتار است.

فصل 6

سیستم های تبدیل متن به گفتار و فناوری جاوا

۶-۱. مقدمه

فناوری گفتار[1]، سابقا مربوط به قلمرو داستان های علمی بود، ولی حالا برای استفاده در برنامه های واقعی موجود است. کتابخانه برنامه نویسی گفتار جاوا[2]، که توسط شرکت سان مایکروسیستمز به کمک شرکت های دیگری که در زمینه فناوری گفتار کار می کنند، توسعه یافته است، یک رابط نرم افزاری را مشخص می کند که توسعه دهندگان بتوانند از برتری های این فناوری برای محاسبات شخصی و سازمانی استفاده کنند. با به کار بردن قدرت ذاتی بستر جاوا، کتابخانه گفتار جاوا، برنامه نویسان می‌توانند برنامه های کاربردی و یا اپلت های زیبایی با قدرت سخن گفتن را بسازند و آن ها را در طیف وسیعی از بستر ها اجرا کنند.

۶-۲. کتابخانه گفتار جاوا چیست؟

کتابخانه گفتار جاوا یک رابط نرم افزاری استاندارد با استفاده آسان است که توانایی اجرا روی بسترها متعدد را برای فناوری مدرن گفتار معین می کند. دو فناوری هسته ای در این کتابخانه پشتیبانی می‌شوند عبارتند از: تشخیص گفتار[3] و تبدیل متن به گفتار.

تشخیص صوت این توانایی را به کامپیوترها می‌دهد که به صحبت شخصی که به یک زبان انسانی سخن می‌گوید گوش کنند و مشخص کنند که آن شخص چه چیزی گفته است. به بیانی دیگر صوتی را که شامل یک گفتار است را دریافت کرده، روی آن پردازش انجام داده و آن را به متن تبدیل می کند.

شرکت ها و افراد می توانند از فواید طیف وسیعی از برنامه هایی که از فناوری گفتار استفاده می کنند سود ببرند. برای مثال، سیستم های پاسخ صوتی فعل و انفعالی[4]، یک راه متناسب برای برقراری ارتباط با کاربران از

1
2 Java Speech API
3 Speech Recognition
4 Interactive voice response systems

طریق رابط‌های تلفنی است. سیستم‌های دیکته‌ای به مراتب سریع‌تر از روش‌های تایپ توسط کاربران هستند. فناوری گفتار توانایی بسیاری از کابران با محدودیت‌های فیزیکی و معلولان را برای ارتباط با کامپیوتر بهتر می‌کند.

رابط‌های گفتاری این فرصت را به برنامه‌نویسان جاوا می دهند که ویژگی‌های مجزا و فعالی را به برنامه‌های خود ببخشند و برنامه‌های خود را از دیگر محصولات متمایز سازند. با یک کتابخانه استاندارد برای گفتار، کاربران می توانند محصولات گفتاری را انتخاب کنند که بهترین تناسب را با احتیاجات و بودجه آن‌ها داشته باشد.

کتابخانه گفتار جاوا در یک فرآیند توسعه باز توسعه یافته است. با شرکت فعال شرکت های پیشتاز در فناوری گفتار، با ورودی از برنامه ها، توسعه‌دهندگان و ماه‌ها بازبینی عمومی و نظردهی ، مشخصات این کتابخانه با درجه بالایی از برتری‌های تکنیکی بدست آمده است. در این توسعه پرسرعت و همگانی شرکت Sun Microsystems این کتابخانه را برای رسیدن به امکانات لازم پشتیبانی نموده است.

کتابخانه گفتار جاوا یک الحاق به بستر جاوا است. الحاق ها بسته هایی از کلاس های نوشته شده به زبان جاوا هستند که ممکن است از کتابخانه های محلی سیستم‌عامل‌ها نیز استفاده کرده باشند.

۳-۶. اهداف طراحی برای کتابخانه گفتار جاوا

مانند سایر کتابخانه های چند رسانه ای جاوا[1]، کتابخانه گفتار جاوا به کاربران اجازه می دهد که رابط‌های کاربری پیشرفته را با برنامه های جاوا متحد کنند. اهداف طراحی برای کتابخانه‌های گفتار جاوا عبارتند از:

- مجتمع سازی با توانایی‌های دیگر بسترهای جاوا مانند کتابخانه‌های چند رسانه‌ای، تهیه پشتیبانی برای تبدیل متن به گفتار برای هر دوی تشخیص دهنده‌های گفتار «دستور بده-و-کنترل کن»[2] و «دیکته ای»[3]

1 Java Media API
2 Command-and-Control speech recognizer
3 Dictation speech recognizer

- تهیه کردن یک رابط چندبستره و چندفروشنده‌ای قدرتمند برای تولید گفتار و تشخیص گفتار.
- توانایی دسترسی به فناوری مدرن گفتار.
- پشتیبانی کردن از جاوا.
- ساده بودن، جمع و جور بودن و یادگیری آسان.

۴-۶. برنامه های جاوا با توانایی های گفتاری

توانایی های موجود بستر جاوا، آن را برای توسعه طیف وسیعی از برنامه ها برای توسعه‌دهندگان جذاب می سازد. با اضافه شدن کتابخانه گفتار جاوا، توسعه دهندگان برنامه های جاوا می توانند رابط های کاربری موجود را با ورودی و خروجی گفتار توسعه داده و کامل کنند.

برای برنامه نویسان برنامه های گفتاری، امروز بستر جاوا امکانات جذاب تازه ای را پیشنهاد می‌دهد:

- توانایی جابه جایی: زبان برنامه نویسی جاوا، کتابخانه ها و ماشین مجازی برای طیف وسیعی از بستر های سخت افزاری و سیستم های عامل موجود است و در مهمترین کاوشگر[1]های وب به خوبی پشتیبانی می شود.
- محیط قدرتمند و فشرده: بستر جاوا یک زبان قدرتمند و شیئ گرا است که توانایی جمع کردن آشغال های حافظه را نیز دارد و توسعه سریع[2] و قدرتمند را برای برنامه نویسان ممکن می کند.
- مبتنی بر شبکه و امن: از همان ابتدای پیدایش، بستر جاوا مبتنی بر شبکه بوده است و توجه به ویژگی های امنیتی به طور کامل در آن لحاظ شده است.

1 Browser
2 Rapid development

۶-۵. گفتار و کتابخانه های دیگر جاوا

کتابخانه گفتار جاوا یکی از کتابخانه های چند رسانه ای جاوا است، یک سوئیت از رابط های نرم افزاری که دسترسی به صدا، ویدئو و نواخت های دیگر چند رسانه ای، گرافیک دو بعدی و سه بعدی، انیمیشن، تلفنی، پردازش تصویر پیشرفته و غیره را برای اجرا در چندین پلت فرم بدون تغییر اضافی در کدها پشتیبانی می کنند.

کتابخانه گفتار جاوا در ترکیب با دیگر کتابخانه های چند رسانه ای جاوا، برنامه نویسان را توانا می سازد تا برنامه های کاربردی و اپلت های خود را با امکانات ارتباطی بسیار پیشرفته که انتظارات کاربران امروز را نیز برآورده کند، مجهز سازند و ارتباطات شخص به شخص[1] را بهبود بخشند.

کتابخانه گفتار جاوا از توانایی های دیگر کتابخانه های جاوا نیز بهره می جوید. امکانات جهانی‌سازی زبان برنامه سازی جاوا به علاوه استفاده از مجموعه کاراکتر های یونیکد، توسعه برنامه های چند زبانه با توانایی های گفتاری را ساده می کند.

کلاس ها و رابط های کتابخانه گفتار جاوا از الگو های JavaBeans پیروی می کنند و کاملا با آنها سازگار هستند. در نهایت، رویداد[2] های کتابخانه گفتار جاوا با توانایی های مکانیزمی AWT، JavaBeans و کلاس های اساسی جاوا[3] مجتمع می شوند.

۶-۶. پیاده سازی ها

کتابخانه گفتار جاوا دسترسی به فناوری های گفتار مدرن و مفید را ممکن می سازد. شرکت سان مایکروسیستمز با شرکت های فناوری گفتار برای پیاده سازی های کتابخانه ها کار می کند. در حال حاضر، تشخیص گفتار و تولید گفتار آن روی بسیاری از بستر ها از طریق این کتابخانه ها قابل دسترسی هستند.

آنچه در زیر می آید، مکانیزم های اصلی برای پیاده سازی کتابخانه هستند:

1 Person-to-Person communication

2 Event

3 Java Foundation Classes (JFC)

- پیاده سازی های محلی[1]: بیشتر فناوری های گفتار موجود به زبان C یا C++ توسعه یافته اند و تنها از طریق کتابخانه های خود بستری که در آن استفاده می شوند قابل دسترسی هستند مانند: Apple Speech Managers و Microsoft's Speech API (SAPI) و یا کتابخانه های تجاری شرکت های مختلف. با استفاده از رابط محلی جاوا[2] و پوشش های نرم افزاری جاوا[3]، شرکت های توسعه دهنده این فناوری می توانند از کتابخانه های جاوا بر روی کتابخانه های خودشان استفاده کنند.
- پیاده سازی های نرم افزاری جاوا: تولید کننده گفتار های گفتار و تشخیص دهنده های گفتار می‌توانند توسط نرم افزار جاوا نوشته شوند. این پیاده سازی ها می توانند از فواید قابل حمل بودن زبان جاوا و بهبودهای فزاینده سرعت اجرای ماشین های مجازی جاوا[4] استفاده کنند.
- پیاده سازی های تلفنی: برنامه های سازمانی تلفنی معمولا با سخت افزار های اختصاصی برای پشتیبانی از ارتباطات همزمان، مانند DSP card ها، توسعه یافته اند. توانایی های تشخیص و تبدیل متن به گفتار روی این سخت افزار های اختصاصی می توانند با نرم افزار جاوا پوشش داده شوند با نوعی از پیاده سازی محلی را برای کتابخانه گفتار جاوا ممکن کنند.

۶-۷. نیازمندی ها

به منظور استفاده از کتابخانه گفتار جاوا، یک کاربر باید مطمئن شود که امکانات نرم افزاری و سخت افزاری مورد نیاز را دارد. در زیر مهمترین نیاز ها را برای اجرای کتابخانه گفتار جاوا می بینید. البته نیازهای فردی برای

---

1 Native implementations

2 Java Native Interfac (JNI)

3 Java software wrappers

wrapper در دنیای نرم افزار برنامه ای است که برنامه ی دیگری را شروع می کند. آن برنامه ممکن است نتواند بعضی کارها را انجام دهد به همین دلیل برنامه ی دیگری را اجرا می کند.

4 Java virtual machine

تولید کننده گفتار و تشخیص دهنده جاوا می تواند در بین کاربران بسیار متفاوت باشد و کاربران باید دقیقا نیاز های محصول مورد استفاده خود را چک کنند:

- نرم افزار گفتار: یک نرم افزار تولید کننده گفتار یا تشخیص دهنده گفتار مبتنی بر Java Speech APIمورد نیاز است.
- نیاز های سیستمی: بسیاری از تشخیص دهنده های صوتی دسکتاپی و بعضی از تولید گفتارکننده-ها احتیاج به کامپیوتر های نسبتا قدرتمندی برای اجرا شدن دارند. وقتی یک محصول را انتخاب می کنید بایستی به احتیاجات حداقلی مورد نیاز برای پردازشگر، حافظه و فضای دیسک خالی نگاهی بیندازید.
- سخت افزار صوتی: تولید کننده گفتار های گفتار احتیاج به خروجی صوتی دارند. تشخیص دهنده های گفتار احتیاج به ورودی صوتی دارند. بیشتر کامپیوتر های دسکتاپ و لپ تاپ ها که در حال حاضر تولید می شوند نیاز های صوتی مورد نظر را پشتیبانی می کنند. بیشتر سیستم های دیکته ای با دستگاه های صوتی با کیفیت بیشتر، بهتر کار می‌کنند.
- میکروفن: سیستم های تشخیص صوت دسکتاپ ورودی صوتی خود را از طریق یک میکروفن دریافت می کنند. بعضی از تشخیص دهنده ها، مخصوصا سیستم های دیکته ای، به میکروفن ها حساس هستند و بسیاری از محصولات میکروفن های خاصی را پیشنهاد می کنند. میکروفن های هدست اغلب بهترین کارایی را دارند، مخصوصا در محیط های پر سر و صدا. میکروفن های رومیزی نیز می توانند در بعضی از محیط ها برای بعضی از کاربرد ها استفاده شوند.

۸-۶. موتورهای گفتار javax.speech

javax.speech بسته[1] ای از کتابخانه گفتار جاوا است که یک نمایش نرم افزاری مجرد[2] از یک موتور گفتار است. موتور گفتار یک اصطلاح کلی برای یک سیستم است که برای کار با هر دوی ورودی و خروجی گفتار ساخته شده است. تولید کننده گفتار ها و تشخیص دهنده های گفتار هر دو نمونه هایی از موتور گفتار هستند. سیستم های تصدیق/تشخیص گوینده و سیستم های هویت سنجی گوینده نیز موتور گفتار هستند اما هنوز جزئی از کتابخانه گفتار جاوا نیستند.

بسته javax.speech کلاس ها و رابط هایی را تعریف می کنند که دربرگیرنده توابع پایه ای یک موتور گفتار است. بسته های javax.speech.synthesis و javax.speech.recognition توابع پایه ای را بسط داده و تکمیل می کنند تا قابلیت های خاص تولید کننده گفتار ها و تشخیص دهنده های گفتار را تعریف کنند.

کتابخانه گفتار جاوا تنها یک فرض را در مورد پیاده سازی یک موتور JSAPI در نظر می گیرد: که این یک پیاده سازی واقعی از همه کلاس ها و رابط های جاوا است که در کتابخانه تعیین شده اند اما پیاده سازی نشده اند. برای پشتیبانی آن کلاس ها و رابط ها، یک موتور گفتار می تواند کاملا بر پایه نرم افزار باشد یا اینکه ترکیبی از سخت افزار و نرم افزار باشد. موتور می تواند محلی باشد یا از روی یک خدمتگذار اجرا شود. موتور می تواند فقط با جاوا پیاده سازی شده باشد یا ترکیبی از نرم افزار جاوا و یک کد محلی باشد.

پردازه های پایه ای برای استفاده از یک موتور گفتار در یک برنامه در زیر آمده است:

۱. مشخص کردن نیاز های تابعی موتور مورد نیاز برنامه (مانند زبان یا توانایی دیکته سنجی)
۲. قرار دادن و ساختن یک موتور که آن نیاز های تابعی را برآورده کند.
۳. تخصیص منابع برای موتور.
۴. برپا کردن موتور.

1 Package
2 Abstract software representation

۵. آغاز کردن عمل آن موتور، به طور تکنیکی ادامه دادن[1] آن.

۶. استفاده از موتور.

۷. بازگرداندن منابع تخصیص داده شده.

گام های ۴ و ۶ در این پروسه، برای دو موتور گفتار تولید گفتار و تشخیص متفاوت عمل می کنند. اما بقیه این گام ها برای همه موتور ها یکسان هستند.

۶-۸-۱. خصوصیات یک موتور گفتار

برنامه های کاربردی هستند که مشخص می کنند به چه نوع از موتور گفتار و با چه خصوصیاتی نیاز دارند. برای مثال یک برنامه ممکن است مشخص کند که به یک تشخیص دهنده دیکته ای برای یک زبان محلی یا یک تبدیل متن به گفتار برای زبان کره ای با یک صدای زنانه، نیاز دارد. همچنین برنامه های کاربردی مشخص می کنند که اگر موتور گفتار مورد نظر با آن خصوصیات خاص موجود نباشد، چه اتفاقی بیفتد.

بر اساس نیاز های تابعی خاص، یک موتور گفتار می تواند انتخاب شود، ساخته شود و شروع شود. این بخش توضیح می دهد چگونه خصوصیات یک موتور گفتار در انتخاب یک موتور استفاده می‌شود و چگونه این خصوصیات در یک برنامه جاوا کنترل می شود.

احتیاجات تابعی در برنامه ها به صورت خصوصیات انتخاب موتور[2] کنترل می شود: هر تولید کننده گفتار و تشخیص دهنده گفتار نصب شده، با استفاده از مجموعه ای از خصوصیات معین می شود. یک موتور نصب شده ممکن است یک یا بسیاری حالت های کاری[3] داشته باشد که هر کدام با یک مجموعه منحصر به فرد از خصوصیات مشخص می شوند و در یک شیئ توصیفگر حالت[4] کپسوله می شود.

خصوصیات پایه ای موتور در کلاس EngineModeDesc مشخص شده اند. خصوصیات اضافی تر برای

1 Resume
2 Engine selection properties
3 Modes of operation
4 Mode discriptor

تشـخیص دهنـده و تولیـد گفتارکننـده گفتـار بـه ترتیـب در کـلاس هـای ReconizerModeDesc و SysthesizerModeDesc که در بسته های javax.speech.recognition و javax.speech.synthesis تعریف شده اند، معین شده است.

علاوه بر شیئ های توصیفگر حالت که در موتورهای گفتار برای توصیف توانایی های آن‌ها به کار می روند، یک برنامه می تواند اشیاء توصیفگر حالت مخصوص خودش را بسازد که نیاز های تابعی مورد نیازش را مشخص کند.

خصوصیات پایه ای برای همه موتور ها در جدول۶-۱ لیست شده اند.

جدول6-1، خصوصیات انتخاب موتور پایه ای EngineModeDesc

نام خصوصیت	توضیحات
EngineName	رشته ای که یک نام را برای موتور گفتار معین می کند. مانند: "Acme Dictation System".
ModeName	رشته ای که وضعیت خاصی را برای عملکرد موتور معین می کند. مانند: "Acme Spanish Dictator".
Locale	یک شیئ از نوع java.util.Locale که زبان مورد استفاده در موتور را معین می کند. همچنین ممکن است کشور و یا استاندارد های یک کشور را نیز معین کند. کلاس Locale از استاندارد ISO 639 برای کد زبان ها و استاندارد ISO 3166 برای کد کشور ها تبعیت می کند. برای مثال کد زیر به معنای مکانی با زبان فرانسوی کانادایی است: Locale("fr"، "ca")
Running	یک شیئ از نوع Boolean که اگر موتور در حال اجرا باشد مقدار TRUE و در غیر این صورت مقدار FALSE می گیرد. با انتخاب یک موتور در حال اجرا می توان از اشتراک گذاری منابع به طور موثرتری استفاده نمود.

یک خصوصیت اضافی در کلاس SynthesizerModeDesc برای موتور های تولید گفتارکننده گفتار معین

شده است:

جدول6-2. خصوصیات انتخاب تولید کننده گفتار SynthesizerModeDesc

نام خصوصیت	توضیحات
List of voices	آرایه ای از صداها که تولید گفتارکننده قادر است تولید کند. هر صدا نمونه ای از کلاس Voice است که دربرگیرنده نام، جنسیت، سن و شیوه سخن گفتن گوینده صدا است.

همچنین دو خصوصیت اضافی تر در کلاس RecognizerModeDesc معین شده است:

جدول6-3. خصوصیات انتخاب تشخیص دهنده RecognizerModeDesc

نام خصوصیت	توضیحات
Dictation supported	یک شیئ از نوع Boolean است که مشخص می کند که آیا تشخیص دهنده‌ی گفتار تشخیص اشکالات گرامری را پشتیبانی می کند یا خیر.
Speaker profiles	لیستی از اشیاء SpeakerProfile برای گویندگانی که تشخیص دهنده را برای تشخیص بهتر صدای خودشان آموزش داده اند. (با تلفظ کردن بعضی از کلمات و جملات درخواستی) اگر تشخیص دهنده از خصوصیت آموزش پشتیبانی نکند مقدار null برگردانده می شود.

هر سه کلاس توصیفگر حالت، EngineModeDesc، SynthesizerModeDesc، و RecognizerModeDesc از الگوهای خصوصیات شبیه JavaBeans استفاده می کنند. مثلا برای خصوصیت Locale، متدهای set و get مانند زیر استفاده می شود:

```
Locale getLocale();
void setLocale(Locale l);
```

۶-۸-۲. محلی کردن، انتخاب کردن و ساختن موتورها

۶-۸-۲-۱. ساختن موتور به صورت پیش فرض[1]

آسان ترین راه برای ساختن یک موتور گفتار، اینست که یک موتور پیش فرض را درخواست کنیم. این کار زمانی مناسب است که یک برنامه موتوری را می خواهد که متناسب با زبان پیش فرض موجود در آن مکان است و هیچ احتیاجات تابعی اضافی برای موتور وجود ندارد. کلاس Central در بسته javax.speech برای محلی کردن و ساختن موتورها استفاده می شود. برای ساخت موتور های پیش فرض از دو تابع استاتیک کلاس Central استفاده می شود:

```
Synthesizer Central.createSynthesizer(EngineModeDesc mode);
Recognizer Central.createRecognizer(EngineModeDesc mode);
```

جدول ۶-۴، کد یک تشخیص دهنده و یک تولید کننده گفتار پیش فرض را نشان می‌دهد.

جدول6-4. ساختن موتور به صورت پیش فرض

```
import javax.speech.*;
import javax.speech.synthesis.*;
import javax.speech.recognition.*;

{
        // Get a synthesizer for the default locale
        Synthesizer synth = Central.createSynthesizer(null);
        // Get a recognizer for the default locale
        Recognizer rec = Central.createRecognizer(null);
}
```

برای هر دوی createSynthesizer و createRecognizer، پارامترهای null نشان می‌دهد که برای برنامه مهم نیست که خصوصیات تولید کننده گفتار یا تشخیص دهنده چه باشد.

اگر برنامه زبان مورد نظر موتور را مشخص نکند، زبان پیش فرض سیستم که توسط تابع زیر نیز قابل

1 Default engine creation

تشخیص است به موتور اعمال می شود:

java.util.Locale.getDefault()

اگر بیش از یک موتور زبان پیش فرض را پشتیبانی کنند، Central موتوری را ترجیح می دهد که در حال اجرا باشد (خصوصیت running آن true باشد). اگر چنین موتوری موجود نبود، موتوری را بر می گزیند که کشور شخص استفاده کننده در مکان های پیش فرض آن تعریف شده باشند.

برای مثال اگر در کشور امریکا باشیم، موتور زبان انگلیسی به عنوان پیش فرض انتخاب می شود و اگر موتور از دو زبان British و American English پشتیبانی کند، دومی انتخاب می شود.

۶-۸-۲-۲. ساختن موتور به صورت ساده[۱]

روش ساده بعدی برای ساخت یک موتور اینست که یک توصیفگر حالت بسازیم، خصوصیات موتور دلخواه را مشخص کنیم و توصیفگر را به متد مناسب برای ساخت موتور در کلاس Centralپاس کنیم. وقتی توصیفگر حالت پاس شده به تابع createSynthesizer یا createRecognizer مقدار null نداشته باشد، موتوری ساخته می شود که با تمام خصوصیات مشخص شده در توصیفگر همخوانی داشته باشد. اگر موتور مناسب موجود نباشد، متد مقدار null را برمی گرداند.

جدول۶-۵، یک تشخیص دهنده دیکته ای را درخواست می کند که مبتنی بر زبان پش‌فرض دستگاه باشد. اگر مقدار null برگشت داده شد، موتور مناسب موجود نیست.

جدول6-5. ساختن موتور به صورت ساده

```
Recognizer createDictationRecognizer()
{
        // Create a mode descriptor with all required features
        RecognizerModeDesc required = new RecognizerModeDesc();
        required.setDictationGrammarSupported(Boolean.TRUE);
```

۱ Simple engine creation

```
        return Central.createRecognizer(required);
}
```

در جدول۶-۶، تولید کننده گفتاری ساخته می‌شود که صدای آن، صدای مردی با زبان اسپانیایی است.

جدول6-6.ساختن موتور به صورت ساده به زبان اسپانیایی

```
/**
  * Return a speech synthesizer for Spanish.
  * Return null if no such engine is available.
  */
Synthesizer createSpanishSynthesizer()
{
        // Create a mode descriptor with all required features
        // "es" is the ISO 639 language code for "Spanish"
        SynthesizerModeDesc required = new SynthesizerModeDesc();
        required.setLocale(new Locale("es", null));
        required.addVoice(new Voice(
                                        null, GENDER_MALE, AGE_DONT_CARE, null));
        return Central.createSynthesizer(required);
}
```

۶-۸-۲-۳. انتخاب موتور به صورت پیشرفته[1]

علاوه بر امکانات ساخت موتور، کلاس Central می تواند لیست هایی از تشخیص دهنده ها و تولید کننده-های گفتار موجود را از طریق دو تابع استاتیک زیر به‌دست دهد:

EngineList availableSynthesizers(EngineModeDesc mode);
EngineList availableRecognizers(EngineModeDesc mode);

اگر به این توابع مقدار null پاس شود، همه موتورهای گفتار موجود برگردانده می شوند. EngineList نوعی بردار است. اگر هیچ موتوری موجود نباشد یک بردار با طول صفر برگردانده می شود نه مقدار null. جدول۶-۷، نشان می‌دهد که چگونه یک برنامه لیستی از موتورهای تولید گفتار صوت به زبان آلمانی را با صدای یک زن بدست می آورد.

1 Advanced engine selection

جدول6-7. ساختن موتور به صورت پیشرفته

```
import javax.speech.*;
import javax.speech.synthesis.*;

// Define the set of required properties in a mode descriptor
SynthesizerModeDesc required = new SynthesizerModeDesc();
required.setLocale(new Locale("de", ""));
required.addVoice(new Voice(
                              null, GENDER_FEMALE, AGE_DONT_CARE, null));

// Get the list of matching engine modes
EngineList list = Central.availableSynthesizers(required);

// Test whether the list is empty - any suitable synthesizers?
if (list.isEmpty()) …
```

۶-۹. حالت های موتور[۱]

رابط Engine مجموعه توابعی را دارد که یک مدیر سیستم حالت تعمیم یافته و کلی را شامل می شود. در این قسمت عملکرد این متدها بررسی می‌شود. در بخش‌های بعدی ما دو سیستم حالت هسته ای که توسط همه موتورها پیاده سازی شده‌اند را مورد بررسی قرار می دهیم. این دو عبارتند از سیستم حالت تخصیص[۲] و سیستم حالت مکث-از سرگیری[۳].

یک وضعیت[۴]، حالت[۵] خاصی از عملکرد یک موتور حالت را تعریف می کند. برای مثال، صف خروجی میان حالت های QUEUE_EMPTY و QUEUE_NOT_EMPTY حرکت می کند. آن چه در ادامه می آید، مبانی مدیریت وضعیت‌ها است.

1 Engine states
2 Allocation state system
3 Pause-resume state system
4 State
5 Mode

تابع getEngineState از رابط Engine وضعیت جاری موتور را بر می‌گرداند. حالت موتور با یک مقدار long نشان داده می شود. بیت‌های مشخص شده از یک وضعیت، نشان می‌دهند که موتور در حالت خاصی قرار دارد. این شیوه نمایش بر مبنای بیت‌ها به این دلیل مورد استفاده قرار می‌گیرد که یک موتور در یک لحظه می-تواند در چند وضعیت قرار داشته باشد.

هر موتور گفتار می‌تواند در یک و فقط یک وضعیت از چهار وضعیت تخصیص قرار داشته باشد. این وضعیت‌ها عبارتند از:

1. DEALLOCATED
2. ALLOCATED
3. ALLOCATING_RESOURCES
4. DEALLOCATING_RESOURCES

وضعیت ALLOCATED خود شامل چند زیروضعیت است. هر موتور ALLOCATED باید در یکی از وضعیت های PAUSED یا RESUMED باشد.

تولید کننده‌های گفتار برای وضعیت‌های صف دارای یک سیستم زیروضعیت جدا هستند. مانند سیستم‌های حالت مکث-از سر گیری، وضعیت های QUEUE_EMPTY و QUEUE_NOT_EMPTY هر دو زیروضعیت-هایی از وضعیت ALLOCATED هستند. علاوه بر این، وضعیت صف و وضعیت مکث-از سر گیری مستقل هستند.

تشخیص دهنده ها سه زیر-وضعیت مستقل در وضعیت ALLOCATED دارند. وضعیت های LISTENING، PROCESSING و SUSPENDED فعالیت جاری پردازه تشخیص را مشخص می کنند. حالت های FOCUS_ON و FOCUS_OFF مشخص می کنند که آیا تشخیص دهنده گفتار در حالت تمرکز بر گفتار قرار دارد یا خیر. برای یک تشخیص دهنده، هر سه زیروضعیتِ وضعیت ALLOCATED مستقلا عمل می‌کنند.

هر کدام از نام های این وضعیت ها با یک مقدار static long، که تنها یکی از بیت ها در آن مقدار یک دارد، مقدار دهی شده‌اند. برای کار با این بیت‌ها می توان از عملگرها بیتی | و & در جاوا استفاده کرد. برای

مثال وضعیت یک تولید کننده گفتار تخصیص داده شده و در حال از سرگیری با یک صف خروجی خالی به صورت زیر مشخص می شود:

(Engine.ALLOCATED | Engine.RESUMED | Synthesizer.QUEUE_EMPTY)

برای تست کردن اینکه یک موتور از سرگرفته شده است یا نه می توان از کد زیر استفاده کرد:

if ((engine.getEngineState() & Engine.RESUMED) != 0) …

برای راحتی، رابط Engine، دو متد اضافی دیگر را برای کنترل کردن وضعیت ها تعریف کرده است. متد testEngineState با دریافت یک وضعیت به عنوان ورودی مشخصی می کند که آیا موتور در آن وضعیت قرار دارد یا خیر (در صورت قرار داشتن مقدار true را بر می گرداند). پس به جای کد بالا می توان کد زیر را نوشت:

if (engine.testEngineState(Engine.RESUMED)) …

به طور تکنیکی استفاده از متد بالا معادل است با

if ((engine.getEngineState() & state) == state)...

آخرین متد وضعیت waitEngineState است. این متد فراخوانی رگه[1] را تا وقتی موتور به حالت خاصی برسد، بلوکه می کند. برا مثال، برای اینکه یک صحبت بایستد چون صف آن خالی شده است از کد زیر استفاده می کنیم:

engine.waitEngineState(Synthesizer.QUEUE_EMPTY);

علاوه بر فراخوانی متدها، برنامه‌ها می توانند وضعیت را از میان سیستم رویداد[2] مانیتور کنند. هر گذار وضعیت با یک EngineEvent مشخص می شوند که توسط رابط SynthesizerListener به Engine وصل می-شوند. کلاس EngineEvent توسط دو کلاس SynthesizerEvent و RecognizerEvent برای گذار وضعیت-هایی که برای آن موتورها مشخص شده‌اند، به ارث برده می‌شود. برای مثال، RecognizerEvent RECOGNIZER_PROCESSING یک گذار از LISTENING به PROCESSING را مشخص می کند وبه این معناست که تشخیص دهنده گفتار را تشخیص می دهد و در حال تولید کردن یک

[1] Thread
[2] Event system

نتیجه است.

۶-۹-۱. سیستم وضعیت تخصیص

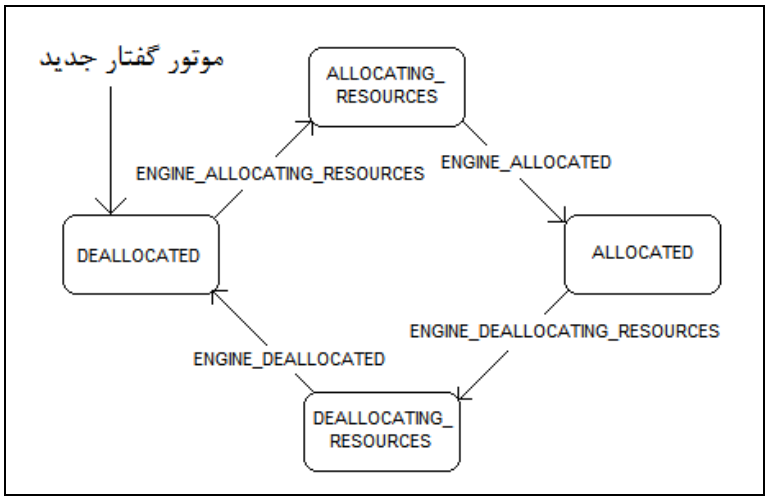
تخصیص موتور پردازه ای است که در آن منابعی که مورد نیاز یک تولید کننده گفتار یا تشخیص دهنده هستند، بدست می آیند. موتورها به محض ساخته شدن، تخصیص داده نمی شوند زیرا موتورهای گفتار می توانند به منابع اساسی (CPU، حافظه یا فضای دیسک) نیاز داشته باشند و ممکن است به منابع انحصاری مانند منبع صدا (مثلا ورودی میکروفن یا خروجی اسپیکر) نیاز داشته باشند. علاوه بر این، تخصیص می تواند برای بعضی از موتورها پروسه بسیار کندی (شاید چند ثانیه یا بیش از یک دقیقه) داشته باشد.

متد allocate از رابط Engine، از موتور می خواهد که تخصیص را انجام دهد و یکی از اولین متدهای مورد استفاده در مورد موتورهای گفتار است. یک موتور که تازه ساخته شده است، در وضعیت DEALLOCATED قرار دارد. فراخوانی متد allocate، یک درخواست از موتور برای رفتن به وضعیت ALLOCATED است .

در حین گذار، موتور موقتا در وضعیت ALLOCATING_RESOURCES قرار می گیرد.

متد deallocate از رابط Engine از موتوری درخواست می کند که منابع تخصیص داده شده را آزاد کند. یک برنامه کارا، تابع deallocate را هر بار که استفاده از یک موتور به پایان می‌رسد، فراخوانی می کند تا منابع برای استفاده برنامه های دیگر آزاد شوند. فراخوانی متد deallocate موتور را به وضعیت DEALLOCATED می برد. در حین گذار، موتور موقتا به وضعیت DEALLOCATING_RESOURCES می رود.

شکل۶-۱ دیاگرام وضعیت سیستم تخصیص وضعیت را نشان می دهد.



شکل6-1. دیاگرام وضعیت سیستم تخصیص وضعیت

هر مستطیل یک وضعیت از موتور را نشان می دهد. یک موتور باید در یکی از چهار وضعیت بالا باشد. در هنگام گذار بین وضعیت ها، رویدادی با نام نوشته شده بر روی منحنی های میان وضعیت ها، از EngineListeners مربوط به موتور صدا زده می شود.

وضعیت عملی نرمال یک موتور وضعیت، وضعیت ALLOCATED است.

۶-۹-۲. وضعیت های تخصیص و بلوکه کردن فراخوانی

برای برنامه های پیشرفته، اغلب دلخواه است که منابع مورد نیاز یک موتور در پشت پرده در رگه دیگری و در هنگامی که بقیه قسمت های برنامه مقدار دهی اولیه می شوند، تخصیص داده شوند و این هنگامی حاصل می‌شود که متد allocate در رگه جداگانه ای فراخوانی شود. جدول۶-۸ را ببینید. برای تعیین این‌که چه زمانی متد تخصیص تمام شده است، در ادامه کد چک می‌شود که موتور در وضعیت ALLOCATED باشد.

جدول6-8. بلوکه کردن فراخوانی

```
Engine engine;
{
        engine = Central.createRecognizer();

        new Thread(new Runnable() {
                public void run() {
```

```
                    try {
                            engine.allocate();
                    }
                    catch (Exception e) {
                            e.printStackTrace();
                    }
            }
    }).start();

    // Do other stuff while allocation takes place
    ...

    // Now wait until allocation is complete
    engine.waitEngineState(Engine.ALLOCATED);
    }
}
```

همچنین برای این‌که در صورت ناموفق بودن برنامه در تخصیص منابع، مشکلی برای برنامه پیش نیاید، این توابع استثنا[1] EngineException را پرتاب می کنند و برنامه نویس می تواند آن‌ها را کنترل کند.

بیشتر متد های Engine، Recognizer و Synthesizer برای اعمال نرمال در وضعیت ALLOCATED معین شده اند. حال اگر فراخوانی برای موتوری که در یک تخصیص دیگر قرار دارد صورت گیرد، چه اتفاقی می افتد؟ برای بیشتر متد ها، عمل به صورتی که در زیر می آید معین شده است:

- وضعیت ALLOCATED: تقریبا برای تمام متد ها رفتار نرمال در این وضعیت تعریف شده اند. (یک استثنا متد allocate است)
- وضعیت ALLOCATING_RESOURCES: بیشتر متد ها در این وضعیت بلوکه می‌شوند. رگه-های فراخوانی تا زمانی که موتور به وضعیت ALLOCATED برود صبر می‌کنند. وقتی آن وضعیت حاصل شد، متد به صورتی که به طور نرمال مشخص شده است به کار خود ادامه می دهد.
- وضعیت DEALLOCATED: بیشتر متد ها برای این وضعیت معین نشده اند، بنابراین

[1] Footnote

EngineStateError پرتاب می شود.

- وضعیت DEALLOCATING_RESOURCES: مانند بالا بیشتر متد ها برای این وضعیت تعریف نشده اند و EngineStateError پرتاب می شود.

تعداد کمی از متد ها در همه وضعیت های بالا به درستی کار می کنند که بعضی از آنها در زیر آمده است:

- getEngineProperties
- getEngineModeDesc
- getEngineState
- testEngineState
- waitEngineState

۶-۹-۳. سیستم وضعیت مکث-از سر گیری

همه موتورهای ALLOCATED وضعیت های PAUSED و RESUMED را دارند. به محض اینکه یک موتور به وضعیت ALLOCATED می رسد، موتور به حالت PAUSED یا RESUMED می رود. فاکتورهایی که بر این حالت اولیه تاثیر می گذارند در زیر توضیح داده شده اند.

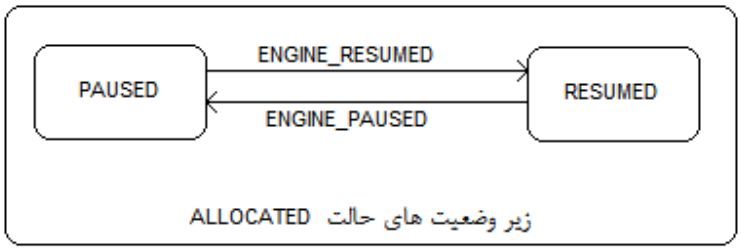
وضعیت PAUSED/RESUMED مشخص می کند که ورودی یا خروجی صدای یک موتور روشن است یا خاموش. یک تشخیص دهنده از سر گرفته شده در حال دریافت یک ورودی صدا است. یک تشخیص دهنده مکث شده، ورودی صدا را نادیده می گیرد. یک تولید کننده گفتار از سرگرفته شده، خروجی صدایی را به منظور سخن گفتن تولید می کند. یک تولید کننده گفتار مکث شده هیچ خروجی صدایی تولید نمی کند.

به عنوان بخشی از سیستم وضعیت موتور، رابط Engine متد های گوناگونی را برای تست وضعیت PAUSED/RESUMED آماده کرده است.

یک برنامه وضعیت PAUSED/RESUMED را توسط متدهای pause و resume کنترل می‌کند. یک برنامه ممکن است به طور نامحدودی باعث مکث یا از سر گیری یک موتور شود. هر بار که وضعیت PAUSE/RESUMED تغییر کرد، یک رویداد ENGINE_PAUSED یا ENGINE_RESUMED از نوع

EngineEvent در EngineListener موتور صدا زده می شوند.

شکل۶-۲، دیاگرام ابتدایی مکث-از سر گیری یک موتور گفتار را نشان می دهد. به عنوان یک سیستم زیروضعیت از وضعیت ALLOCATED ، وضعیت های مکث و از سرگیری در زیروضعیت ALLOCATED در شکل نشان داده شده اند.



شکل6-2. وضعیت های مکث و از سر گیری یک موتور

در شکل۶-۲ بلوک بزرگ‌تر وضعیت ALLOCATED است که در داخل آن دو زیر وضعیت PAUSED و RESUMED قرار گرفته اند.

۶-۱۰. رویداد های گفتار

موتور های گفتار تعداد زیادی از رویداد ها را تولید می کنند. برنامه ها لازم نیست همه آن‌ها را کنترل کنند، هر چند بعضی از آنها اهمیت خاصی برای پیاده‌سازی در برنامه های پردازش گفتار دارند.

رویداد های کتابخانه گفتار جاوا، از مدل رویداد JavaBeans تبعیت می کنند. رویدادها به شنونده[۱] ای ارسال می شوند که به یک شیئ متصل شده است.

همه رویدادهای گفتار از کلاس SpeechEvent از بسته javax.speech مشتق شده اند.

۱ Listener

جدول6-9. رویداد های گفتاری: بسته javax.speech

نام	توضیحات
SpeechEvent	کلاس مادر تمام رویداد های گفتاری.
EngineEvent	نشان می دهد که تغییری در وضعیت موتور گفتار صورت گرفته است.
AudioEvent	نشان می دهد که یک رویداد خروجی یا ورودی صدا رخ داده است.
EngineErrorEvent	زیر کلاسی از EngineEvent است که مشخص می کند یک ایراد ناهمگام در موتور رخ داده است.

رویداد های بسته javax.speech.synthesis در جدول۶-۱۰ آمده اند.

جدول6-10. رویدادهای گفتاری: بسته javax.speech.synthesis

نام	توضیحات
SynthesizerEvent	از EngineEvent مشتق می شود می شود تا رویداد های خاصی از تولید کننده گفتار را توسع دهد.
SpeakableEvent	پیشرفت در خروجی متن تولید گفتار شده را مشخص می کند.

۶-۱۱. بسته javax.speech.synthesis

یک سنتر کننده گفتار موتوری است که متن را به گفتار تبدیل می کند. بسته javax.speech.synthesis، رابط Synthesizer را برای پشتیبانی از تبدیل متن به گفتار تعریف می کند.

بسیاری از متدهای کلاس Synthesizer مانند بقیه موتور های گفتار از رابط Engine ارثبری شده اند که در بالا نیز توضیح داده شد.

در ادامه توضیحات کوتاهی راجع به چگونگی استفاده از تولید کننده گفتار در جاوا می آید.

۶-۱۲. برنامه Hello World!

جدول۶-۱۱، یک استفاده ساده از تولید کننده گفتار را برای گفتن عبارت Hello world نشان می دهد.

جدول6-11. بلوکه کردن فراخواني

```
import javax.speech.*;
import javax.speech.synthesis.*;
import java.util.Locale;

public class HelloWorld {
	public static void main(String args[]) {
		try {
			// Create a synthesizer for English
			Synthesizer synth = Central.createSynthesizer(
				new SynthesizerModeDesc(Locale.ENGLISH));

			// Get it ready to speak
			synth.allocate();
			synth.resume();

			// Speak the "Hello world" string
			synth.speakPlainText("Hello، world!"، null);

			// Wait till speaking is done
			synth.waitEngineState(Synthesizer.QUEUE_EMPTY);

			// Clean up
			synth.deallocate();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

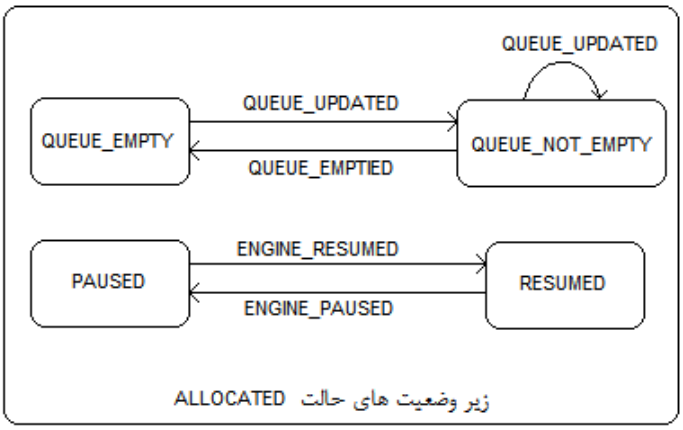
مثال نشان داده شده در بالا دارای سه گام اساسی است که همه برنامه های تولید گفتار صوت بایستی آن ها را انجام دهند:

- ساختن: کلاس Central از بسته javax.speech برای ساختن یک تولید کننده گفتار صوت با استفاده از متد createSynthesizer مورد استفاده قرار می گیرد. این متد قبلا به تفصیل توضیح

داده شد. در این مثال تولید کننده گفتاری که به زبان انگلیسی سخن می گوید، ایجاد شده است.

- تولید: متد speakPlainText گفتار تولید شده از یک رشته را درخواست می کند.
- آزاد سازی منابع (Deallocate): waitEngineState سخنگو را تا رسیدن به وضعیت QUEUE_EMPTY بلوکه می کند تا سخن گفتن آن به پایان برسد. تابع deallocate منابع تولید کننده گفتار را آزاد می کند.

در شکل۶-۳ مهمترین تغییر وضعیت های یک تولید کننده گفتار نشان داده شده است.



شکل6-3. وضعیت های تولید کننده گفتار

آنچه در بالا آمد، مهمترین مفاهیم موجود در یک تولید کننده گفتار بود. با این مقدمه می توان وارد بحث بررسی کد شد و چگونگی استفاده از موتور تولید کننده گفتار را درک تشریح نمود.

۶-۱۳. زبان نشانه گذاری کتابخانه گفتار جاوا[1]

مجموعه عناصر JSML محتوی انواع زیادی از عناصر است. اول، سند JSML می‌تواند حاوی عناصر ساختاری که پاراگراف ها و جملات را مشخص می کند، باشد. دوم اینکه عناصری از JSML وجود دارند که تولید

[1] Java Speech API Markup Language (JSML)

گفتار را کنترل کنند، مثلا تلفظ کلمات و عبارات، استرس و لهجه‌ی کلمات، مکان مرزها و مکث ها، کنترل زیر و بمی و نرخ سخن گفتن[1]. و در نهایت JSML در برگیرنده عناصری است که نشانه گذاری‌های درونی در متن را پشتیبانی می کنند.

برای نمونه در مثال زیر،می توان با استفاده از تگ هایی از نوع تگ های XML، از JSML برای مشخص کردن ساختار جمله برای تاکید روی کلمه can استفاده نمود:

```
<div type="sentence">Computers <emphasis>can</emphasis> speak!</div>
```

6-13-1. اهداف JSML

اصلی ترین اهداف توسعه JSML موارد زیر بودند:

۱. JSML باید با خروجی صدای تولید شده از تولید کننده گفتار، سازگار باشد.
۲. بتوان با استفاده از آن گفتار خروجی را با استفاده از متن تولید کرد.
۳. جهانی سازی شده باشد و بتواند خروجی گفتار را در بسیاری از زبان ها تا حد ممکن تولید کند.
۴. نوشتن و پردازش آن ساده باشد.
۵. برای راحتی، همه امکانات آن تا حد امکان قابل پیاده سازی در برنامه ها و فناوری های موجود باشد.
۶. برای انسان خوانا[2] باشد.
۷. ایجاز در اهمیت کمتری قرار دارد.

۶-۱۳-۲. پردازش اسناد JSML

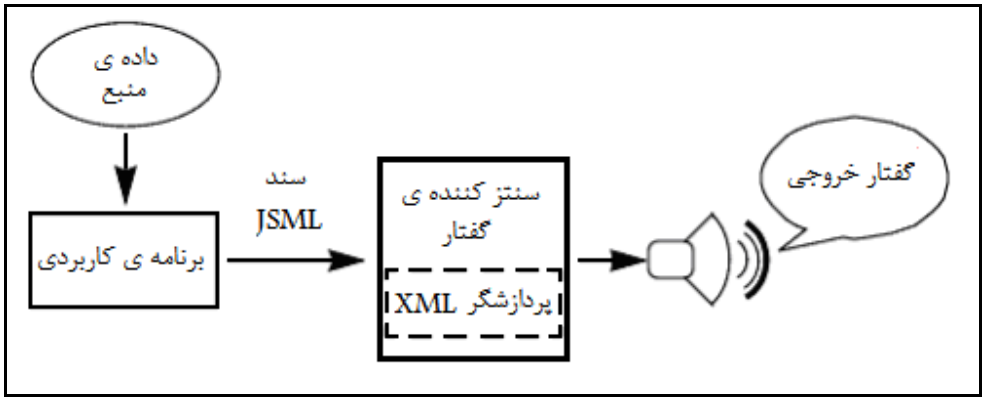
در اصطلاحات XML، یک تولید کننده گفتار یک برنامه XML است و دربرگیرنده یک پردازشگر XML می‌باشد. پردازشگر XML یک سند JSML را می خواند، و عناصر، داده ها و اطلاعات دیگر را از سند استخراج

[1] Speaking rate
[2] Human Legible

می کند. آنگاه تولید کننده گفتارمسئول تفسیر کردن سند و تبدیل آن به صوت است.

یک تولید کننده گفتار، ممکن است به درستی نتواند تشخیص دهد که چگونه یک متن ساده را از انواع مختلفی از منابع مانند کتاب ها، متون تکنیکی، پیام های الکترونیکی، صفحات اینترنتی و ... بخواند. مثلا یک پیام الکترونیکی که مملو از شکلک های قراردادی مانند :) است و بسیار با یک متن عادی متفاوت است، چگونه باید خوانده شود؟

نقش JSML مانند یک نشانه گذار سازگار برای متن های دریافت شده از چنین منابعی است. بنابراین، یک برنامه یا کاربر می تواند یک متن عادی را به آسانی طوری تغییر دهد که سازگار با JSML باشد و به آسانی توسط یک تولید کننده گفتار خوانده شود. شکل۶-۴ مراحل این پردازش را نشان می دهد.



شکل6-4. پردازش اسناد JSML

برای مثال خواندن یک صفحه وب را در نظر بگیرید. داده منبع برای یک صفحه وب معمولا یک صفحه HTML است که ممکن است حاوی داده های صوتی، یا داده هایی که چگونگی جلوه های بصری یا صوتی آن را مشخص می کند نیز، باشد. معمولا صفحات وب با استفاده از مرورگرهای اینترنتی نمایش داده می شوند. برای رندر کردن بصری، مرورگر یک نمایشگر گرافیکی را کنترل می کند تا کاراکتر ها و تصاویر را نمایش دهد. برای ارائه کردن صوتی، مثلا خواندن آن، مرورگر یک تولید کننده گفتار را کنترل می کند تا متن ها توسط آن خوانده شوند.

یک مثال معمول دیگر خواندن یک پیام الکترونیکی است. وقتی یک خواننده پستی[۱]، یک پیام الکترونیکی را به متن خوانده شده تبدیل می کند، می تواند سرآیند آن پیام الکترونیکی را تشخیص دهد یا مواردی مانند تاریخ، ساعت یا آدرس الکترونیکی را در درون پیام تشخیص دهد و آنها را به طور واضحی بخواند. برنامه پیام الکترونیکی می تواند پردازش های خاصی را نیز بر متن بدنه پیام انجام دهد. مثلا می تواند فایل های پیوست شده، متن های تو رفته، مخفف های پرکاربرد در پیام های الکترونیکی و غیره را تشخیص دهد. در جدول۶-۱۲، مثالی از یک پیام الکترونیکی تبدیل شده به JSML را آمده است.

جدول6-12. مثالی از یک پیام الکترونیکی

```
<jsml>
<div type="paragraph">Message from
<emphasis>Alan Schwarz</emphasis> about new synthesis
technology. Arrived at <sayas class="time">2pm</sayas>
today.</div>
<div type="paragraph">I've attached a diagram showing the new way we do speech synthesis.</div>
<div>Regards، Alan.</div>
</jsml>
```

6-13-3. عناصر JSML

یک سند JSML یک عنصر ریشه ای را در بر می گیرد که می تواند یکی از موارد ساختاری[۲]، تولید[۳] و متفرقه[۴] باشد. همه عناصر JSML برای مهیا کردن یک تولید کننده گفتار با اطلاعاتی در مورد چگونگی خواندن متنی که درون آن‌ها قرار گرفته اند، طراحی شده اند. جدول۶-۱۳، کلیاتی را راجع به عناصر JSML نشان می دهد.

---

[۱] Email reader

[2] Structural

[3] Production

[4] Miscellaneous

جدول6-13. عناصر اصلي JSML

عملکرد عنصر	نام عنصر	نوع عنصر	توضیح عنصر
ساختار	jsml	ظرف یا Container	عنصر ریشه ای برای سند های JSML
	div	ظرف	ساختار های محتوایی متن مانند پاراگراف و جمله را مشخص می کند.
تولید	voice	ظرف	یک صدای گوینده در حال صحبت را برای متن در بر گرفته شده مشخص می کند.
	sayas	ظرف	مشخص می کند که متن در بر گرفته شده چگونه خوانده شود.
	phoneme	ظرف	مشخص می کند که متن در بر گرفته شده یک رشته واجی است.
	emphasis	ظرف	تکیه را برای متن در بر گیرنده مشخص می کند و آن متن با تکیه بیشتری تلفظ می شود.
	break	خالی	یک وقفه را در گفتار مشخص می کند.
	prosody	ظرف	یک مالکیت عروضی را برای متن در بر گرفته شده مشخص می کند. مانند زیر و بمی، نرخ صحبت (تندی و کندی) یا حجم.
متفرقه	marker	خالی	یک اطلاع را وقتی گفتار به این نشان می‌رسد، درخواست می کند.
	engine	ظرف	دستورالعمل محلی برای یک تولید کننده گفتار ی گفتار خاص.

6-13-4. مثالی از JSML

در جدول۶-۱۴ مثالی از JSML دیده می‌شود.

جدول6-14. مثالي از JSML

```
<?xml version="1.0"?>
<!--
```

```
/** * Copyright 2001 Sun Microsystems، Inc.
 *  * See the file "license.terms" for information on usage and
 * redistribution of this file، and for a DISCLAIMER OF ALL
 * WARRANTIES.
 */-->
<jsml lang="en.us">

<engine name="guisynthesizer،sapiSynth" data="For GUI and SAPI synthesizers." mark="engineTag">For all other synthesizers.</engine>

<break time="5s"/>
<marker mark="MARKER TEST"/>

<!-- comment -->

<!-- Make sure that non-JSML elements are ignored -->
<test>here</test>

<!-- Test division and sayas elements (though not all
capabilities of either of them) -->
<div type="para">
Today's date is <sayas class="date" mode="mdy">1/7/99</sayas>.
</div>

<!-- Simple phoneme element -->
The word of the day is <phoneme>kro:k</phoneme>.

<!-- Simple examples of CDATA -->
Send email to <![CDATA[<joe.doe@acme.com>]]>.

<!-- CDATA vs. XML element -->
<![CDATA[<greeting>Hello، world!</greeting>]]>
```

```
<greeting>Hello، world!</greeting>

<!-- Try some prosodic controls -->
<prosody pitch="">This is a <emphasis level="strong" mark="atEmp"> new test
 of <break/> <prosody rate="-10%">emphasis</prosody></emphasis>.</prosody>
<prosody pitch=" +20 % " rate="+20    " range="+12.2">This is a test.</prosody>
<prosody volume="1.5" pitch="205">This <prosody pitch="reset">is a </prosody>test.</prosody>

<!-- Final simple test -->
<div type="para" mark="!HERE!">This is a test.</div>
</jsml>
```

۶-۱۴. انتخاب پیاده سازی مناسب برای کتابخانه گفتار جاوا

همانظور که در بالا نیز اشاره شد، پیاده‌سازی‌های گوناگونی از کتابخانه گفتار جاوا وجود دارند که در این بخش به بررسی و مقایسه اجمالی آنها پرداخت شده است.

۶-۱۴-۱. پیاده سازی های تولید کننده گفتار

۶-۱۴-۱-۱. پیاده سازی FreeTTS

FreeTTS که همان کتابخانه استفاده شده در این پروژه است، یک پیاده سازی متن باز از JSAPI است که به طور کامل به زبان جاوا نوشته شده است. این پیاده سازی بر مبنای Flite، که یک تولید کننده گفتار است، کار می کند. Flite در دانشگاه کارنگی ملون[۱] توسعه داده شده است. FreeTTS یک پیاده سازی کامل از JSAPI نیست زیرا بسته تشخیص گفتار (javax.speech.recognition) را پیاده سازی نکرده است.

از مهمترین خصوصیات این کتابخانه می توان به موارد زیر اشاره کرد:

- بسته استاندارد که با سه صدا می آید (دو صدای مرد و یک زن)

[۱] Carnegie Mellon

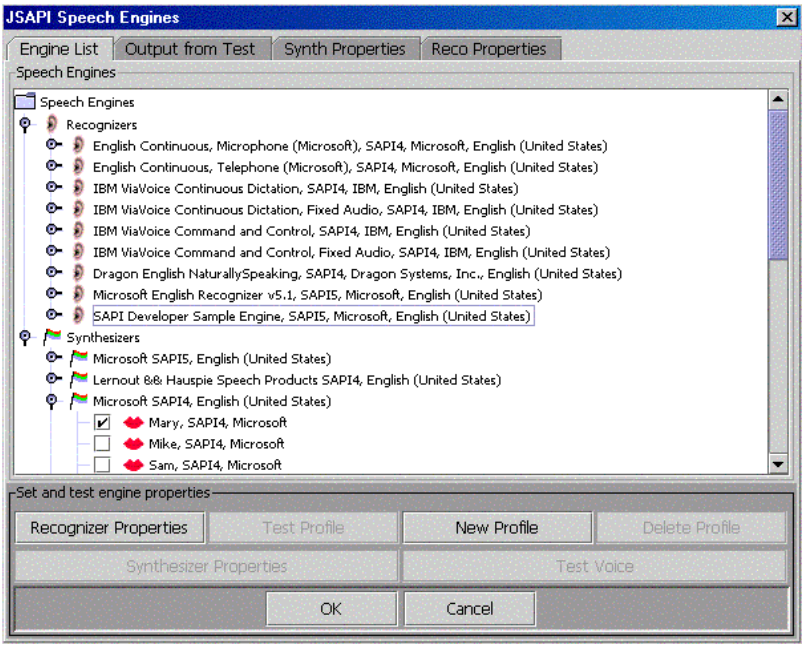
- صدا های بیشتر برای زبان انگلیسی که بر پایه پروژه FestVox باشد قابل تبدیل و اضافه شدن بـه این پروژه است.
- از صداهای پروژه متن باز MBROLA پشتیبانی می شود.
- عملکزد بهتری نسبت به Flite بر روی بیشتر بستر ها دارد.

یکی از بزرگترین مشکلات FreeTTS پشتیبانی نکـردن از JSML یـا Java Speech Markup Language است. این کتابخانه داده های JSML را پردازش می کند اما آن را به صورت یک متن ساده در می آورد.

۶-۱۴-۱-۲. پیاده سازی Talking Java SDK

این پیاده سازی یکی از بهترین و کامل ترین پیاده سازی های کتابخانه گفتار جاوا است که توسط شـرکت CloudGarden صورت گرفته است. مهمترین خصوصیات این پیاده سازی عبارتند از:

- پیاده سازی کامل کتابخانه گفتار جاوا (هم تولید کننده گفتار و هم تشخیص دهنده)
- پشتیبانی از محدوده بسیار وسیعی از زبان ها
- تطابق با طیف وسیعی از موتور های تولید گفتار و تشخیص گفتار SAPI4 و SAPI5
- رابط کاربری بسیار قوی (شکل۶-۵)
- آموزش موتور تشخیص توسط کاربر
- گفتار تشخیص داده شده و تولید شده می توانند به راحتی به صورت یک فایل صوتی ذخیره شوند
- تشخیص دهنده گفتار توانایی تشخیص گوینده، از میان چند گوینده که از برنامه استفاده می‌کنند را دارد.
- نمایش گویش انسانی در این پیاده سازی وجود دارد، به گونه ای که تصویری از لب های انسان در هنگام سخن گفتن تولید کننده گفتار نشان داده می شود و حرکات آن کـاملا متناسـب بـا مـتن خوانده شده است.



شکل6-5. رابط کاربری Talking Java SDK

- می تواند در appletجاوا و plugin استفاده شود [۳۲].

۶-۱-۱۴-۳. پیاده سازی JSpeechLib

این کتابخانه، یک پیاده سازی قدیمی است که تنها بر روی سیستم عامل Mac OS 9 و بعد از آن قابل اجرا است. این برنامه موارد زیر را شامل می شود:

- پیاده سازی کتابخانه گفتار جاوا
- مدیریت گفتار Mac (Mac Speech Manager)
- مثال هایی برای javax.speech.synthesis

این بسته حاوی کلاس هایی است که با استفاده از مدیریت گفتار مکینتاش یا MacinTalk که به طور پیش‌فرض در روی کامپیوتر های مکینتاشی که در چند سال اخیر فروخته می شود نصب شده اند، کتابخانه گفتار جاوا را پیاده سازی می کند[۲].

با این اوصاف این کتابخانه در سطح بسیار محدودی قابل استفاده است. همچنین این کتابخانه از پروانه

LGPL استفاده می کند که تنها می توان از آن در پروژه هایی استفاده نمود که پروانه آنها متناسب با این پروانه باشد.

۶-۱-۱۴-۴. پیاده سازی IBM Speech API

یک پیاده سازی از شرکت IBM است که در سال ۱۹۹۸ پایان کار خود را اعلام کرد و دیگر توسعه داده نشد.

IBM Speech for Java 1.0، توانایی دسترسی به تشخیص دستور، دیکته و تبدیل متن به گفتار را فراهم می‌کرد[۳۱].

۶-۱-۱۴-۵. پیاده سازی Lernout & Hauspie's TTS for Java Speech API

یک پیاده سازی بر پایه موتورهای ASR1600 و TTS3000 است که خصوصیات "دستور بده و کنترل کن" و تولید گفتار را پشتیبانی می کند. این پیاده سازی، ۱۰ صدای مختلف را برای زبان انگلیسی پشتیبانی می-کند. همچنین توانایی کنترل زیر و بمی صدا و برد آن، نرخ سخن گویی[1] و حجم صدا را دارد [۳۵].

۶-۱-۱۴-۶. پیاده سازی Conversa Web 3.0

یک مرورگر اینترنت با توانایی صوتی است که برد وسیعی از امکانات گشت و گذار در وب را با استفاده از تشخیص و تولید گفتار فراهم می کند. توسعه دهندگان این مرورگر از یک پیاده سازی خود ساخته از کتابخانه گفتار جاوا استفاده می کنند. این پیاده‌سازی تنها در محیط ویندوز قابل استفاده است [۳۵].

۶-۱-۱۴-۷. پیاده سازی Elan Speech Cube

یک مؤلفه[2]ی نرم افزاری سیستم متن به صوت چند زبانه، چند کاناله و چند سیستم عامله برای معماری های سرویس دهنده-مشتری است. این پیاده سازی با دو تکنولوژی متن به صوت و ۱۱ زبان پر کاربرد قابل استفاده است.

[1] Speaking rate
[2] Component

این پیاده سازی از JSML/JSAPI[1] پشتیبانی می کند. همچنین از نسخه سوم جاوا استفاده می‌نماید و روی سیستم عامل های لینوکس، ویندوز و سولاریس شرکت سان مایکروسیستم قابل اجرا است [۳۵].

۶-۱۴-۱-۸. پیاده سازی Festival

Festival یک سیستم تبدیل متن به گفتار چند زبانه و چند کاربرده است که توسط مرکز تحقیقات فناوری گفتار در دانشگاه ادین برو[2] توسعه داده شده است. Festival یک سیستم متن به صوت کامل را با API های گوناگون و نیز محیطی برای توسعه و تحقیق فناوری های تبدیل متن به گفتار فراهم می‌کند.

هسته این کتابخانه با زبان C++ توسعه داده شده است و سپس برای کتابخانه گفتار جاوا آماده شده است. در Festival زبان های انگلیسی (بریتانیایی و آمرکایی)، اسپانیایی و ولزی پشتیبانی می شوند.

این پیاده سازی در بیشتر سیستم عامل های یونیکس (SunOS، Solaris، FreeBSD، لینوکس، SGI و ...) پشتیبانی می شود. (البته به صورت ابتدایی تر در ویندوز نیز پشتیبانی می شود) [۳۳].

۶-۱۴-۲. پیاده سازی های تشخیص دهنده گفتار

۶-۱۴-۲-۱. پیاده سازی Sphinx 4

Sphinx 4 یک سیستم تشخیص گفتار جذاب است که با جاوا نوشته شده است. این سیستم نیز توسط دانشگاه کارنگی ملون توسعه داده شده است. نسخه های قبلی با زبان C پیاده سازی شده بودند. این سیستم نیز یک سیستم متن باز است.

مهمترین خصوصیات این موتور عبارتند از:

- برد وسیعی از فرمت های گرامری مانند Java Speech Grammar Format، SimpleWorldListGrammer، LMGrammer و FSTGrammer را پشتیبانی می کند.
- تشخیص گفتار پیوسته و محدود به یک گرامر خاص

[1] Java Speech Markup Language
[2] Centre for Speech Technology Research at the University of Edinburgh

- گنجینه لغت وسیع
- پشتیبانی جزئی از JSAPI
- پشتیبانی کامل از فرمت گرامری گفتار جاوا[1] (JSGF)
- استفاده از جاوای ۴
- پشتیبانی از آموزش به صورت مدل های جدید صوت شناسی
- پشتیبانی از مدل های زبان های جدید و ساخت کاربر
- پشتیبانی از افزودن لغت نامه های جدید

پیاده سازی Talking Java SDK

۶-۱۵. نتیجه گیری

کتابخانه پردازش گفتار جاوا از دو بخش مهم تشکیل شده است. این دو بخش عبارتند از تبدیل متن به گفتار و تشخیص گفتار. این کتابخانه توانایی ارسال متن به یک تابع به صورت متن عادی یا زبان نشانه گذاری JSML را برای ما فراهم می سازد. و متن بعد از تولید گفتار به صورت گفتار در می آید. در بخش تشخیص گفتار، موتور گفتار توسط کاربر آموزش داده می شود و سپس تشخیص گفتار در مراحل بعدی صورت می گیرد. در این سیستم تصحیح غلط های دیکته ای نیز پیش بینی شده است.

از میان ۹ پیاده سازی نام برده شده در طول بخش ۶-۱۵، ۶ پیاده سازی FreeTTS، TalkingJava SDK، JSpeechLib، Festival، Sphinx 4 و IBM از بقیه مهمتر و پر کاربردتر می باشند. برای انتخاب پیاده سازی مناسب توانایی های متفاوتی از آنها باید با یکدیگر مقایسه شوند تا انتخاب درستی صورت گیرد.

از مهمترین ویژگی های موتورهای گفتار می‌توان به توانایی گفتار با صداهای متفاوت، پشتیبانی از زبان‌های

[1] Java Speech Grammar Format

گوناگون، اضافه کردن صداها و گوینده هایی توسط کاربر، پشتیبانی از زبان های گوناگون، پشتیبانی از JSML برای تولید گفتار صوت، توانایی آموزش موتور برای تشخیص گفتار و توانایی تصحیح دیکته ای گفتار تشخیص داده شده، اشاره کرد.

در جدول۶-۱۵ مقایسه ای بین ۶ پیاده سازی همه‌گیر کتابخانه گفتار جاوا صورت پذیرفته است. همانطور که در جدول نیز به وضوح دیده می شود، قدرتمند ترین پیاده سازی برای کتابخانه گفتار جاوا TalkingJava SDK می باشد.

جدول6-15. مقایسه میان شش پیاده سازی مهم کتابخانه گفتار جاوا

Sphinx	Festival	IBM	JSLib	TJSDK	FTTS	
û	ü	ü	ü	ü	ü	تبدیل متن به گفتار
ü	ü	ü	ü	ü	û	تشخیص گفتار
ü	ü	û	ü	ü	ü	گویش های متفاوت
û	ü	û	ü	ü	ü	چند زبانی
û	ü	û	û	û	ü	اضافه نمودن گوینده توسط کاربر
ü	û	û	û	ü	û	استفاده از JSML
ü	ü	û	û	ü	û	آموزش موتور تشخیص
ü	ü	ü	û	ü	û	غلط های دیکته ای
û	؟	û	û	ü	ü	ذخیره سازی متن یا صوت تولید شده در فایل
ü	û	ü	؟	ü	û	استفاده در Applet ها
ü	ü	ü	û	ü	ü	استفاده در چند سیستم عامل
ü	ü	û	û	û	ü	متن باز

فصل آتی به بررسی تعدادی از استفاده‌های عملی سیستم‌های تبدیل متن به گفتار، می‌پردازد.

فصل 7
استفاده از سیستم‌های تبدیل متن به گفتار در کاربردهای عملی

۷-۱. مقدمه

در فصول قبل استاندارد های تحت وب مورد استفاده در سیستم های صوتی را به ترتیب بررسی شدند. در این فصل چند مورد از نمونه های کاربردی ساخته شده بر اساس این استاندارد را مرور می‌شود.

استانداردهای منتشر شده توسط کنسرسیوم وب جهانی به منظور استاندارد سازی سیستم های دیالوگ و مرورگر های وب هستند و استفاده از آن ها در این سرویس ها مزایایی را در بر خواهد داشت . یکی از این مزایا کاهش هزینه توسعه سیستم های دیالوگ، و افزایش قابلیت انتقال این گونه سیستم ها می باشد . همچنین قرار گرفتن در جایگاه خوب بازار برای سیستم هایی که از این استاندارد ها استفاده می کنند، باعث شده است که توسعه دهنده های مختلف از آن ها را در سرویس های خود استفاده کنند.

در ادامه ابتدا یک نمونه کاربردی که در محیط‌های آموزشی مورد استفاده قرار می گیرد آورده شده است. سپس یک پروژه دیگر که در سیستم بانکی مورد استفاده قرار گرفته و یک شبیه ساز که بر پایه VoiceXML ساخته شده است، بررسی می‌شود. در پایان نیز چند سیستم دیالوگ چند زبانه از جمله عربی ، چینی ، انگلیسی مرور شده است.

۷-۲. سیستم دیالوگ چند زبانه در محیط آموزشی

یکی از سیستم هایی که از استانداردهای نامبرده شده در زمینه توسعه خود استفاده کرده است، سیستم دیالوگ چند حالته ، چند زبانه برای فعل و انفعال متقابل در یک فضای آموزشی است که در پروژه UCAT[1] انجام شده است . هدف از این سیستم کمک به معلمان و دانش آموزان در فعالیت های معمولشان در یک فضای آموزشی است (برای مثال در یک دانشکده دانشگاه).

[1] Ubiquitous Collaborative Adaptive Training

فعل و انفعالات بین کاربر و سیستم توسط گفتار، متن و گرافیک با به کاربری مستقیم زبان های اسپانیایی و یا انگلیسی است.

سیستم قابلیت تطبیق با سلیقه‌ها و ویژگی‌های از پیش ذخیره شده کاربران در پایگاه داده پروفایل کاربران را دارد . محاسبات Ubiquitous یک تکنولوژی است که برای پیشنهاد دادن فرصت ها و چالش هایی که برای به کار بردن رابط های کاربر جدید در ارتباط با محیط لازم است به وجود آمده است. در بین دیگر فیلد برنامه ها این تکنولوژی برای محیط های آموزشی به کار برده می شود.

برای مثال در پروژه کلاس ۲۰۰۰ یک سیستم برای ذخیره فعالیت های استاد روی تخته سیاه توسعه پیدا کرد که دسترسی دانش آموزان را از طریق صفحات وب میسر می سازد. استاد از یک تخته الکترونیکی برای توضیحات خود استفاده می کند و از اسلاید و وسایل اضافی دیگر نیز بهره می برد. تمام فعالیت های معلم توسط یک سیستم Ubiquitous همراه با زمان آن ها ذخیره می شوند و برای دانش آموزان موجود خواهند بود.

در این سیستم از استاندارد های W3C برای انجام فعل و انفعالات و تولید دیالوگ ها استفاده شده است. برای مثال خروجی سیستم، تبدیل متن به گفتار توسط جملاتی که در برچسب های <prompt> استفاده شده در VoiceXML وجود دارند تولید می شود.[۲۴]

۳-۷. پروژه GEMINI

یکی دیگر از موارد استفاده از استانداردهای ارائه شده توسط W3C ، پروژه GEMINI[1] است که دو هدف اساسی دارد. اول توسعه یک زمینه انعطاف‌پذیر که قادر به تولید رابط های دیالوگ چند زبانه و چند حالته فعل و انفعال کاربر با حداقل تلاش انسانی باشد. هدف دوم اثبات کارایی زمینه بوسیله توسعه دو برنامه کاربردی مختلف بر اساس این زمینه است. برنامه اول، EG_Banking است که یک پرتال صدا برای فعل و انفعال با کیفیت بالا

[1] Generic Environment for Multilingual Interactive Natural Interfaces

برای مشتریان بانک می‌باشد. برنامه دوم یک CitizenCare برای ارتباط شهروندان با وزارت خانه‌های دولتی است که برای ارتباط کاربر بر پایه وب و گفتاری موجود است. سیستم باید قادر به تولید اتوماتیک دیالوگ های لازم برای اجرای سرویس باشد.

فعالیت اصلی EG_Banking شامل یک قسمت اطلاعات عمومی (کارت های تجاری ، حساب‌ها) که برای قسمت عمومی موجود است و یک بخش نقل و انتقالات که تنها برای مشتریان بانک موجود است . برنامه های چند زبانه از طریق شبکه تلفنی ثابت و سلولی قابل دستیابی هستند . یک نسخه دستی برنامه های تولید شده در بانک Egnatia در یونان نصب شده است و به عنوان یک بانک تلفنی مورد استفاده قرار می گیرد.

یکی از برتری های این سیستم در مقایسه با سیستم های مشابه استفاده از زبان های توصیفی استاندارد مثل VoiceXML برای تولید دیالوگ ها است بنابراین هیچ نیازی برای توسعه یک سیستم در حال اجرا مخصوص وجود ندارد .یکی دیگر از مزایای استفاده از VoiceXML این است که اجرای دیالوگ های تولید شده با همه مفسرهای VoiceXML را میسر می سازد و همچنین استفاده از این استانداردها باعث پیدا کردن یک جایگاه خوب در بازار برنامه های صوتی این سیستم ها می شود[۲۵].

۴-۷. شبیه ساز کاربر بر پایه VoiceXML

در این قسمت برای ارزیابی سیستم های دیالوگ گفتاری، یک شبیه ساز کاربر بر پایه VXML بررسی می‌شود. یک شبیه ساز کاربر یک روش برای ارزیابی سیستم های دیالوگ گفتاری بدون استفاده از ارزیاب های انسانی است.

ویژگی جدید این شبیه ساز این است که از توصیفات VoiceXML که رفتار سیستم های دیالوگ را توضیح می‌دهند، استفاده می‌کند. با استفاده از توصیفات VoiceXML، شبیه ساز پیشنهادی می تواند برای هر سیستم دیالوگی که با VoiceXML کار می‌کند استفاده شود. نتایج آزمایشی نشان می دهد که دیالوگ بین شبیه ساز و یک سیستم دیالوگ خواص مشابهی با دیالوگ های بین انسان و سیستم های دیالوگ دارد.

برای توسعه دادن یک سیستم دیالوگ واقعی اجرای آزمایشات زیاد اجتناب ناپذیر است. برای یک سیستم دیالوگی که برای ارتباط با انسان طراحی شده است ، اجرای این آزمایشات زیاد به تعداد زیادی نیروی انسانی نیاز دارد که این خود نیازمند صرف هزینه و وقت بسیار است.

بنابراین برای کاهش تعداد آزمایشات انسانی، ارزیابی شبیه سازی سیستم دیالوگ گفتاری پیشنهاد داده شده است. Lopez Cozar یک شبیه ساز کاربر را برای ارزیابی سیستم دیالوگ SAPLEN پیشنهاد کرد که دیالوگ های مربوط به سفارشات در یک رستوران غذا را اجرا می‌کرد[۲۶].

شبیه ساز کاربر پیشنهاد شده در این مقاله از یک تولید کننده گفتار برای تولید صداهای کاربر شبیه سازی شده استفاده می کند. علاوه بر این شبیه ساز از توصیفات VXML که رفتار سیستم های دیالوگ گفتاری را برای ارزیابی توضیح می دهند، بهره می برد.

رفتار شبیه ساز کاربر به این شکل است، ابتدا شبیه ساز فایل VoiceXML را، که رفتار سیستم دیالوگ گفتگو را توضیح می دهد، می خواند. سپس توصیفات را تحلیل می کند و اطلاعات زیر را جمع آوری می کند:

- تمام متغیرها و فرم ها بر طبق قلم هایی که باید پر شوند.
- تمام گرامرهایی که برای تشخیص صدا استفاده می شوند.

در این مرحله شبیه ساز برای فرمان سیستم دیالوگ منتظر می ماند. وقتی که سیستم دیالوگ اولین پیغام فرمان را تلفظ می کند، شبیه ساز هدف دیالوگ را با استفاده از متغیرها و گرامرهای توصیفات دیالوگ نوشته شده در VoiceXML تولید می کند.

وقتی که سیستم دیالوگ برای مقدار یک قلم از کاربر سوال می کند، شبیه ساز آن را بر اساس هدف دیالوگ تولید و تلفظ می کند. وقتی که سیستم دیالوگ برای تصدیق سوال می کند ، شبیه ساز مقادیر متغیرها را در سیستم دیالوگ با هدف دیالوگ مقایسه می کند. اگر قلم های تشخیص داده شده با قلم های موجود در هدف دیالوگ متفاوت باشند، شبیه ساز تلفظ های صحیح را تولید می کند. در غیر این صورت شبیه ساز 'yes' را تلفظ می کند و دیالوگ خاتمه پیدا می کند.

۱-۴-۷. توصیفات VoiceXML

در ادامه مثال هایی را از فایل VoiceXML آورده شده است. این مثال سفارش یک نوشیدنی را نمایش می دهد. در این مثال سیستم دیالوگ نظر کاربر را در مورد نوشیدنی و مقدار سوال می کند. هر قلم توسط یک <field> مشخص می شود و پرسش ها در درون یک تگ <form> قرار می گیرند. تگ <prompt> پیغام فرمان را برای تولید کردن گفتار، نشان می دهد و تگ <grammar> گرامری که برای انطباق ورودی کاربر به کار می‌رود مشخص می کند.

```
<form id="beverage_service">
<field name="beverage">
<prompt>
What would you like to drink?
</prompt>
<grammar src="drink_grammar.gsl"/>
</field>
<field name="num">
<prompt>How many cups?</prompt>
<grammar src="num_grammar.gsl"/>
</field>
<filled>
<goto next="#yesno">
</filled>
</form>
```

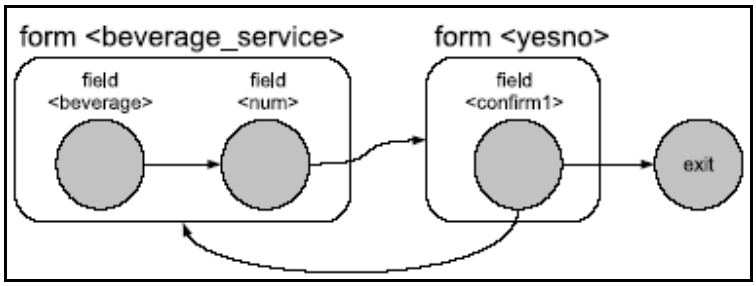
در زیر مثالی از تصدیق آورده شده است. در این مثال، پیغام در تگ <prompt> کاربر را با نتایج تشخیص نشان می دهد، و سیستم دیالوگ برای صدای تصدیق کاربر منتظر می ماند. زمانی که تصدیق مورد قبول واقع شود ، حالت دیالوگ (تگ <if>) بر طبق سخن کاربر انشعاب پیدا می کند.

```
<form id="yesno">
<field name="confirm1">
```

```
<prompt>
I will bring <value expr="num">
<value expr="beverage">. OK?
</prompt>
<grammar src="confirm_grammar.gsl"/>
</field>
<filled>
<if cond="confirm1 == yes">
<prompt>
OK. I will bring <value expr="num">
<value expr="beverage">.
</prompt>
<exit/>
<else/>
<goto next="#beverage_service"/>
</if>
</filled>
</form>
```

زمانی که پاسخ 'yes' باشد سیستم دیالوگ خارج می شود و مقادیر متغیرها (در این مثال نوع نوشیدنی و تعداد) به سیستم نهایی فرستاده می شوند.

جریان دیالوگ نوشته شده در VoiceXML می تواند به عنوان یک آتاماتای حالت متناهی نمایش داده شود. آتاماتا بر طبق مثال بالا در شکل۷-۱ نشان داده شده است. تگ فیلد مطابق با حالات در آتاماتا است و تگ‌های فرم منطبق با گروه های حالات. زمانی که سیستم دیالوگ سخن کاربر را قبول می کند، یک انتقال رخ می دهد[۲۷].



شکل7-1. انتقال حالت ها در نمونه VXML

۷-۵. توسعه سیستم دیالوگ گفتگو بر پایه GALAXY/VoiceXML

این سیستم اولین سیستم دیالوگ زبان اسلواکی است. سیستم پیشنهادی شامل تشخیص گفتار اتوماتیک ، گفتار به متن و ماژول مدیریت دیالوگ VoiceXML است. این سیستم ارتباط چند لایه را در زبان اسلواکی میسر می کند.

کارکرد سیستم از طریق دو برنامه آزمایشی تست شده است، " پیش‌بینی آب و هوا برای اسلواکی" و " جدول زمان بندی راه آهن اسلواکی. اطلاعات مورد نیاز از منابع اینترنت در وضعیت چند کاربر بازیابی می شود.

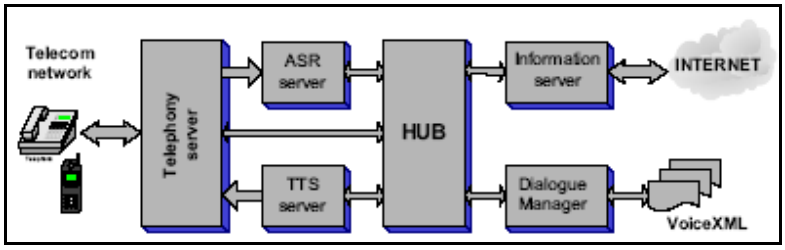
بر طبق پیشرفت های تشخیص گفتار و فهم گفتار، سیستم های دیالوگ گفتار به عنوان یک جایگزین کاربردی برای یک رابط کامپیوتر محاوره تبدیل شدند. از آن جایی که این سیستم ها ارتباطات آزادتر و طبیعی-تری را میسر می سازند و با حالت های مختلف ورودی و خروجی بصری قابل ترکیب هستند، این سیستم ها بسیار موثرتر از سیستم های IVR هستند.

این سیستم در فاصله سال‌های ۲۰۰۳ تا ۲۰۰۶ توسعه پیدا کرده است. هدف اصلی این پروژه در تحقیق و توسعه یک سیستم دیالوگ برای بازیابی اطلاعات با استفاده ارتباطات مابین انسان و کامپیوتر است.

۷-۵-۱. معماری سیستم

این سیستم شامل یک هاب و ۶ ماژول سیستم است : ماژول تلفن، ماژول تشخیص گفتار اتوماتیک، ماژول

تبدیل متن به گفتار، ماژول انتقال و ماژول انتهایی مدیریت دیالوگ. رابطه بین مدیر دیالوگ، هاب Galaxy و دیگر ماژول های سیستم در شکل۷-۲ نمایش داده شده است.

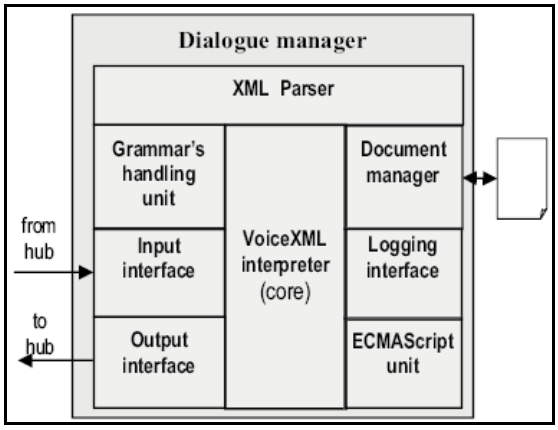


شکل7-2. معماری سیستم دیالوگ Galaxy

۷-۵-۲. مدیر دیالوگ

دستاوردهای متعددی برای چگونگی حل واحد مدیر دیالوگ و همچنین زبان های متعدد برای نوشتن کد آن وجود دارد. VoiceXML برای تولید رابط های کاربر صوتی که از تشخیص گفتار اتوماتیک و تبدیل متن به گفتار استفاده می کنند، استفاده می‌شود . تعداد زیادی از برنامه های کاربردی بر پایه VoiceXML تجاری روی محدوده وسیعی از صنعت و سرویس های مالی، دولتی، بیمه، ارتباطات تلفنی، سفر و بیمارستان ها وجود دارد.

اجزای اصلی مدیر دیالوگ در شکل۷-۳ نشان داده شده است[۲۸].



شکل7-3. اجزای اصلی واحد مدیریت دیالوگ بر پایه VXML

۷-۶. مرورگر صوتی HearSay

سیستم مرورگر صوتی Hearsay، سیستمی است که دستیابی کارآمدی را به وب برای افرادی با عدم توانایی بینایی فراهم می کند. Hearsay شامل جداسازی بر اساس محتوای صفحات وب و یک رابط مشتق گفتاری برای محتوای نتیجه است.

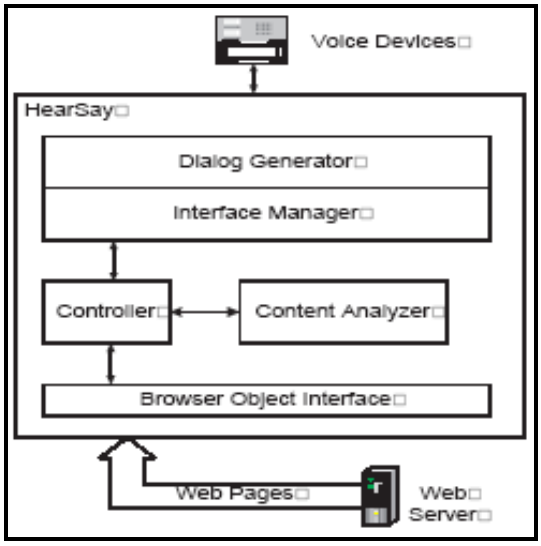
وب جهانی به یک منظر ضروری در جامعه تبدیل شده که برای تحصیلات ، تجارت ، پزشکی، سرگرمی و ... استفاده می شود. وسیله اصلی دستیابی به وب از طریق حالت دیداری و بینایی است که افراد ناتوان را در دسترسی به وب محدود می کند.

۷-۶-۱. سیستم Hearsay

Hearsay دسترسی به اطلاعات اخبار ، تجارت و تحصیل را از طریق صفحات وب در روش ساده و کارآمد فراهم می کند و یک رابط قابل انعطاف و قابل کنترل دارد.

۷-۶-۲. معماری Hearsay

معماری مرورگر صوت Hearsay در شکل ۷-۴ نشان داده شده است. تحلیل‌گر محتوا یک صفحه وب را به یک ساختار منطقی از بخش هایی که شامل مفاد مربوط به عناصر با تحلیل ساختار صفحه است، تقسیم بندی می کند. خروجی تحلیل گر محتوا، یک درخت افراز از محتوی صفحه وب ورودی است.



شکل7-4. معماری HearSay

مدیر رابط هر قسمت را در درخت افراز برچسب گذاری می کند. این برچسب ها توسط تولید کننده دیالوگ که به طور اتوماتیک دیالوگ های VoiceXML را تولید می کنند، به کار برده می شوند. در این سیستم از مفسر VoiceXML استفاده شده است، که تبدیل کننده متن به گفتار و تشخیص دهنده های گفتار برای اجرای دیالوگ های VoiceXML در آن موجودند.

۷-۶-۳. استفاده از Hearsay

خروجی اصلی این سیستم به صورت گفتار است و ورودی را در دو حالت گفتار و متن می پذیرد. برای مثال کاربری که توانایی دیدن ندارد با گفتن "New York Times" وارد این سایت می شود . بعد از بارگذاری این صفحه که در شکل۷-۵ نشان داده شده است، سیستم به صورت اتوماتیک صفحه را به بخش هایی که از نظر محتوا به هم مربوط اند تقسیم می کند و دیالوگ VoiceXML را برای محتوای بخش ها تامین می کند.

شکل7-5. تصویری از سایت New York Times

برای این صفحه سیستم ممکن است که عبارت "There are two sections. Section one keywords are personalize your weather , Updated Friday.Navigate it ?"،New York Times را تولید کند ، اگر کاربر این قسمت را نخواهد با گفتن "No" خواست خود را به سیستم می گوید. سپس سیستم عبارت " Section two keywords are : news , international , business , Navigate it ?" را تولید می کند و کاربر تصمیم می گیرد که در این قسمت به جستجو بپردازد و این را به سیستم اطلاع می‌دهد.

برای تولید دیالوگ ها درخت افراز ساخته شده از محتوای صفحات وب، به عنوان ورودی به جزء تولید کننده دیالوگ Hearsay داده می‌شود. این جزء در طول درخت حرکت می کند و دیالوگی مانند ر،ا برای جستجوی محتوا، با استفاده از گفتار ایجاد می کند. در Hearsay این دیالوگ ها با استفاده از یکسری از نمونه-

های VoiceXML ساخته شده‌اند[۲۹].

۷-۷. پرتال صدای دو زبانه عربی/انگلیسی

این قسمت به بررسی دستاوردی برای ساختن یک پرتال صدای دو زبانه (انگلیسی / عربی) با بهره بری از استانداردهای موجود از جمله VoiceXML می پردازد. این استاندارد محدودیت های خاص خودش را در ورود به زبان عربی دارد. زبان عربی نرمال بدون نشانه ها نوشته می شود و این هم برای تشخیص و هم تبدیل متن به گفتار مشکل ایجاد می کند.

تشخیص جنسیت گوینده نیز برای پاسخ‌های صحیح سیستم مورد نیاز است. سیستم همچنین بر اهمیت تشخیص زبان برای سوئیچ کردن یکپارچه بین هردو زبان تاکید می کند. دو نمونه اولیه با استفاده از این سیستم ساخته شده است، که در این قسمت بررسی می‌شوند.

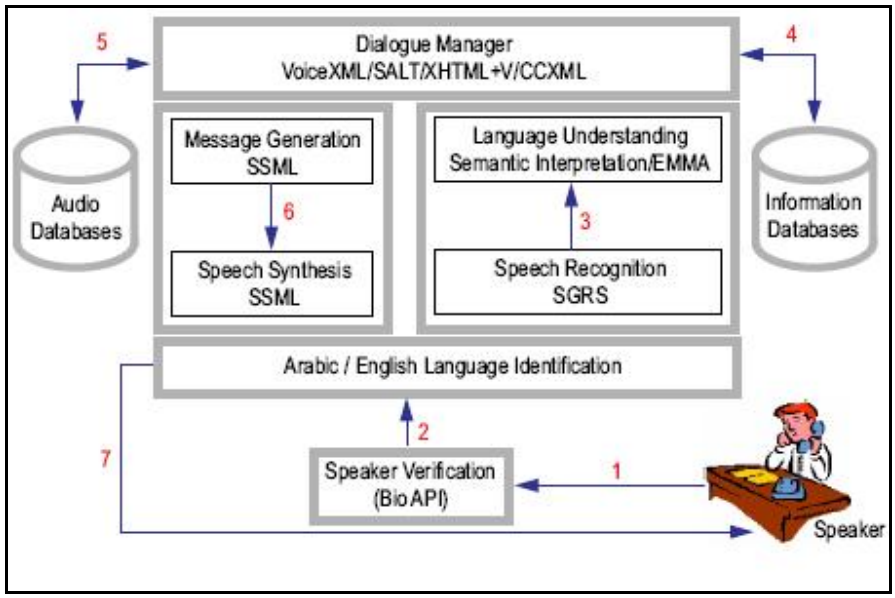
در منطقه عرب ایده دستیابی به اطلاعات اینترنت با استفاده از تکنولوژی تشخیص صدا تنها یک توسعه طبیعی از اینترنت بصری نیست، بلکه همچنین دستیابی به اطلاعاتی را که برای قسمت زیادی از جمعیت غیر موجود است را فراهم می کند. ایده استفاده از شبکه تلفن سوئیچ عمومی (خطوط زمینی) و شبکه های بی سیم (موبایل) برای دستیابی به اطلاعاتی که به صورت خواندنی بر روی وب موجودند به یک مورد مطلوب و کاربردی تبدیل شده است.

پرتال های صدا که با پرتال های وب برابری می کنند دستیابی به اطلاعات وب را از طریق یک رابط صدا اجازه می دهند و یک نقش اصلی و بزرگ را در این منطقه بازی می کنند.

۷-۷-۱. استانداردها و تکنولوژی

در پرتال صدای تک زبانه اجزای تکنولوژی وجود دارند که یک ارتباط بین کاربر و سیستم را میسر می‌سازد. این اجزا عبارتند از: مدیر دیالوگ ها، ماژول تشخیص گفتار، ماژول فهم یک زبان، ماژول تبدیل متن به گفتار که

شامل تولید پیغام و تبدیل متن به گفتار می شود (شکل۷-۶).



شکل7-6. پرتال دو زبانه

وظیفه مدیر دیالوگ دیکته کردن دنباله فرمان ها و جواب ها یا حالت های دیالوگ است، با اطلاعات و پایگاه داده صدا ارتباط برقرار می کند و تماس های تلفنی را مدیریت می کند. برای مثال انتقال تماس در حالت یک سیستم پرس و جوی کتاب راهنما. تشخیص گوینده یک ماژول اضافی است که می تواند به سیستم اضافه شود .

در یک پرتال صدای دو زبانه ، ماژول اضافی دیگر تشخیص زبان می تواند به سیستم اضافه شود که مسئول سوئیچ کردن اتوماتیک بین دو زبان عربی و انگلیسی (یا هر زبان دیگری مثل فرانسه در لبنان و کشورهای شمال آفریقا) است.

با هدف تبدیل شدن پرتال های صدا به یک تکنولوژی اصلی، یک استاندارد open مثل استاندارد HTML در وب، برای تکنولوژی تشخیص صدا مورد نیاز است. تعدادی از این استانداردها در حال حاضر موجود است و تعدادی هم در دست ساخت است. همان طور که در شکل۷-۶ قابل مشاهده است، هر استاندارد می تواند به یک ماژول متصل شود. تنها ماژولی که استاندارد مخصوص به خودش را ندارد ماژول تشخیص زبان است. چون اکثر

استانداردها مسلط به یک زبان و اساسا برای انگلیسی فراهم می آورند.

به هر حال تعدادی از استانداردها از جمله VoiceXML ساختار درونی چند زبانی دارند اما از آنجایی که عربی متعلق به کلاس زبان هایی است که با الفبای لاتین قابل نمایش نیست ، پیاده سازی پرتال های صدای عربی احتیاجات خاصی را لازم دارد که باید تامین شود.

VoiceXML اساسا برای پذیرفتن دیالوگ های صدا که شامل تشخیص صدا، تبدیل متن به گفتار و ضبط و پخش صدا می باشد، DTMF یا ورودی لمسی و تلفن طراحی شده است. VoiceXML همچنین از مکالمات آمیخته اولیه زمانی که تماس گیرنده و سیستم در به وجود آوردن یک دیالوگ نوبت می گیرند پشتیبانی می-کند.

دیگر استانداردهایی که بسیار نزدیک به VoiceXML هستند، استانداردهایی هستند که توسط W3C تحت کالبد رابط گفتار توسعه پیدا کرده اند مانند SRGS ، SSML ، EMMA (زبان علامت دار تفسیر چندوجهی قابل توسعه) یا SI (تفسیر معنایی).

Call Control XML (CCXML) : XML کنترل تماس، زبان علامت دار دیگری است که کنترل پردازش سیگنال گفتار و محیط تلفنی را میسر می کند و به VoiceXML لینک می شود. نمایش تماس، جواب و پاسخگویی به تماس، انتظار برای تماس و انتقال تماس مثال هایی از کارکرد مدیریت تماس هستند.

آخرین جز مهم سیستم پرتال تشخیص صدا که در شکل نشان داده شده است تشخیص و شناسایی گوینده است (SV [1]) . تشخیص گوینده یک جز امنیتی را به سیستم نهایی اضافه می کند .

بر طبق آنچه که در بالا گفته شد به نظر می رسد که VoiceXML به حدی که برای سیستم های دو زبانه لازم است دسترسی پیدا کرده است. اگر چه اختیار یک استاندارد مثل VoiceXML محدودیت های خودش را دارد که عبارتند از:

[1] Speaker Verification

- وضعیت‌های دیالوگ به دیالوگ های مستقیم و آمیخته آغازی محدود می شوند.
- تفسیر معنایی به جفت مقدار/کلید در گرامر محدود می شود.
- یک پشتیبانی محدودی برای چند زبانی بودن وجود دارد
- مشخصا جایی برای تشخیص زبان نیست.

۲-۷-۷. دوزبانی بودن و VoiceXML

VoiceXML یک پشتیبانی محدودی برای چند زبانه بودن بیان کرده که اساسا در صفت xml:lang برای فرمان ها و گرامرها مشخص می شود. برای مثال سیستم می تواند در عربی و انگلیسی فرمان دهد:

```
<prompt xml:lang=”en-US”>
                Hello</prompt>
<prompt xml:lang=”ar-UAE”>
                ب ح ر م</prompt>
```

به طور مشابه در تشخیص دو کلمه بالا در عربی و انگلیسی تگی که برای مشخص کردن گرامر مورد نیاز است به صورت زیر است:

```
<grammar xml:lang=”en-US”>
                Hello</grammar>
<grammar xml:lang=”ar-UAE”>
                ب ح ر م</grammar>
```

عربی تعدادی از مشکلاتی را که به عنوان بخشی از استاندارد در نظر گرفته نشده است، مطرح می کند. مهمترین آن‌ها املای صحیح عربی (متن نوشته) و مشخص کردن و جدا کردن آن‌ها و تشخیص جنسیت (مذکر یا مونث بودن) است. به‌علاوه هیچ قانون یا ماده ای برای سو ئیچ کردن اتوماتیک بین عربی و انگلیسی وجود ندارد، برای مثال یک ساختاری که تشخیص زبان را پشتیبانی کند.

۳-۷-۷. عربی کردن VoiceXML

عربیِ نوشتاری بر پایه عربی کلاسیک است. عربی گفتاری مشکلات زیادی را نسبت به عربی استاندارد مدرن MSA دارد. یکی از دلایل آن تنوعات زبانی است که بر اساس محل های جغرافیایی مختلف، وضعیت اجتماعی و همچنین تاثیر پذیری از فرهنگ یا زبان های خارجی مثل کلمات انگلیسی به لهجه محلی وارد شده-اند.

یک ویژگی تمیز دادن عربی از زبان های دیگر این است که به صورت نرمال بدون حرکت یا تفکیک کننده و نشان تشخیص نوشته می شود. این هم بر تشخیص و هم بر تبدیل متن به گفتار دلالت دارد. برای تشخیص گفتار سه مشکل اساسی وجود دارد:

- ابهام معنایی: از آن جا حاصل می‌شود که معنی یک کلمه می تواند بر اساس نوع حرکت هایی که تولید می شود، عوض شود.
- وجود نداشتن حرکت واکه می تواند نشان دهد که مدل های صوتی درست آموزش ندیده اند.
- مدل های زبان آماری اگر در یک متن بدون تفکیک کننده و نشان آموزش دیده باشند ممکن است که درست قابل پیشگویی نباشند.

اگر یک متن قبل از فرستادن به موتور تولید گفتار تفکیک و نشانه گذاری نشود، تلفظ کلمه و عبارت بر روی تبدیل متن به گفتار به صورت ناسازگار تاثیر می گذارند. یک دستاورد آشکار برای مقابله کردن با این موضوع استفاده از یک جدا کننده و نشان‌گذار تجاری به عنوان یک مرحله پیش‌پردازش برای تشخیص و تبدیل متن به گفتار است.

یک دستاورد امید بخش برای املای صحیح عربی لاتین کردن است، که می تواند با نشانه گذاری اتوماتیک ترکیب شود و به دستاوردی که می تواند به عنوان لاتین سازی خودکار نام‌گذاری شود، تبدیل گردد.

بزرگترین مزیت استفاده از لاتین سازی این است که بدون نگرانی از اینکه از کدام کدگذاری استفاده کنیم، انگلیسی و عربی می توانند در یک متن VoiceXML مشابه ترکیب شوند.

نه تشخیص جنسیت و نه لاتین سازی خودکار حداقل به صورت مستقیم توسط VoiceXML پشتیبانی نمی شوند. اما می توانند با بهره برداری از شی تگ یا با شامل شدن آن ها به عنوان بخشی از ماژول‌های تشخیص و تولید گفتار، پیاده سازی شوند. این موضوع همچنین برای تشخیص زبان صحت دارد.

۷-۷-۴. توسعه دادن نمونه های اولیه

دو نمونه اولیه برای نشان دادن ایده های عمومی بیان شده، ساخته شده است. این دو نمونه عبارتند از:

- نمونه[1] پرداخت جریمه پلیس دبی
- یک برنامه کاربردی برای بچه ها

۷-۷-۴-۱. نمونه پرداخت جریمه

یکی از تصمیمات اولیه این پروژه ساختن یک پرتال صدا برای سیستم پرداخت جریمه ترافیک بود. جریمه-های ترافیک پلیس دبی ضبط می شوند و سپس به اینترنت پست می شوند و به این صورت اتوماتیک می شوند. گوینده با استفاده مواردی مثل : شماره شناسایی، شماره گواهینامه و شماره پلاک ماشین می تواند از جریمه خود پرس و جو کند.

برای پیاده سازی انگلیسی_عربی دو زبانه کارهای زیر باید انجام شود:

هرگاه برنامه به زبان مختلف سوئیچ می شود ساختار "xml:lang = "language باید استفاده شود (هر جا که زبان en_US یا ar_UAE است). این هم برای تگ های فرمان و هم برای گرامر باید انجام شود.

تشخیص عربی با استفاده از متن عربی در گرامرها که از یک محل ar_UAE استفاده می کند و کامپایل می شود، پیاده سازی می شود.

[1] Demo

سرور وب صدا هیچ مقدار TTS به زبان عربی ندارد که به محیط VoiceXML لینک شود بنابراین صدا برای زبان عربی باید از قبل ضبط شود.

پیاده‌سازی تشخیص زبان و جنسیت به صورت offline است. هر دو ماژول در محیط VoiceXML به عنوان Object وجود دارد. همچنین تلاش‌هایی در مجتمع کردن موتور متن به گفتار، که با نشانه گذاری و نرمال کردن جلو به عقب متن ترکیب شود، وجود دارد.

۲-۴-۷-۷. برنامه ای برای بچه ها

Modhesh یک شخصیت و یک کاراکتر برای بچه ها است که پیغام های اجتماعی می دهد.

با این برنامه به مواردی مثل حقایق جالب ، ضرب المثل ها ، داستان ها و اخبار اخیر می توان گوش داد. در شکل۷-۷ مثالی از کد VoiceXML برای ارتباط با منو نشان داده شده است[۳].

```
<?xml version="2.0" encoding="windows-1256"?>
<vxml version="2.0">

   <!-- Example Menu -->
     <menu id="menu1">

        <prompt xml:lang="ar-UAE"> _________ ____ __ ________ ________ </prompt>
        <prompt xml:lang="ar-UAE">  _____ __________ ___________________ _________ _____
</prompt>
        <prompt xml:lang="ar-UAE"> ________ ___ </prompt>
        <prompt xml:lang="ar-UAE"> ________ ___ </prompt>
        <prompt xml:lang="ar-UAE"> _________ ____ </prompt>
        <prompt xml:lang="ar-UAE"> ________ </prompt>

        <!-- Choices are expressed using Buckwalter transliteration  -->
        <choice next="#Haqa'eq"> Haqa'q  </choice>
        <choice next="#Tara'ef"> Tara'ef </choice>
        <choice next="#Hekayat"> Hekayat </choice>
        <choice next="#'axbaar"> 'axbaar </choice>

     </menu>

     <!-- Example Form -->
     <form id="2a5baar">
        <block>
              <prompt xml:lang="ar-UAE"> _____ _________ ____ _____ ________ _______
</prompt>
        </block>
     </form>

</vxml>
```

شکل7-7. نمونه از کد VXML

۷-۸. مرورگر وب دو زبانه انگلیسی/ چینی

در این قسمت یک مرورگر بر پایه صدای دو زبانه چینی/انگلیسی که با بستر VXML پیاده سازی شده بررسی می‌شود. دو سیستم کاربردی یکی برای اعلام خبر CU News و دیگری برای اعلام وضعیت هوا CU Weather توسعه داده شده است که در ادامه توضیح داده می شود.

۷-۸-۱. بستر مورد استفاده

مرورگر CU Voice Browser بر اساس OpenVXI 2.0 توسعه داده شده است. امروزه تلاش ها در این جهت است که بستری که توسعه داده می شود دسترس پذیری جهانی داشته باشد. XSL زبانی است برای تبدیل زبان های مختلف نمایش اطلاعات به یکدیگر. یک مرورگر صوتی وب یک برنامه است که بر روی Voice Gateway اجرا می‌شود و می تواند اسناد VoiceXML را ترجمه کند.

۷-۸-۲. برنامه CU Weather

این نرم افزار یک نرم افزار دو زبانه است که اطلاعات هواشناسی در آن به صورت بلادرنگ از وب سایت هواشناسی هونگ کونگ[1] به روز می شود. یک نمونه از دیالوگ انسان_ کامپیوتر در جدول زیر آمده است.

C: Welcome to CU Weather. Today is 22/11/2003.
Please select Hong Kong weather or world weather.
H: World weather
C: Please say the city name.
H: Shanghai
C: The current weather conditions for Shanghai is…

سند VXML مربوطه در ادامه آمده است.

```
<?xml version="1.0"?>
```

[1] www.hjo.gov.hk

```
<vxml version= “1.0”>
<link next="#exit">
<!--This is a link at the VXML level and so the grammar
remains active throughout the document. The user can exit
the dialog by saying goodbye of byebye -->
<grammar>goodbye | byebye </grammar>
</link>

<form id=”welcome” domain =”document”>
<block>
<!-- The [date] information below is automatically filled in
as XSL transformation takes place to generate the VXML
document from XML content -->
<prompt> Welcome to CU Weather. Today is [date]
</prompt>
<goto next="#type" />
</block>
</form>

<menu id="type" domain="document">
<prompt> Please select Hong Kong weather or world
weather </prompt>
<choice next="#local">
<!--Jumps down to the form with id local below -->
<grammar>hong kong weather | hong kong<grammar>
</choice>
<choice next="#world">
<!--Jumps down to the form with id world -->
<grammar>world weather | world<grammar>
</choice>
</menu>
```

<form id="local" دامنه="document">

<!-- form with id *local* -->

<block>

<prompt> Hong Kong's current weather condition is…

</prompt>

<goto next="#type" />

</block>

<form>

<form id="world" domain ="document">

<!-- form with id *world* -->

<field namelist="other_city">

<grammar src="eng_city.gram"/>

<!--the list of city names is obtained from the grammar file eng_city.gram for filling in the variable other_city. -->

<prompt> Please say the city name </prompt>

<!-- Based on the filled variable، the system retrieves the relevant weather information. -->

…..

</form>

<form id="exit">

<block>

<prompt> Thank you for calling.</prompt>

<exit expr="success" />

</block>

</form>

</vxml>

۳-۸-۷. برنامه CU News

این برنامه نیز مانند مثال قبل یک نرم افزار دو زبانه است و بر خلاف نمونه قبلی که با دامنه لغت محدود (حدود ۶۰ اسم شهر به دو زبان) بود، دامنه لغات پویا دارد که بر اساس تیتر اخبار می باشد. کاربر می تواند با استفاده از این سیستم از اخبار مطلع شود.

در فصل بعد، مروری به طراحی و ساختار برخی پایگاه دادگان گفتاری و کاربرد آن‌ها در تولید گفتار خواهیم داشت.

فصل 8
پایگاه دادگان گفتاری

۸-۱. مقدمه

استفاده از گفتار جهت برقراری ارتباط با ماشین‌ها در سال‌های اخیر مورد توجه پژوهشگران قرار گرفته است. امروزه با توجه به افزایش قدرت پردازشی پردازنده‌ها و حجم حافظه‌ها تمایل افراد به سمت ذخیره‌ی اطلاعات گفتاری در رایانه‌ها بیشتر شده و این سبب شده است تا جستجو، تشخیص و پردازش‌های لازم مستقیما روی فایل‌های گفتاری ذخیره شده در رایانه اعمال شود[۳۶].

توسعه‌ی نرم‌افزارهای لازم برای پردازش سیگنال‌های گفتار در کاربردهای متعدد از جمله تشخیص و سنتز نیازمند دادگان‌های گفتاری می‌باشد. در تشخیص گفتار، دادگان‌ها می‌توانند به منظور آموزش مدل‌های آکوستیکی و زبانی بکار روند، به علاوه به منظور مقاوم‌سازی تشخیص گفتار، استفاده از دادگان‌های زبانی ضرورت دارد. در سنتز گفتار به منظور انتخاب واحدهای ساختاری گفتار مانند، واجها، سیلابها، کلمات، عبارات، جملات و یا گذارها از یک واج به واج دیگر متناسب با نوع روش به کار رفته برای سنتز گفتار، دادگان‌های بزرگ یا کوچک مورد استفاده قرار می‌گیرند.

در تهیه‌ی دادگان‌های صوتی قواعد خاصی مورد نظر قرار می‌گیرد؛ از جمله پایگاه داده باید شامل کلمات و عبارات مناسبی باشد که معرف مجموعه گفتاری زبان مورد نظر باشد و از طرفی گویندگان باید از جنسیت، سن و سطح سوادهای مختلف جهت پوشش دادن مشخصه‌های گفتاری مختلف، انتخاب شوند.

معیارهایی که معمولا در ایجاد یک پایگاه داده‌ی گفتاری موثرند عبارتند از:

- داده‌های موجود در پایگاه داده که شامل گفتارهای معمولی (Spontaneous) و گفتارهای خواندنی (Read) می‌باشند.
- فرکانس نمونه‌برداری، که برای پایگاه داده‌های تلفنی معمولا ۸، ۱۱ یا ۱۶ کیلو هرتز می‌باشد.
- استفاده از افراد با گویش‌های محلی(در صورتی که یک طرح ملی در دست اجرا باشد)
- گفتار گسسته یا پیوسته در دادگان گفتاری با توجه به کاربرد مورد نظر

پایگاه داده‌های گفتاری معروفی که در این زمینه ایجاد شده‌اند، شامل موارد زیر می‌باشند.

۸-۲. پایگاه داده‌ی گفتاری TIMIT

پایگاه داده‌ی TIMIT توسط شرکت TI[1] و دانشگاه MIT[2] تهیه شده و اداره‌ی استاندارد آمریکا NIST[3] آنرا تایید کرده است. TIMIT یک بانک اطلاعاتی فونتیک/آکوستیکی از گفتار پیوسته‌ی انگلیسی می‌باشد که در بسیاری از مقالات مربوط به شناسایی گفتار از آن استفاده می‌شود، کلیه کلمات و واج‌های موجود در جملات این بانک اطلاعاتی دارای برچسب زمانی هستند[۳۷، ۳۸] .

پایگاه داده‌ی TIMIT عاری از نویز است و معمولا برای ارزیابی نرخ بازشناسی واج‌ها در بازشناسی گفتار پیوسته مورد استفاده قرار می گیرد. برای کاربرد این دادگان در ارزیابی روشهای مقاوم سازی بازشناسی گفتار، باید نویز را بطور مصنوعی به این دادگان اضافه نمود[۳۹، ۴۰] .

۸-۳. پایگاه داده‌ی گفتاری Tfarsdat

اولین پایگاه داده‌ی گفتاری آکوستیک/فونتیک فارسی است که هدف آن ایجاد یک سیستم پایه-تلفنی برای تشخیص خودکار گفتار می‌باشد.

از آنجایی که یکی از کاربردهای مهم این پایگاه داده، در تشخیص گفتار می‌باشد و در تشخیص گفتار سعی می‌شود مستقل از گوینده عمل شود، از افراد مختلف با سن و جنسیت‌های متفاوت استفاده شده است[۴۱].

[1] Texas Instruments
[2] Massachusetts Institute of Technology
[3] National Institute for Standards and Technologies

۸-۴. پیکرهٔ متنی زبان فارسی

پیکره متنی زبان فارسی عبارتست از حجم انبوهی از متون الکترونیکی فارسی که برحسب متغیرهای نوع و موضوع از منابع معتبر موجود و به همراه تجزیه و تحلیل زبانی داده ها و طراحی مناسب برای استفاده، از کاربران تهیه شده است. پیکره متنی، تنوع داده های زبانی و موضوعات مختلف ممکن را شامل می شود. پیکرهٔ متنی نمایندهٔ کاملی از جامعه آماری متون فارسی بوده و از نظر مرز کلمات یکدست و یکنواخت می باشد[۴۲]. پیکره-های متنی برای تخمین پارامترهای مدلهای زبانی مورد استفاده قرار می‌گیرند.

۸-۵. پایگاه داده‌ی گفتاری TIDIGITS

پایگاه داده‌ی TIDIGITS یک بانک اطلاعاتی گفتار انگلیسی است که در شرکت TI تهیه شده است. هدف از تهیه این دادگان طراحی و ارزیابی الگوریتمهای بازشناسی مستقل از گوینده برای دنباله متصل اعداد[۱] می‌باشد[۹].

۸-۶. پایگاه داده‌ی گفتاری AURORA 2

پایگاه داده‌ی گفتاری AURORA 2 برای ارزیابی دقت بازشناسی سیستم‌های بازشناسی گفتار در محیطهای نویزی و به ویژه برای ارزیابی روشهای جبران ویژگی در شرایط نویزی متفاوت مورد استفاده قرار می-گیرد. این دادگان با استفاده از گفتار بزرگسالان دادگان گفتاری TIDIGIT ساخته شده است. به این ترتیب که ابتدا نرخ نمونه برداری به ۸ کیلو هرتز کاهش یافته است و سپس فیلتری برای شبیه سازی تاثیر کانال انتقال تلفنی بر روی پایگاه داده اعمال گردیده است. برای شبیه‌سازی اثر خط تلفن، دو نوع فیلتر استاندارد مورد

[۱] Connected Digit Sequences

استفاده قرار گرفته‌اند که با نامهای G.712 و MIRS [۴۳، ۴۴] شناخته می شوند.

۸-۷. دادگان مورد استفاده در سنتز گفتار

دو تکنولوژی اولیه و اصلی برای تولید و سنتز گفتار عبارتند از سنتز اتصالی[۱] و سنتز فرمانت. هر کدام از این دو تکنولوژی دارای نقاط ضعف و قدرت خاص خود هستند. بر اساس کاربرد مورد نظر یکی از این دو رویکرد مورد استفاده قرار می‌گیرند. در این بخش به دادگان مورد استفاده برای سنتز اتصالی پرداخته شده است.

۸-۷-۱. سنتز اتصالی

رویکرد سنتز اتصالی بر اساس الحاق بخش‌هایی از گفتار ضبط شده به یکدیگر می‌باشد. سه زیر بخش مهم از این رویکرد عبارتند از:

- سنتز با امکان انتخاب واحد[۲]
- سنتز Diphone
- سنتز با دامنهٔ تعریف مشخص[۳]

۸-۷-۱-۱. سنتز با امکان انتخاب واحد

این رویکرد از سنتز اتصالی از دادگان بزرگ (در حد گیگا بایت) گفتار ضبط شده استفاده می‌نماید. در طول آماده‌سازی دادگان، هر عبارت ضبط شده به بخش‌های منفرد واج، diphone، نیم واج، سیلاب، morpheme، کلمه، عبارت و جمله تقسیم می‌شود. سپس شاخص هر واحد در دادگان گفتار بر اساس پارامترهای

[۱] Concatenative

[۲] Unit selection synthesis

[۳] Specific domain synthesis

آکوستیکی نظیر فرکانس فرمانت (pitch)، موقعیت در زیر بخش، واج‌های همسایه و طول زمانی آن واحد ایجاد می‌شود. در زمان اجرا، عبارت مورد نظر با تشخیص بهترین دنباله از واحدهای انتخاب شده از دادگان گفتار، ایجاد می‌شود.

۸-۷-۱-۲. سنتز Diphone

سنتز diphone از یک پایگاه دادهٔ کمینه استفاده می‌کند که شامل تمام گذارهای واج به واج (diphone) می‌باشد که در یک زبان رخ داده‌اند. تعداد این گذارهای واج به واج به زبان مورد نظر وابسته است. مثلا در زبان اسپانیولی تعداد این diphoneها، حدود ۸۰۰ عدد می‌باشد و در زبان آلمانی ۲۵۰۰ diphone وجود دارد.

در سنتز diphone، تنها یک نمونه از هر diphone در دادگان گفتار نگهداری می‌شود. در زمان اجرا prosody مورد انتظار یک جمله، با ترکیب این واحدهای کمینه و با به‌کارگیری تکنیک‌های پردازش سیگنال رقمی حاصل می شود. کیفیت گفتار حاصل شده در این رویکرد، ضعیف تر از کیفیت گفتار حاصل شده در رویکرد انتخاب واحد می‌باشد.

۸-۷-۱-۳. سنتز در دامنهٔ تعریف مشخص

در این رویکرد کلمات و عبارات از پیش ضبط شده برای تولید عبارت نهایی مورد استفاده قرار می‌گیرند. بنابراین، این رویکرد در کاربردهای خاصی مانند گزارشات هواشناسی یا اطلاعات برنامهٔ مسافرت‌ها مورد استفاده قرار می‌گیرد.

از آنجا که در دادگان گفتار مربوط به این رویکرد تنها کلمات و عبارات محدودی ذخیره می‌شوند، این دادگان همه منظوره نبوده و تنها برای کاربردهای از پیش برنامه‌ریزی شده مورد استفاده قرار می‌گیرد.

جمع‌بندی

امروزه شبکه جهانی وب در زمینه های مختلف تجارت، آموزش، اطلاع رسانی و زمینه های متنوع دیگر کاربرد دارد . از آنجایی که تمامی افراد در تمامی نقاط دنیا دارای دسترسی آسان به شبکه جهانی وب نیستند، استفاده از صوت و شبکه تلفن عمومی در دسترسی به محتوای اینترنت یک امر قابل قبول است .

همچنین استفاده از صوت در ارتباط با کامپیوتر برای افرادی که توانایی بینایی ندارند کارایی قابل توجهی دارد.

سیستم های دیالوگ یک دستاورد مناسب برای این سرویس ها هستند، که قابلیت استفاده از وب را نیز افزایش می دهند. امروزه انواع مختلفی از مرورگرهای صوتی و سیستم های دیالوگ با پشتیبانی از زبان های مختلف برای کاربران با زبان های متفاوت به وجود آمده اند و تشخیص دهنده گفتار و سیستم تبدیل متن به گفتار آن‌ها نیز از این استانداردها پشتیبانی می‌کنند. استفاده از صوت دارای مزایای است که چند مورد آن از این قرار است :

- استفاده از سیستم ها را ساده و آسان می سازد .
- انواع مختلف ارتباط از جمله منوها، پر کردن فرم ها و فرمان ها را پشتیبانی می کند.
- قابلیت ترکیب با حالت های دیگر ورودی غیر صوت را دارد.
- برای انواع مختلف کاربران از زبان های مختلف، مهارت ها و سن های مختلف قابلیت تطبیق دارد.

به منظور استانداردسازی این گونه سیستم ها کنسرسیوم وب جهانی، استانداردهایی را برای تشخیص گفتار ، تولید گفتار از متن و تفسیر معنایی منتشر کرده است که در فصول قبل به تعدادی از آن‌ها اشاره شد.

استاندارد VXML برای تولید و ایجاد دیالوگ ها در این سیستم ها استفاده می شود و کنترل جریان دیالوگ ها را بر عهده دارد. VXML برای ایجاد دیالوگ های صوتی که بر اساس گفتار ترکیبی، صدای دیجیتالی،

تشخیص صدا، ورودی کلید با DTFM[1]، ضبط صدای ورودی و تلفنی می باشند، طراحی شده است و از گرامر تشخیص گفتار SRGS و استاندارد تبدیل متن به گفتار در اسناد VXML استفاده می شود.

گرامر های تشخیص گفتار به دو بخش تقسیم می شوند: بخش نحوی که دنباله‌ای از کلمات را که گوینده می‌تواند برای تشخیص تلفظ کند را محدود می کند، و بخش معنایی که به تولید نتایج از تشخیص گفتار کمک می‌کند. تفسیر معنایی قسمت دوم یک گرامر نامیده می شود. گرامر تشخیص گفتار استاندارد ارائه شده برای بخش اول است که کلمات و الگو های کلمات را که تلفظ می شوند بیان می کند .

زبان علامت گذاری تبدیل متن به گفتار، یک روش استاندارد جدید برای تولید مواردیست که باید توسط سیستم تبدیل متن به گفتار بیان شود. SSML یک زبان علامت گذاری برای تولید فرمان ها با استفاده از ترکیب موارد ضبط شده تبدیل متن به گفتار است. کنترل کردن خصوصیات صدای تولید گفتار شده مثل جنسیت مخاطب (مرد یا زن بودن)، سن، سرعت، شدت، پیچ و تاکید امکان پذیر می باشد.

استفاده از این استانداردها باعث کاهش هزینه توسعه سیستم های دیالوگ، آسان نمودن تکنولوژی های مجتمع سازی، به حداقل رساندن فعل و انفعالات بین سرویس گیرنده و سرویس دهنده، دور کردن برنامه نویس از جزئیات سخت افزاری و سطح پائین و افزایش دادن قابلیت انتقال سیستم می شود. این امر باعث شده توسعه دهنده‌ها به سمت استفاده از این استانداردها در سیستم های خود پیش روند.

[1] Dual-Tone Multi-Frequency

منابع و مراجع

[1]. Lemmetty Sami, “Review of Speech Synthesis Technology”, *Thesis of Helsinki University of Technology, Department of Electrical and Communications Engineering*,1999.

[2]. Apple Speech Technologies Home Page (1998). <http://www.apple.com/macos/speech/>.

[3]. Habib Talhami, “The Bilingual Voice Portal in the Arab Region: Voice Browsing in Arabic, English, or Mixed Language “, *The European Journal for the Informatics Professional*, UPGRADE Vol. V, No. 3, June 2004,

[4]. Barber S., Carlson R., Cosi P., Di Benedetto M., Granström B., Vagges K. (1989). A Rule Based Italian Text-to-Speech System. *Proceedings of Eurospeech 89*, pp. 517-520.

[5]. Bell Laboratories TTS Homepage (1998). <http://www.bell-labs.com/project/tts/>.

[6]. Benoit C. (1995). Speech Synthesis: Present and Future. *European Studies in Phonetic& Speech Communication*. Netherlands. pp. 119-123.

[7]. Bernstein J., Pisoni D. (1980). Unlimited Text-to-Speech System: Description and Evaluation of a Microprocessor Based Device. *Proceedings of ICASSP 80*, pp. 574-579.

[8]. H. Meng, Y.Chi Li, T. Fung et al “Bilingual Chinese/English Voice Browsing based on a VoiceXML Platform “, *ICASSP* SS-1.7, 2004.

[9]. Tobias Knoben, “ Voice Browsing “, IKP, Rheinische Friedrich- Wilhelms Universitat Bonn, Germany, European *Masters in Language and Speech*, 2005

[10]. Doug B. Brown, “Speech Recognition Redefines Self-Service“, eloyalty, *OPTIMIZING CUSTOMER INTERACTIONS*, 2003

[11]. Michael K Brown, Stephen C. Glinski, Brian C. Schmult, "Web Page Analysis for Voice Browsing", *Proceedings of the, First International Workshop on, Web Document Analysis*, WDA2001.

[12]. Ramón López-Cózar, Zoraida Callejas, Germán Montoro," DS-UCAT: A New Multimodal Dialogue System for An Academic Application ", *Interspeech 2006 - ICSLP Satellite Workshop*, 2006.

[13]. Hakan Melin, "ATLAS : A generic software platform for speech technology based applications ", Center *for Speech Technology (CTT)*, TMH, KTH, Drotting Kristians vag 31, SE- 100-44 Stockholm, TMH-QSPR 1/2001

[14]. Voice Extensible Markup Language (VoiceXML) Version 2.0, http://www.w3.org/TR/2004/REC_VoiceXML 20_20040316.

[15]. Marta Gatius , Meritxell Gonzalez, Shelya Militello, "Integrating Semantic web Language Technologies to improve the online public Administration Service ", *Copyright is held by the international World Wide Web Conference Committee (IW3C2)*, ACM 1-5959-323-9/06/0005, 2006.

[16]. N.Petkovski, M.Gusev,"VoiceXML", *Institute of Informatics, Faculty of Natural Sciences and Mathematics, Ss.Cyril and Methodius University* , 2003.

[17]. Heather Du, "Vocal Information Retrieval: A System Prototype and User Interface Design Issues", Department *of Computer and Information Sciences University of Strathclyde*, 2002.

[18]. Jim A. Larson, "VOICEXML 2.0 AND THE W3C SPEECH INTERFACE FRAMEWORK", *Intel Corporation*.

[19]. Kundan Singh, Ajay Nambi and Henning Schulzrinne, "Integrating VoiceXML with SIP services" , *Communications. ICC'03, IEEE International Conference*, 2003.

[20]. Paolo Baggia, "Impact of standards on Automatic Speech Recognition ", *Loquendo*, 2005.

[21]. "Voice Browser Activity" : http://www.w3.org/Voice.

[22]. SSML 1.0: an XML-based languaged aimed to improve TTS rendering", http://www.SSML 1_0 an XML-based languaged aimed to improve TTS rendering.htm

[23]. "EMMA: Extensible MultiModal Annotation markup language", W3C Working Draft 16 September 2005, http://www.w3.org/TR/2005/WD-emma-20050916/.

[24]. Ramón López-Cózar, Zoraida Callejas, Miguel Gea, Germán Montoro, "Multimodal, Multilingual and Adaptive Dialogue System for Ubiquitous Interaction in an Educational Space", *Proceedings of ISCA Workshop on Applied Spoken Language Interaction in Distributed Environments*, ITRW, 2005.

[25]. Stefan W. Hamerich, Volker Schubert, Volker Schless, "Semi-Automatic Generation of Dialogue Applications in the GEMINI Project", *SIGdial*, 2004

[26]. R. L´opez-C´ozar, A. de la Torre, J.C. Segura, A.J. Rubio, V. S´anchez, "Testing Dialogue Systems By Means of Automatic Generation of Conversations," *Interacting with Computers*, vol. 14, no. 5, pp. 521–546, 2002.

[27]. Akinori Ito, Keisuke Shimada, Motoyuki Suzuki, Shozo Makino, "A User Simulator based on VoiceXML for evaluation of spoken dialog systems ", *INTERSPEECH* 2006.

[28]. Jozef Juhar1, Stanislav Ondas1, Anton Cizmar1, "Development of Slovak GALAXY/VoiceXML Based Spoken Language Dialogue System to Retrieve Information from the Internet", *INTERSPEECH* 2006.

[29]. "Dialog Generation for Voice Browsing ", Zan Sun, Amanda Stent, I.V. Ramakrishnan, *Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility*, 2006.

[30]. 27. Shanmugham,S, "A Media Resource Control Protocol Developed by Cisco, Nuance, and Speechworks", draft-shanmugham-mrcp-07 (work in progress), April 2005

[31]. IBM Press room, "IBM releases First Java Speech API Implementation" article, http://www-03.ibm.com/press/us/en/pressrelease/2413.wss

[32]. Cloud Garden, TalkingJava SDK with Java Speech API implementation, http://www.cloudgarden.com/JSAPI/

[33]. Festival project online resources, The Festival Speech Synthesis System, http://www.cstr.ed.ac.uk/projects/festival/

[34]. Java Speech API Markup Language Specification, http://java.sun.com/products/java-media/speech/forDevelopers/JSML/Specification.html#18867

[35]. Java Speech API, Code samples document, http://java.sun.com/products/java-media/speech/reference/codesamples/index.html

[36]. Saeedeh Momtazi, Hossein Sameti, Saman Vaisipour and Meysam Tefagh, "Introducing a Framework to Create Telephony Speech Databases from Direct Ones", *Systems, Signals and Image Processing and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services*, pp. 327-330, 2007.

[37]. Kai-Fu Lee and Hsiao-Wuen Hon, "Speaker-Independent Phone Recognition Using Hidden Markov Model", *IEEE TRANSACTIONS ON ACOUSTICS. SPEECH, AND SIGNAL PROCESSING*, Vol. 37. No. 11. 1989.

[38]. Santiago Fernández, Alex Graves and Jürgen Schmidhuber, "Phoneme recognition in TIMIT with BLSTM-CTC", *Technical Report* No. IDSIA-04-08, 2008.

[39]. L.F. Lamel and J.L. Gauvain, "High Performance Speaker-Independent Phone Recognition Using CDHMM", *Speech Communication Group*, pp. 121-124, 1993.

[40]. Fatouma Boukadida and Noureddine Ellouze, "Arabic Intonatif Speech Database", *IEEE International Conference on Industrial Technology, ICIT*, pp. 1408-1411, 2004.

[41]. Mahmood Bijankhan, Javad Sheykhzadegan, Mahmood R. Roohani, Rahman Zarrintare, Seyyed Z. Ghasemi and Mohammad E. Ghasedi, "Tfarsdat - the Telephone Farsi Speech Database", *EUROSPEECH*, pp. 1525-1528, 2003.

[۴۲]. پیکرهٔ متنی زبان فارسی, پژوهشکدهٔ پردازش هوشمند علائم.

[43]. David Pearce and Hans-Günter Hirsch, “The Aurora Experimental Framework for The Performance Evaluation of Speech Recognition Systems Under Noisy Conditions”, *ICSLP 2000 (6th International Conference on Spoken Language Processing)*, 2000.

[۴۴]. بابک ناصر شریف, "مقاوم سازی بازشناسایی گفتار در محیط با نویز ناهمبسته با استفاده از پردازش چند باندی", پایان نامه جهت دریافت درجه‌ی دکتری در رشته‌ی هوش مصنوعی, *دانشکده کامپیوتر، دانشگاه علم و صنعت ایران*، اردیبهشت ۸۶.