Adina Svensson
Handledare: Mats Eeg-Olofsson & Johan Frid
Lunds universitet
Institutionen för lingvistik

Talande webbläsare

c-uppsats i datalingvistik

vt 2001

Innehållsförteckning

Inledning

Jag har valt att skriva ett program som utifrån webbsidor genererar uppläsbar text och anropar talsyntesprogrammet Festival som läser upp texten. Målet var att göra ett användarvänligt och väl fungerande program samt att ta reda på vilka hjälpmedel som redan finns för t ex blinda och synskadade för att navigera på internet. Jag har skrivit i programspråket Java, eftersom det är det jag har mest erfarenhet av att programmera i.

I uppsatsen beskriver jag hur jag har gått till väga när jag skrev programmet och hur det fungerar. Jag behandlar även andra liknande program och alternativa lösningar, utvecklingsmöjligheter och olösta problem hos mitt eget program.

För att göra programmet maximalt användarvänligt skulle man kunna lägga till taligenkänning så att man skulle slippa trycka på några tangenter och istället säga till programmet vilken sida man vill ha uppläst och vilka länkar man vill följa. Detta är dock inget jag från början tänkte göra och det skulle antagligen kräva minst lika mycket tid till som jag redan lagt ner på den här uppsatsen.

Webbsidors tillgänglighet

För att göra webbsidor tillgängliga för så många som möjligt har www-konsortiet (w3c) dragit upp ett antal riktlinjer för hur man skriver webbsidor som är tillgängliga i så stor utsträckning som möjligt (http://www.w3.org/TR/WAI-WEBCONTENT).

En av de viktigaste punkterna säger att man bör tillhandahålla motsvarande, alternativ information för auditoriskt och visuellt innehåll. Detta gäller t ex alternativa texter till bilder. När sådana finns kan även blinda och synskadade ta del av informationen som bilderna innehåller, förutsatt att det program de använder läser upp de alternativa texterna. Mitt program läser upp alternativa texter till bilder när sådana förekommer, men på sidor med många bilder utan alternativa texter blir den tillgängliga informationen väldigt begränsad.

En annan viktig punkt säger att man inte ska använda tabeller för att presentera icke-tabulär information. Detta beror på att just tabeller ofta medför problem vid användning av skärmläsare och talande webbläsare. Om man ändå vill använda tabeller för layout som inte kan läsas rad för rad, bör man tillhandahålla en alternativ version. Detta kan vara t ex en linjär version av tabellen. Mitt program klarar endast att läsa tabeller rad för rad och skulle därför oftast återge informationen bättre med en alternativ version.

Att ange det huvudsakliga språket i ett dokument samt att tydligt ange när språket i ett dokument ändras är också viktigt. När det gäller dokumentets huvudsakliga språk har mitt program en reservfunktion som letar efter enskilda ord som är typiska för svenska respektive engelska. Om det huvudsakliga språket är angivet i dokumentets huvud slipper programmet gissa och risken för att det försöker läsa upp dokumentet på fel språk minimeras. Programmet utgår från att hela det dokument det läser är skrivet på samma språk. Genom att leta efter kod som säger att språket ändras, skulle det kunna skicka information till talsyntesprogrammet Festival om att tillfälligt ändra uppläsningsspråk. När koden sedan anger att det huvudsakliga språket återigen gäller skulle programmet kunna skicka ny information till Festival om att ändra tillbaka till utgångsspråket.

Kort introduktion till HTML

De flesta webbsidor skrivs i ett märkspråk som heter HTML och betyder HyperText Markup Language. Mitt program är skrivet för att kunna läsa och generera uppläsbar text utifrån HTML-dokument.

För att även de som inte är insatta i HTML ska kunna förstå programbeskrivningen nedan följer här en kort introduktion till HTML.

Ett HTML-dokument är indelat i textelement som har hand om olika saker. Alla textelement börjar med tecknet ’<’ följt av elementets typ och tecknet ’>’ och slutar på samma sätt fast med

ett snedstreck precis före typen. Snedstrecket betyder alltså att textelementet slutar där. Det som står utanför vinkelparenteserna är brödtext som visas på webbsidan, medan det som står inom vinkelparenteserna, märkorden, beskriver hur text, bilder mm ska se ut på skärmen och vilken funktion de har. Där finns t ex information om vilket typsnitt och vilken teckenstorlek texten ska ha, var på skärmen texten ska visas och vilken färg texten ska ha. Där kan också finnas information som talar om att texten är t ex en rubrik, ett citat eller en förkortning eller att ett ord eller en mening ska betonas särskilt. Vid bilder kan där finnas alternativa texter som beskriver bilderna och vid länkar finns där information om länkarnas adress.

Först i dokumentet finns ett huvud som innehåller information om dokumentet som helhet. Här kan bl a finnas information om vilket språk dokumentet är skrivet på och vilket program som använts för att skapa dokumentet. Body är det största textelementet som innehåller informationen som visas på webbsidan. I bodyn finns ett antal andra textelement som kan innehålla text, tabeller, bilder, länkar mm. En webbsida kan innehålla flera ramar, s.k. frames, som hänvisar till sidor som i sin tur innehåller var sin body. Ramar används ofta när man har en meny med länkar till delsidor inom en sida. Då ändras inte innehållet i menyn när man klickar på en länk utan endast det övriga innehållet på sidan.

Nedan följer ett exempel på hur en webbsida kan vara uppbygd med HTML.

```
<html>
<head>
<meta http-equiv="Content-Language" content="sv">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Kamrérbaletten</title>
</head>



<body bgcolor="#000000" text="#FFFFFF" link="#FFFFFF" vlink="#FFFFFF">

<p align="center"><b><font size="4" face="Galant">Välkommen till</font></b></p>
<p align="center"><b><font size="7" face="Galant">Kamrérbaletten</font></b></p>
<p align="center"><font face="Galant"><img border="0" src="stork5.JPG" width="416"
height="277"></font></p>
<p align="center"><font face="Galant" size="4">På studentorkesterfestival i Uppsala
12-14 maj 2000</font></p>
<p align="center"><b><font size="4" face="Galant">Balettens motto:</font></b></p>
<p align="center"><b><font size="7" face="Galant">ÄT MER SÅ SYNS VI
BÄTTRE!</font></b></p>
<p align="center"> </p>
<p align="center"> </p>

</body>

</html>
```

Alternativa märkspråk

Eftersom HTML är gjort främst för visuell layout utvecklas nya märkspråk speciellt utvecklade för talkommunikation. Olika företag har utveckat sina egna märkspråk för talkommunikation med olika namn. IBM:s ursprungliga språk hette SpeechML, AT&T och Lucent har båda utvecklat var sitt språk med samma namn: PML (Phone Markup Language) och Motorolas ursprungliga språk hette VoxML. Tillsammans har IBM, AT&T, Lucent och Motorola sedan

utvecklat VoiceXML, som www-konsortiet har använt som modell när de utvecklade Dialog ML. (http://www.w3.org/Voice/)

Tanken är att man ska kunna få tillgång till internet via vanliga telefoner genom användning av talsyntes, inspelat tal och taligenkänning. På så vis skulle tillgängligheten öka dramatiskt, eftersom många fler har tillgång till telefon än till dator och internetuppkoppling.

Talsyntesprogrammet Festival

Festival är ett flerspråkigt talsyntesprogram för omvandling av text till tal. Det är skrivet i programspråket C++ med en Scheme-baserad kommandotolk och stöder märkspråket Sable som är baserat på XML. Genom att sätta olika värden till olika parametrar kan man få det syntetiserade talet att låta olika. Intonationsparameterns start- och slutdefaultvärden är 130 Hz resp. 110 Hz, men genom att sätta dem till andra värden kan man ändra intonationen. Durationsparameterns defaultvärde är 100 millisekunder för varje segment, men detta kan också ändras genom att sätta ett annat värde för att öka eller minska durationen. Genom att sätta olika värden till röstparametern kan man ändra språket som texten ska läsas upp på. För flera språk finns det flera olika röster att välja mellan och i språkparametern anges vilken röst som ska användas. För att kunna läsa upp texter som är olika strukturerade på olika sätt finns det flera olika s.k text-modes. När ett visst text-mode används läser Festival upp texten på det sätt som det aktuella text-modet anger. Man kan själv skapa ett nytt text-mode om man har en text som är strukturerad på ett särskilt sätt. (http://www.cstr.ed.ac.uk/projects/festival/)

Mitt program

När man startar programmet uppmanas man först att skriva in adressen till den sida man vill ha uppläst. Om sidan inte hittas, avslutas programmet och ett felmeddelande visas på skärmen. Annars börjar programmet läsa igenom sidan och skapa en textfil som senare ska läsas upp.

Programmet har två olika metoder (subrutiner i Java) för att läsa in text från HTML-dokumentet. Den ena metoden, readLink, läser märkorden innanför vinkelparenteserna och den andra, readText, läser brödtexten som står utanför. De anropas varannan gång tills hela dokumentet har lästs igenom. Ibland är strängen som readText läser in tom och då händer ingenting, utan readLink anopas igen och så fortsätter det.

ReadLink kollar varje gång ifall strängen den läser in betyder att något särskilt ska göras. Här följer en förteckning över alla möjliga textelement som finns (http://www.w3.org/TR/REC-html40/index/elements.html) och vilka åtgärder programmet tar för varje element:

Beteckning	Betydelse	Åtgärd
a	Ankare	Om elementet följs av attributet ”href” - ange att här finns en länk till en annan webbsida samt länkens nr och skapa ett nytt javaobjekt av klassen Link så att man kan följa länken senare
abbr	Förkortning	Ingen
acronym	Akronym	Ingen
address	Information om författaren	Ingen
applet	Java applet	Ange att en applet finns men hoppas över
area	Specificerar en bildkartas geometriska regioner	Ingen

b	Fet stil	Ingen
base	Dokumentbas URI	Ingen
basefont	Base-storlek	Ingen
bdo	Åsidosätter BiDi-algoritmen	Ingen
big	Stor text	Ingen
blockquote	Långt citat	Ingen
body	Dokumentkropp	Vid elementets slut, sluta läsa dokument
br	Radbrytning	Ingen
button	Knapptryckning	Ingen
caption	Tabellbeskrivning	Ange att där finns en tabellbeskrivning samt skriv ut tabellbeskrivningen
center	Förkortning för DIV align=center	Ingen
cite	Citat eller åberopande av annan källa	Ingen
code	Fragment av datorkod	Ingen
col	Tabellkolumn	Ingen
colgroup	Grupp av tabellkolumner	Ingen
dd	Definitionsbeskrivning	Ange att där finns en definitionsbeskrivning samt skriv ut definitionsbeskrivningen
del	Borttagen text	Ingen
dfn	Instansdefinition	Ingen
dir	Biblioteksklista	Ingen
div	Allmän språk/stil-behållare	Ingen
dl	Definitionslista	Ange att där finns en definitionslista
dt	Definitionsterm	Ange att där finns en definitionsterm samt skriv ut definitionstermen
em	Tonvikt, betoning	Ingen
fieldset	Grupp av formulärhanterare	Ingen
font	Lokal ändring av typsnitt	Ingen
form	Interaktiv form	Ingen
frame	Delsida	Spara adressen till delsidan för senare uppläsning. Vid elementets slut – sluta läsa dokument
frameset	Mängd av delsidor	Ingen
h1	Rubrik 1	Ingen
h2	Rubrik 2	Ingen
h3	Rubrik 3	Ingen
h4	Rubrik 4	Ingen
h5	Rubrik 5	Ingen
h6	Rubrik 6	Ingen
head	Dokumenthuvud	Ingen
hr	Horisontalt streck	Ingen
html	Första elementet	Ingen
i	Kursiv stil	Ingen
iframe	Sida i sidan	Ingen
img	Bild	Ange att där finns en bild samt eventuell alternativ text
input	Formulärhanterare	Ingen
ins	Insatt text	Ingen

isindex	Enradsprompt	Ingen
kbd	Text som ska skrivas in av användaren	Ingen
label	Ledtext	Ingen
legend	Tillåter beskrivning av fieldset	Ingen
li	Listelement	Ingen
link	Referens till annat dokument	Ingen
map	Bildkarta	Ingen
menu	Menylista	Ingen
meta	Allmän metainformation	Ingen
noframes	Alternativ framställning för program som inte kan läsa frames	Ingen
noscript	Alternativ framställning för program som inte kan läsa scripts	Läs istället för script
object	Allmänt inbäddat objekt	Ingen
ol	Ordnad lista	Ange att där finns en ordnad lista
optgroup	Grupp av valmöjligheter	Ingen
option	Valmöjlighet	Ingen
p	Paragraf	Lägg till en paus i textfilen
param	Parametrar i applet-element	Ingen
pre	Förformatterad text	Ingen
q	Kort citat	Ingen
s	Genomstruken textstil	Ingen
samp	Testoutput från program, scripts mm	Ingen
script	Manus	Hoppa över all text i detta textelementet
select	Skapar en meny med valmöjligheter	Ingen
small	Liten stil	Ingen
span	Allmän språk/stil-behållare	Ingen
strike	Genomstruken text	Ingen
strong	Stark betoning	Ingen
style	Stilinformation i form av stilmallar	Hoppa över all text i detta textelement
sub	Index	Ingen
sup	Exponent	Ingen
table	Tabell	Ange att där finns en tabell
tbody	Tabellkropp	Ingen
td	Datacell i tabell	Ingen
textarea	Flerrads textfält	Ingen
tfoot	Tabellfot	Ingen
th	Rubrikcell i tabell	Ingen
thead	Tabellhuvud	Ingen
title	Titel	Ange titeln
tr	Tabellrad	Ingen
tt	Textstil teletype eller	Ingen

	monospace	
u	Understruken textstil	Ingen
ul	Oordnad lista	Ange att här finns en oordnad lista
var	Variabel eller programargument	Ingen

Programmet bryr sig bara om några få av alla textelement som finns. Detta beror bl a på att många av textelementen beskriver att texten ska se ut på ett särskilt sätt och eftersom texten som programmet genererar ska läsas upp bryr det sig inte om ifall texten är t ex fet eller kursiv.

För att veta när hela dokumentet har lästs igenom kollar readLink om den aktuella strängen som lästs in är ”/body” eller ”/frame”. När den stöter på någon av dessa strängar avbryts inläsningen.

Så länge ingen sträng med något av de två prefixen ovan påträffas läser programmet framåt i filen och skickar relevant text till textfilen som senare ska läsas upp.

I HTML-dokumentets huvud, som finns före body, finns ofta information om vilket språk dokumentet är skrivet på. Programmet letar därför efter denna information för att veta på vilket språk sidan ska läsas upp. Eftersom det inte är helt säkert att det hittar den sökta informationen i huvudet, räknar readText förekomsterna av orden ”och” och ”and” för att kunna gissa om sidan är skriven på svenska eller engelska.

När programmet träffar på en länk skapas ett nytt javaobjekt av klassen Link som representerar den aktuella länken. Där sparas information om länkens namn och adress. Länkobjektet får också ett nummer som fungerar som ett idnummer. Till textfilen skickas en mening som talar om att där finns en länk och om den har något namn eller är kopplad till en bild eller en emailadress anges även detta.

När programmet träffar på en bild kollar det om det finns en beskrivning till bilden. Sedan skickas en mening som talar om att där finns en bild och eventuell beskrivning till textfilen.

Brödtexten skickas till textfilen för uppläsning, men först kollar programmet om den innehåller några strängar som representerar å, ä eller ö. I HTML-kod representeras nämligen dessa tecken med konstiga strängar (”ö” används t ex för att representera ’ö’) som när de hittas av programmet byts ut mot rätt tecken. Samtidigt räknar programmet hur många ’å’, ’ä’ och ’ö’ det hittar på varje sida för att senare veta säkert om sidan är svensk eller inte, ifall ingen information om detta hittats i huvudet. Det finns många fler tecken som representeras på liknande sätt, men jag tar bara hand om några få eftersom de flesta förekommer väldigt sällan.

Programmet letar också efter strängar som innehåller tecknet ’@’ och punkter som inte efterföljs av blanktecken. Sådana strängar är vanliga på webbsidor eftersom både webbadresser och mailadresser är uppbyggda så, men talsyntesprogrammet Festival klarar inte att läsa sådana strängar. Därför görs de om så att tecknet ’@’ byts ut mot strängen ” snabel a ” och punkter som inte efterföljs av blanktecken byts ut mot strängen ” punkt ” så att Festival ska kunna läsa upp det. Strängen ”webmaster@lu.se” görs tex om till ”webmaster snabel a lu punkt se”. Även strängar som innehåller bindestreck kan innebära problem vid uppläsningen. Därför söker programmet också efter sådana strängar och ersätter bindestrecket med lämplig sträng utifrån kontexten.

När hela sidan lästs igenom skickar programmet en fråga till textfilen huruvida man vill följa någon av de påträffade länkarna, höra sidan igen eller avsluta programmet. Därefter kommer namnen på alla länkarna i nummerordning efter deras idnummer. Frågan är formulerad så att man uppmanas att trycka in ett länknummer om man vill följa någon länk, trycka på s om man vill ha sidan uppläst en gång till eller trycka på a om man vill avsluta programmet.

Sedan kollar programmet ifall information om vilket språk sidan är skriven på hittades i huvudet. Om ingen information hittades kollar det hur många ’och’, ’and’ och ’å’, ’ä’ och ’ö’ det har stött på och om det har hittat fler ”and” än ”och” och det inte finns några ’å’, ’ä’ eller ’ö’ gissar det att sidan är skriven på engelska. Annars gissar det att sidan är skriven på svenska, för några andra alternativ har det inte.

När det har bestämt sig för vilket språk sidan ska läsas upp på görs en liten textfil som innehåller kommandon till talsyntesprogrammet Festival. I filen anges om texten ska läsas på svenska eller engelska och sedan anropas Festival med den stora textfilen som skapades när programmet läste igenom sidan.

När man har hört hela sidan och frågan på slutet väljer man att avsluta, höra sidan igen eller följa en länk. Om man väljer att följa en länk börjar programmet om med den valda länken som ny input.

Andra liknande program

När jag har letat efter liknande program har jag funnit att den sorts program som blinda och synskadade använder mest är s.k. skärmläsare. Både på svenska Hjälpmedelsinstitutets hemsida, http://www.hi.se/default.shtm, och på den brittiska sidan för blinda (Royal National Institute for the Blind), http://rnib.org.uk/, kan man läsa om olika skärmläsare som det främsta datorhjälpmedlet för blinda och synskadade. Skärmläsare är ett program som fångar upp den information som datorn presenterar på bildskärmen och presenterar den som syntetiskt tal eller som punktskrift på en punktskriftsskärm. Skärmläsarna fungerar även när man kör andra program än webbläsare och har därför ett brett användningsområde. Eftersom de bara ser det som visas på skärmen är skärmläsarna beroende av det program som körs. Webbläsaren Netscape Navigator visar t ex inte ALT-texterna (de beskrivande alternativtexter som ibland finns till bilder) och därför har skärmläsarna ingen chans att veta vad bilderna på webbsidor innehåller och användaren får därmed heller ingen information om detta.

Jag har också hittat ett antal program vars uppgift är att presentera just webbsidor med hjälp av tal. Nedan följer en kort presentation av några av dem.

Företaget Phoneticom (http://www.phoneticom.com/) utvecklar produkter som gör att man kan få webbsidor upplästa via dator eller telefon. ISI Browser är en tjänst till företag som gör att man kan lyssna på deras hemsida utan kostnad. Det är företaget som betalar Phoneticom för tjänsten, som gör att deras hemsida blir mera tillgänglig. ISI Phone Access är ett system som gör webbsidor tillgängliga via telefon.

Jag har testat en tidig version av ISI Browser och den fungerar bra. Man kan välja att få bara text, bara länkar eller allt på sidan uppläst. Man väljer genom att klicka på någon av ikonerna på Phoneticoms panel som visas överst på sidan. Om man vill följa en länk skriver man in länkens nummer i ett inmatningsfält och klickar på knappen ”Gå till”. Ljudkvaliteten är ganska dålig. Företagets policy är att man inte ska behöva ha en snabb dator, bredband eller behöva ladda ner

några program för att använda tjänsten och därför är ljudet väldigt komprimerat, vilket gör att det blir brus.

VocalPoint Technologies (http://www.vocalpoint.com/index.html) erbjuder företag att skapa röststyrda tjänster för existerande webbinformation. VocalPoint VoiceBrowser gör talkommunikation med naturligt tal möjlig från en vanlig telefon till en existerande webbsida med HTML- eller XML-kod. På företagets webbsida finns en demonstration av VocalPoint VoiceBrowser som visar hur en italiensk användare kan få trafikinformation genom telefonen direkt från en webbsida.

One Voice Technologies (http://www.onevoicetech.com/index.html) har utvecklat IVAN (the Intelligent Voice Animated Navigator), som är en liten animerad gubbe som hjälper en att navigera på internet. IVAN förstår naturligt tal och talar själv ganska naturligt. Man kan be honom söka efter specifika saker eller sidor eller berätta om olika saker på en webbsida.

Analys

Här kommer jag att ta upp möjliga alternativa lösningar, utvecklingsmöjligheter och olösta problem.

Alternativ

När man tittar på en webbsida läser man oftast inte allt som står på sidan utan bara valda delar. Det finns bl a en undersökning gjord av Jakob Nielsen och John Morkes där bara 16 % av testanvändarna läste webbsidor ord för ord och hela 79 % scannade av nya webbsidor som de såg för första gången. (http://www.useit.com/alertbox/9710a.html) Istället för att låta programmet läsa upp en hel sida på en gång skulle man därför kunna låta det läsa upp valda delar av en sida. Detta skulle kräva att programmet först läste igenom sidan en gång och sparade informationen i t ex olika textfiler så att det, sedan användaren valt vilka delar som skall läsas upp, kunde anropa Festival med de aktuella textfilerna. För att veta vad som över huvud taget finns på sidan måste emellertid användaren först få en överblick för att kunna välja vilka delar man vill höra. Detta skulle kräva att man först presenterade ett antal rubriker eller, i brist på detta, inledande meningar för varje stycke. Alternativt skulle man när som helst under uppläsningen av hela sidan kunna avbryta och hoppa till nästa stycke. Detta skulle i princip kräva att programmet skickade texten mening för mening till Festival och mellan varje mening kollade ifall användaren hade tryckt ner någon tangent som betydde att man skulle hoppa till nästa stycke.

Man skulle också kunna göra som Phoneticom och erbjuda möjligheten att välja att höra bara text, bara länkar eller allt på sidan. Detta skulle kunna åstadkommas genom att programmet, medan det läste igenom sidan, sparade informationen i tre olika textfiler; en med bara text, en med bara länkar och en med både text och länkar. Sedan användaren valt vad som skulle läsas upp skulle programmet kunna anropa Festival med den textfil som innehåller vald information.

En annan sak som man skulle kunna ändra på för att öka användarvänligheten en aning är att erbjuda användaren att när som helst kunna skriva in adressen till en helt ny webbsida. Som det är nu måste man först avsluta programmet och sedan starta det igen, vilket kan verka ganska klumpigt. För att avhjälpa detta skulle man kunna utöka alternativen som redan finns (höra sidan igen, följa en länk, avsluta programmet) med ett fjärde alternativ; att höra en helt ny

webbsida. När detta fjärde alternativ valts skulle programmet starta om från början och glömma alla gamla länkar och adresser som det tidigare sparat.

Istället för att skriva all källkod själv skulle jag ha kunnat skriva ett nytt text-mode för HTML till talsyntesprogrammet Festival och sedan använda det för att läsa upp webbsidor. Ett annat alternativ hade varit att använda en färdig HTML-parser för att strukturera upp webbsidor i en trädstruktur så att programmet lättare hade kunnat hoppa mellan olika delar av en sida och endast läsa upp valda delar. Anledningen till att jag inte valt det första alternativet är att jag då skulle ha blivit tvungen att lära mig Scheme och framför allt att jag upptäckte möjligheten först när programmet var nästan färdigt. Anledningen till att jag inte valt det senare alternativet är att jag inte hittat någon parser som jag kunde installera och köra hemma (där jag har gjort den största delen av uppsatsen) och som lämnade en output som var lämplig att använda (javaklasser eller rena textfiler).

Utvecklingsmöjligheter

Ur användarvänlighetssynpunkt finns det flera delar av programmet som skulle kunna utvecklas och byggas på. En del nämnde jag redan i inledningen, nämligen att inte bara använda talsyntes utan att även koppla in taligenkänning så att användaren efter att ha startat programmet inte skulle behöva använda varken tangentbord, mus eller bildskärm. Ett sådant program skulle kunna användas även av personer med nedsatt rörelseförmåga och dessutom vara enklare för alla med en fungerande talapparat att använda, förutsatt att taligenkänningen fungerar tillfredställande.

En annan del som man skulle kunna utveckla är möjligheten att direkt följa länkar till mailadresser. Som programmet är skrivet nu, talar det om när det hittar en länk till en mailadress och namnet på adressen, men man har ingen möjlighet att direkt följa länken och skicka ett mail till adressen. Man skulle alltså kunna utveckla en metod (subrutin i Java) som utför det som händer när man klickar på en länk till en mailadress i en vanlig webbläsare, nämligen öppna ett mailprogram och en ruta för att skicka ett nytt email. Detta borde inte vara omöjligt, men min nuvarande kunskap räcker inte till för att beskriva hur man skulle gå till väga för att skriva en sådan metod.

Vid listor, citat och betonade ord eller meningar skulle ett lite tätare samarbete mellan mitt program och Festival kunna bidra till större känsla i uppläsningen.

När man läser upp en lista använder man prosodi för att tala om ifall det kommer fler punkter eller om det är den sista man just läser upp. Med rätt direktiv till Festival skulle man kunna få ungefär samma effekt vid uppläsning av en lista på en webbsida. Detta skulle man kunna åstadkomma genom att man varje gång man hittar ett textelement som talar om att där finns en lista av något slag, avslutar aktuell textfil och skickar den till Festival och därefter läser igenom hela listan så att man vet hur många listelement den innehåller. Sedan skulle man kunna anropa Festival med en fil som innehåller prosodiska direktiv som stämmer för alla utom det sista listelementet, en textfil med alla utom det sista listelementet och till sist prosodiska direktiv som ställer in normal prosodi igen. Sedan skulle det sista listelementet kunna vara början på nästa textfil och läsas upp som vanligt oberoende av vad som kommer efter. Vid definitionslistor markerade med textelementet ’dl’ skulle man kunna ange en särskild typ av intonation som Festival skulle använda för just den här typen av listor. Vid de olika elementen ’dt’ och ’dd’ i en definitionslista skulle man också kunna använda sig av olika prosodiinställningar för att få det att låta mer naturligt.

Vid enstaka betonade ord eller meningar skulle man kunna göra ungefär likadant. När programmet upptäcker något av textelementen ’em’ eller ’strong’ skulle den aktuella textfilen avslutas och skickas till Festival. Därefter skulle man kunna anropa Festival med en fil som innehåller direktiv om att ändra betoningsgraden (olika grad beroende på om ’em’ eller ’strong’ påträffats), en textfil med ordet eller meningen som ska betonas samt direktiv om att ändra betoningsgraden till normal igen.

Vid citat som är markerade genom textelementen ’q’, ’cite’ och ’blockquote’ skulle man kunna ändra röst i Festival, så att själva citatet läses upp med en annan röst än den som läser den övriga texten. Ett annat alternativ är att använda prosodiförändring för att visa att det är ett citat. Man skulle också kunna lägga till ordet ”citat” före citatet och ”slut på citat” efter för att ytterligare förtydliga var citatet börjar och slutar.

Möjligheten att backa till föregående sida finns inte i programmet som det ser ut nu. Det skulle man kunna lösa genom att lägga till ett alternativ, backa, till valmöjligheterna och om detta alternativ väljs använda adressen till den förra sidan som man sparat i en variabel.

Olösta problem

När man använder en vanlig webbläsare kan man ibland skriva in text i särskilda textfält. Detta behövs t ex när man fyller i formulär eller använder användaridentitet och lösenord för att komma åt personliga uppgifter. Mitt program har inget sätt att hantera sådana situationer och detta är en stor begränsning.

Tabeller kan vara svåra att förstå eftersom de läses rad för rad.

Slutsats

När mer och mer av informationssökandet sker på internet ökar behovet av att göra informationen på webbsidor tillgänglig för fler. Jag har själv upptäckt, när jag skrev mitt progam, att det är ganska klumpigt och svårt att göra informationen på vanliga webbsidor, skrivna i HTML, tillgänglig via talsyntes. Trots att det finns riktlinjer för hur man gör webbsidor mer tillgängliga, är de allra flesta webbsidor skapade helt utan vetskap om dessa riktlinjer och därmed ganska otillgängliga.

Eftersom det visat sig att HTML inte är särskilt väl lämpat för talkommunikation utvecklas nya märkspråk, särskilt utvecklade för talkommunikation, såsom VoiceXML. Jag tror att detta i framtiden kommer att leda till att åtminstone en del av informationssökandet på internet kommer att ske via telefon.

Referenser

		Besökt senast
Phoneticom AB(företag)	http://www.phoneticom.com/	010603
VocalPoint Technologies (företag)	http://www.vocalpoint.com/index.html	010603
One Voice Technologies (företag)	http://www.onevoicetech.com/index.html	010603
Svenska Hjälpmedelsinstitutet	http://www.hi.se/default.shtm	010603
Useit.com Jakob Nielsens webbsida	http://www.useit.com/alertbox/9710a.html	010603
Royal National Institute for the Blind	http://rnib.org.uk/	010527
www-konsortiet	http://www.w3.org/	010603
Synskadades Riksförbund	http://www.srfriks.org/	010603

Dokumentation av Festival http://www.festvox.org/docs/manual-1.4.1/festival_toc.html 010605
Festivals hemsida http://www.cstr.ed.ac.uk/projects/festival/ 010604