Introduktion till XHTML och CSS

Övningar

Mikael Gunnarsson

27 oktober 2020

Följande övningar baseras på tidigare beprövade övningar. Illustrationer ges med hänsyn till att du använder OXYGEN men kan följas även med andra editorer efter att du själv anpassar anvisningarna till vad som kan gälla där.

I materialet ges emellanåt informationsrutor med en lila-aktig bakgrund. De finns där bl a som extra kommentarer på basis av vanliga misstag som vi fått erfara genom åren.

1 Inledning

I dessa övningar skall du försöka identifiera en trädstruktur och dess element i en given oformaterad text och därefter märka upp den med den repertoar av HTML-element som står till buds. Du ska bekanta dig med textstruktur, hyperlänkar, samt hur du kan infoga bilder i en text. Det är säkert frestande att fråga hur man kan göra si eller så för att få grafiskt mer tilltalande form. Vi väntar emellertid med den saken helt och hållet tills vi introducerat CSS.

Hädanefter kommer vi för enkelhets skull att emellanåt tala i termer av html och inte XHTML.

> *Uttrycket 'XHTML' avser alltid XML-varianter av äldre SGML-varianter som väsentligen är identiska.* `XHTML 1.0 Strict` *är identisk med* `HTML 4.01 Strict` *men i XML-syntax. I båda fallen är dock rotelementets namn* `html`*.*

En html-fil kan precis som en XML-fil produceras med vilken texteditor som helst (men observera vad som sagts i inledningen till de första XML-övningarna, om tec-

kenkodning). Här utgår vi dock från att du använder OXYGEN. Starta därför OXYGEN och välj *New* från *File*-menyn. Välj därefter i listan `XHTML 1.0 Strict`. OXYGEN (eller EDITIX) kommer nu att infoga en grundläggande uppmärkning som liknar följande:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title></title>
7      </head>
8      <body>
9      </body>
10 </html>
```

I formellt avseende är detta en korrekt bas att utgå ifrån. Alla html-dokument har ett rotelement `html` i vilket ingår ett `head` och ett `body`, samt att det i elementet `head` är obligatoriskt med ett element `title` som givetvis skall innehålla en titel (det är den som framträder i webbläsarfönstrets titeletikett och vanligen används av sökmaskiner vid listning av träffar). Emellertid är det i praktiken fördelaktigt att något revidera uppmärkningen. Dels är den första xml-deklarationen (rad 1) med sin deklaration av teckenkodning inte nog för alla webbläsare, utan dessa behöver ytterligare en instruktion i ett s k meta-element som är definierat som ett tomt element. I själva verket är det lika bra att exkludera xml-deklarationen av skäl som vi tills vidare ignorerar. Vidare kan det vara fördelaktigt att ange det språk som dokumentet är skrivet på, med ett attribut till rotelementet. Notera också namnrymdsdeklarationen med `xmlns` på rad 4. Modifiera därför enligt följande:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en" lang="en">
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html;
        charset=utf-8" />
    </head>
    <body>
    </body>
</html>
```

I exemplet ovan kan du se att utf-8 har valts som teckenkodning. Valet av teckenkodning är inte ovidkommande och kan vara ett besvärligt kapitel. I många fall räcker det att göra som ovan, och helt enkelt bara tala om vad man har använt, men vissa webbservrar kan ställa till problem om de inte är konfigurerade att beakta vad som sägs i ett meta-element enligt ovan. Varje webbserver skickar alltid med en s k *http-header*, som du som användare aldrig ser, där teckenkodning skall anges, och där vad som anges kan ges prioritet över vad som sägs i dokumentets meta-element. Ett sätt att inte behöva bekymra sig om detta, är att ersätta alla svenska tecken (och andra icke-engelska) med teckenentiter, å med å, ä med ä och ö med ö Men det är i de flesta fall helt onödigt och gör texten svår att redigera.

Det är nu dags att fylla ”skelettet” med innehåll. Kopiera och klistra in från `https://raw.githubusercontent.com/Mikael61/htmlAndCSS/main/cmplglixcol.txt` texten mellan de båda `body`-märkena och spara filen som exempelvis `karlgren.htm` på en plats i filsystemet där du kan hitta tillbaka till den. Observera att filnamnet skall ha ändelsen `htm` eller `html` (`xhtml` kan också passera), och endast bör innehålla alfanumeriska tecken utan diakriter. **Mellanslag bör undvikas i filnamn**. Förse också dokumentet med en lämplig titel, dvs skriv något i elementet `title`, förslagsvis samma som artikelns titel.

Den ursprungliga artikeln finns på `http://soda.swedish-ict.se/56/1/cmplglixcol.pdf`. Observera att figurer och tabeller inte finns med i textversionen. Vi ska tills dels ta hand om dessa lite senare. Observera också att illustrationen nedan endast redovisar en del av texten.

Innehållet har placerats inuti ett element `div` som i sin tur ligger inuti `body`. Det är för att exemplet fortsatt ska vara giltigt. Elementet `body` får nämligen inte innehålla text, utan endast andra element (och bara vissa element).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="sv" lang="sv">
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html;
        charset=utf-8" />
    </head>
    <body>
      <div>
      Recognizing Text Genres with Simple Metrics Using Discriminant Analysis

Jussi Karlgren jussi@sics.se Swedish Institute of Computer Science Box
1263, S -- 164 28 Kista, Stockholm, Sweden


Douglass Cutting cutting@apple.com Apple Computer Cupertino, CA 95014, USA

October 25, 2005


    Abstract

A simple method for categorizing texts into pre-determined text genre
categories using the statistical standard technique of discriminant
analysis is demonstrated with application to the Brown corpus.
Discriminant analysis makes it possible use a large number of parameters
that may be specific for a certain corpus or information stream, and
combine them into a small number of functions, with the parameters
weighted on basis of how useful they are for discriminating text
genres. An application to information retrieval is discussed.


    Text Types

There are different types of text. Texts 'about' the same thing may be
in differing genres, of different types, and of varying quality. Texts
vary along several parameters, all relevant for the general information
retrieval problem of matching reader needs and texts. Given this
variation, in a text retrieval context the problems are (i) identifying
genres, and (ii) choosing criteria to cluster texts of the same genre,
with predictable precision and recall. This should not be confused with
the issue of identifying topics, and choosing criteria that discriminate
one topic from another. Although not orthogonal to genre-dependent
variation, the variation that relates directly to content and topic is
along other dimensions. Naturally, there is co-variance. Texts about
certain topics may only occur in certain genres, and texts in certain
genres may only treat certain topics; most topics do, however, occur in
several genres, which is what interests us here.

Douglas Biber has studied text variation along several parameters, and
found that texts can be considered to vary along five dimensions. In his
study, he clusters features according to covariance, to find underlying
dimensions (1989). We wish to find a method for identifying easily
computable parameters that rapidly classify previously unseen texts in
general classes and along a small set -- smaller than Biber's five -- of
dimensions, such that they can be explained in intuitively simple terms
to the user of an information retrieval application. Our aim is to take
a set of texts that /has/ been selected by some sort of crude semantic
analysis such as is typically performed by an information retrieval
system and partition it /further/ by genre or text type, and to display
this variation as simply as possible in one or two dimensions.


/.../

      </div>
    </body>
</html>
```

Spara det du gjort. Växla till Utforskaren (eller Finder i Mac-miljö), leta rätt på

filen och öppna den med en valfri webbläsare. Eftersom texten inte har någon uppmärkning alls blir det en enda lång ström av ord som läggs ut i webbläsarfönstret. Karaktäristiskt för såväl XML som SGML är också att tabbar, serier av mellanslag och radbrytningstecken i normalfallet ignoreras när text skall renderas på skärm.

Betrakta nu originaltexten i pdf-form och fundera en liten stund på hur den kan analyseras med avseende på innehållselement. Vilka element kan du identifiera och vad skulle du kalla dem. Det kan ses som en olägenhet att vi endast har htmls cirka 100 element att välja från, och ingen av de element vi själva identifierar och namnger tycks finnas. Det finns dock sätt att komma tillrätta med det, som vi ska se framöver.

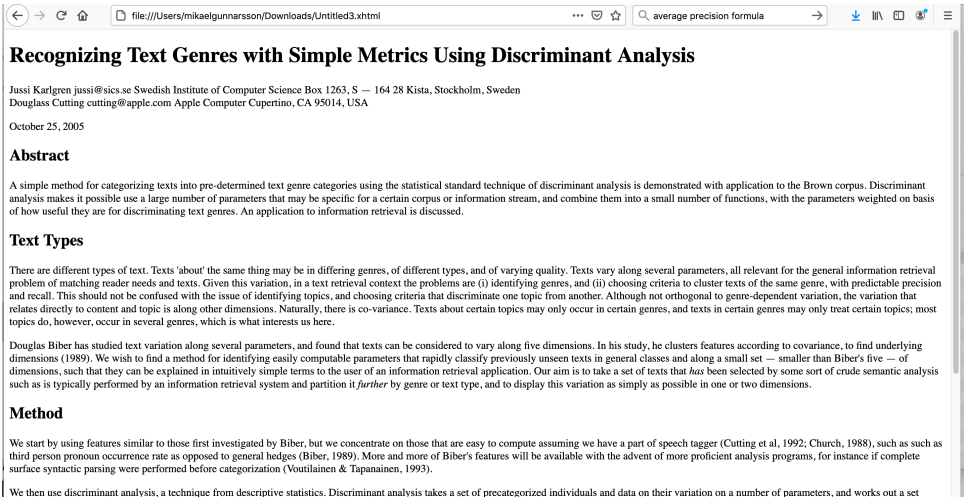
Även för en relativt komplex text klarar du dig faktiskt med ett fåtal element ur html-repertoaren

Utöver de element vi redan sett i skelettet ovan så ska du nu söka identifiera och märka upp rubriker och stycken. För rubriker räcker i allmänhet tre nivåer (elementen `h1`, `h2` och `h3`). Stycken delas in med `p`. Alla dessa är normala s k blockelement och måste stängas. Kanske funderar du över upphovsuppgifterna allra först i texten och platshållarna (”captions”) för tabeller och figurer. Behandla dem som stycken tills vidare.

Alla dessa element är normalt av typen *block*, vilket innebär att de vid rendering på skärm eller papper alltid föregås och efterföljs av en radbrytning. Blockelement kan normalt inte placeras sida vid sida.

Använd nu alltså dessa för att märka upp texten. Markera respektive avsnitt av text som du avser att märka upp med något av elementen, välj ur Document - Markup - Surround by tags (eller dess motsvarande snabbkommando) och därefter det element du vill märka upp avsnittet med.

Granska nu din text i webbläsaren som tidigare, utan att stänga OXYGEN och genom att endast uppdatera webbläsarens innehåll (vilket görs på olika sätt beroende på i vilken miljö du sitter, F5 eller Command+R fungerar ofta). Sannolikt liknar den ungefär nedanstående skärmdump.



Här finns en betydelsefull skillnad mellan andra xml-tillämpningar och xhtml som kräver en förklaring. Så här långt har vi inte definierat någon stil för vår text och ändå ser vi att rubriker och stycken formateras på ett bestämt sätt. I samband med övningarna i XML så presenterades all uppmärkning också i webbläsarfönstret innan någon stil definierats, men när det gäller xhtml och html så finns i webbläsarna en fördefinierad stil definierad för varje html-element. Exempelvis är det meta-element som ligger inom head osynligt och rubriker på nivå 1 har ett något större typsnitt än de på nivå 2.

Inledningen av `div`-elementets innehåll ser ut ungefär som följer, men det finns här några val att kommentera. Jämför först med hur du själv gjort.

```
1
2   <div>
3        <h1>Recognizing Text Genres with Simple Metrics Using Discriminant Analysis </h1>
4
5           <p>Jussi Karlgren jussi@sics.se <br/>
6           Swedish Institute of <br />
7           Computer Science Box 1263, S &mdash; 164 28 Kista, Stockholm, Sweden
8           </p>
9           <p>Douglass Cutting <br />
10          cutting@apple.com  <br />
11          Apple Computer  Cupertino, CA 95014, USA </p>
12
13          <p>October 25, 2005 </p>
14
15
16          <h2> Abstract </h2>
17          <p> A simple method for categorizing texts into pre-determined text genre categories
18          using the statistical standard technique of discriminant analysis is demonstrated
19          with application to the Brown corpus. Discriminant analysis makes it possible use a
20          large number of parameters that may be specific for a certain corpus or information
21          stream, and combine them into a small number of functions, with the parameters
22          weighted on basis of how useful they are for discriminating text genres. An
23          application to information retrieval is discussed. </p>
24
25          <h2>Text Types</h2>
26          <p> There are different types of text. Texts 'about' the same thing may be in
27          differing genres, of different types, and of varying quality. Texts vary along several
28          parameters, all relevant for the general information retrieval problem of matching
29          reader needs and texts. Given this variation, in a text retrieval context the
30          problems are (i) identifying genres, and (ii) choosing criteria to cluster texts of
31          the same genre, with predictable precision and recall. This should not be confused
32          with the issue of identifying topics, and choosing criteria that discriminate one
33          topic from another. Although not orthogonal to genre-dependent variation, the
34          variation that relates directly to content and topic is along other dimensions.
35          Naturally, there is co-variance. Texts about certain topics may only occur in
36          certain genres, and texts in certain genres may only treat certain topics; most
37          topics do, however, occur in several genres, which is what interests us here.
38          </p>
39          <p> Douglas Biber has studied text variation along several parameters, and found
40          that texts can be considered to vary along five dimensions. In his study, he
41          clusters features according to covariance, to find underlying dimensions (1989). We
42          wish to find a method for identifying easily computable parameters that rapidly
43          classify previously unseen texts in general classes and along a small set --  smaller
44          than Biber's five --  of dimensions, such that they can be explained in intuitively
45          simple terms to the user of an information retrieval application. Our aim is to take
46          a set of texts that <em>has</em> been selected by some sort of crude semantic analysis
47          such as is typically performed by an information retrieval system and partition it
48          <em>further</em> by genre or text type, and to display this variation as simply as
49          possible in one or two dimensions. </p>
50  /.../
51
52
53
```

Det första vi ska vara klara över är att det inte finns ett helt korrekt sätt att märka upp en sådan här text på, men att de val vi gör kan ses som mer eller mindre väl valda. Har vi att göra med en större mängd texter av samma typ bör vi utveckla en mer konkret idé med rekommendationer för hur vi hanterar varje typ av innehållselement. Eller kanske hellre välja en annan existerande xml-tillämpning för ändamålet, som t ex JATS eller DocBook.

Här har `h1` valts för artikelns titel och `h2` för avsnittsrubriker. Vidare har `p`, tills vidare, valts för de inledande metadata kring artikeln. Men här kan också ses hur

det tomma elementet `br` (radbrytning) tagits i bruk för att särskilja olika typer av metadata, vilket, inom parentes, kan sägas vara ett tveksamt val. Du bör vara mycket restriktiv med radbrytningar i den här formen. Det finns nästan alltid bättre lösningar, som inte ger sken av att blanda form och innehåll, genom att använda CSS-egenskaper, som du ska se här så småningom.

Du kanske noterar två hittills främmande fenomen här. På rad 7 hittar du `&mdash;` som är ett sätt att ”koda” ett ”em dash” (tankestreck, på svenska, tror jag). Bindestrecket (`-`, hyphen) används i typografin huvudsakligen för avstavningar och språkliga sammansättningar, medan ett längre streck betyder andra saker. På rad 46 och 48 finner du ett element `em` som ska användas för att ge ord eller fraser *emfas*, medan `strong` används för extra stark emfas. Elementen är s k inline-element, vilket betyder att de *inte* genererar en radbrytning före och efter i en webbläsare. Det finns ett flertal sådana inline-element i html-repertoaren som kan vara användbara, t ex `code`, `cite` och `q`.

`span` hör också hit, vilket är en inline-motsvarighet till `div`. Vi ska återkomma till dessa två längre fram och ignorerar dem tills vidare.

En av de mest utmärkande företeelser för webbsidor är ”hyperlänkar” som kan beskrivas som en definierad relation mellan ett område i din text och en annan webbsida (eller ett annat område i texten), där ett område vanligen utgörs av ett stycke text eller en bild. Länkar (som är ett inline-element) deklareras med elementet `a` som är en förkortning av anchor (ankare) och har alltid ett eller flera attribut.

För den här texten som är daterad 1994 har vi inga goda exempel att använda, så vi får improvisera och ge ett ”mervärde” till texten genom att länka något koncept som kan behöva en förklaring. Låt oss ta diskriminantanalys och länka till Wikipedias artikel.

/.../

<h2> Abstract </h2>

 A simple method for categorizing texts into pre-determined text genre categories using the statistical standard technique of
 discriminant analysis
is demonstrated with application to the Brown corpus. Discriminant analysis makes it possible use a large number of parameters that may be specific for a certain corpus or information stream, and combine them into a small number of functions, with the parameters weighted on basis of how useful they are for discriminating text genres. An application to information retrieval is discussed.

/.../

Värdet till attributet `href` (hyperlink reference) är alltså i det här fallet en URL. Prova själv.

Bilder är givetvis en annan väsentlig företeelse i samband med webbsidor. Det element som används är `img` som alltid har minst ett attribut, `src`, vars värde är en URI eller URI referens som identifierar en bildfil.

Vi kan nu länka in figurer och tabeller om vi har dem som digitala filer i lämpligt format. Identifiera platshållaren för figur 2, som du kanske gett följande utformning:

```
<p> Figure 1: Distribution, 2 Categories </p>
```

och utvidga med, efter att du laddat ner bildfilerna från Github-arkivet.
https://github.com/Mikael61/htmlAndCSS
Annars är det möjligt att i stället för filnamnet ange URLen
https://raw.githubusercontent.com/Mikael61/htmlAndCSS/main/figure2.png

```
<p>
<img src="figure2.png" alt="Distribution, 2 categories"/>
Figure 1: Distribution, 2 Categories </p>
```

Bilder lagras alltså separat från html-filen och bäddas in i texten i webbläsaren när

denna läser in html-filen. De är normalt av en av tre olika typer, gif, jpg eller png. `img` är ett tomt element, men måste dock ”stängas” med `/`.

> *Värdet till attributet `alt` skall alltid anges för den händelse användaren inte kan ta del av bildens innehåll.*

Det finns i Github-arkivet bildfiler även för tabell 1 och 2. Om du vill kan du göra samma sak med dem. Inom parentes sagt så finns i html-repertoaren också element för att skapa tabeller. Det är inte svårt men något tidsödande, så vi hoppar över det här. Två andra fenomen som kan vara användbara är onumrerade och numrerade listor, `ul` + `li`, samt `ol` + `li`. Du kan själv hitta anvisningar för hur du hanterar dem, se t ex
`https://www.w3schools.com/html/` men observera att anvisningarna där gäller för html5, inte xhtml. OXYGEN kommer dock att varna dig om du försöker göra något som inte är giltigt.

2 CSS

Så här långt har du varit hänvisad till de default-inställningar för presentation (rendering) av html-element som webbläsarna erbjuder. Det har sannolikt varit frestande att söka html-element som kan ge en mer tilltalande presentation, kanske t ex rendera upphovsuppgifterna centrerat. Allt sådant ska dock hänskjutas till CSS-regler, som är den teknik som erbjuds för hantering av typsnittets storlek, färger, marginaler mm. Här behöver vi inga djupdykningar utan kan nöja oss med relativt enkla medel för att hantera den text vi har att göra med.

Det finns ingen anledning att anstränga sig för att exakt *imitera* formgivningen i den givna pdf-filen, men du kan hämta inspiration från den.

Se till så att du har en version av html-filen öppen i din editor, men observera att du normalt inte ska behöva redigera html-uppmärkningen (vi kommer dock att göra avsteg från det här, eftersom vi inte är helt klara med redovisningen av väsentliga html-element). Utgångspunkten är att uppmärkningen alltid ska vara färdig när man börjar arbeta med CSS, dvs formgivningen. Förbered gärna och öppna samma fil i FIREFOX (du kan även öppna den i t ex CHROME, eftersom detaljer kan skilja sig åt). På så vis så kan du enkelt växla mellan editorn och webbläsaren i takt med att du gör förändringar. Om du ännu inte sett det användbara i tangentkombinationerna Alt+tab PC och Command+tab MAC så vänj dig vid att använda dem för att växla mellan fönster i stället för att stänga eller minimera fönster.

Observera att 1) HTML-koden måste vara helt giltig för att inte formgivningen ska bli oförutsägbar, och 2) att CSS är extremt känslig för fel i notationen/syntaxen. Ett `{` för mycket eller ett `;` för lite kan göra att hela stilmallen upphör att fungera.

De element du använt dig av är antagligen inte särskilt många, men vi skall nu se hur du kan göra för att redigera deras form med hjälp av CSS-regler. Låt oss starta med att styra typsnittet på texten. CSS-egenskapen för detta är `font-family`, och det är alltså den som skall sättas. Det följande är en s k *deklaration* som säger att typsnittet skall vara Verdana, och om det inte finns installerat så skall webbläsaren välja ett annat typsnitt utan serif (sans-serif). Det sista är alltså en s k fallback-åtgärd.

```
font-family : Verdana, sans-serif
```

Emellertid räcker inte detta, utan vi måste också tala om för *vad* den här deklarationen skall gälla. Om det är grundtypsnittet vi vill komma åt, så är det för elementet `body` som den här regeln skall sättas. Alla s k barn (element inuti), barnbarn, barnbarnsbarn osv, till `body` ärver nämligen somliga av sina föräldrars egenskaper.

CSS-specifikationen anger exempelvis för vilka egenskaper arvregeln gäller, det gäller nämligen inte alla egenskaper, marginalbredd t ex. *Kontrollera alltid gentemot denna (kolumn 5)*. Olika webbläsare har också tyvärr olika väl utvecklat stöd för CSS, vilket gör att du bör kontrollera om det fungerar i flera webbläsare.

```
body {
font-family : Verdana, sans-serif
}
```

är nu en fullständig CSS-regel som reviderar det typsnitt som webbläsaren har konfigurerats att rendera all text i. `body` refererar till elementet '`body`' men kallas i kontexten av CSS för en *selektor*. Emellertid återstår att knyta CSS-regeln till HTML-dokumentet. Det finns tre sätt att göra det på: 1) inline-stil, 2) inbäddad stilmall, och 3) extern stilmall. En extern stilmall är alltid att föredra om du avser att ge samma stil åt mer än en fil. En inbäddad stilmall är tillämpligt om vi vet

att vi endast skall åstadkomma en enda HTML-fil med samma stil, eller om vi vill modifiera något i specifikation från en extern stilmall. Inline-stilen är nästan aldrig värt att öda kraft på och är ganska irrationell.

2.1 Inline

Det mindre lämpliga sättet, inline, görs genom att lägga till ett attribut style för det element i dokumentet som man vill ge en särskild stil

```
<body style="font-family : Verdana, sans-serif">
  ...
</body>
```

Notera också att selektorn då inte skall vara med. Prova att göra detta en gång, för att se effekten, men ta sen genast bort det igen. Det är inget bra förfarande.

2.2 Inbäddad

För att bädda in en stilmall måste du skapa (i `head`) ett element `style`, där du sen kan bädda in fler regler.

```
<head>
  ...
  <style type="text/css">
  body {
  font-family : Verdana, sans-serif
  }
   ...
  </style>
</head>
```

Attributet `type` är nödvändigt (i XHTML) och deklarerar vilken typ av stilmallsteknik det är frågan om, med hjälp av en standardiserad MIME-kod, `text/css`.

Prova nu med att göra detta i editorn och se vad som händer i webbläsaren (du måste förstås uppdatera med den nya versionen av filen). Låt det sedan ligga kvar.

2.3 Extern

Det tredje sättet är att tillämpa CSS-regler som ligger lagrade i en (eller flera) separat(a) fil(er).

I det här fallet skapar du en helt ny fil som du döper till, exempelvis *stil.css*. Gör så, samtidigt som du sparar den i samma mapp som din html-fil. Skriv in i denna (och spara)

```
body {
font-family : "Times New Roman", serif
}
```

Nu skall du länka till denna fil med ett element `link`, som är definierat som ett tomt element.

```
<head>
  ...
  <link rel="stylesheet" type="text/css" href="stil.css" />
  <style type="text/css">
  body {
  font-family : Verdana, sans-serif
  }
   ...
  </style>
</head>
```

Med attributet `href` ges filnamnet (eller en URI) till stilmallen. Med attributet `rel` (relation) deklareras att det är en stilmall. Granska nu resultatet. Ser du någon skillnad?

Låt oss nu se på vår artikel för att exemplifiera några andra egenskaper och samtidigt introducera två element från html-repertoaren och ett väsentligt attribut därifrån. Därefter får du själv utforska de relativt komplexa teknikerna för formgivning med CSS.

CSS Snapshot 2018 (`https://www.w3.org/TR/css-2018/`) är en god utgångspunkt, men vill du ha en förteckning av CSS-egenskaper så kan du ta Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification (`https://www.w3.org/TR/C`

som utgångspunkt, särskilt https://www.w3.org/TR/CSS2/propidx.html. W3Schools har annars också mer lättillgänglig dokumentation, men W3C är normativ.

Vi kan konstatera att pdf-källan har satt titeln i vad som brukar kallas ”small-caps”, dvs att bokstäverna alla är omvandlade till versaler men de som är givna som gemener i källtexten renderas med något lägre höjd än de egentliga versalerna.

Vi kan också konstatera att radlängden och textytan är alldeles för bred om vi öppnar den i en webbläsare med god upplösning. Typografiska riktlinjer brukar ge att radlängder över ett visst antal (mellanslag inkluderade) sänker läsbarheten betydligt. En pocketroman har ofta omkring 60 tecken per rad, vilket kan vittna om vartåt vi vill ha radlängden. Vi behöver därför sätta ett mått för ungefär hur många tecken som ska rymmas på en rad, oavsett typsnittets storlek. Måttenheten `em` är bra för ändamålet — `1em` är liktydigt med höjden på grundtypsnittets glyf för bokstaven ’m’.

Här är en tänkbar lösning på dessa två problem.

```
1  /.../
2      <head>
3          <title>Recognizing Text Genres with Simple Metrics Using
4           Discriminant Analysis &mdash; Jussi Karlgren,
5           Douglass Cutting</title>
6          <style type="text/css">
7              body { width : 36em ;
8              margin : 1em 1em 1em 1em ;
9              padding : 1em 1em 1em 1em ;
10             border : solid red 1px ;
11             font-size : 12pt
12             }
13             h1 { font-variant : small-caps;
14             }
15         </style>
16     </head>
17 /.../
```

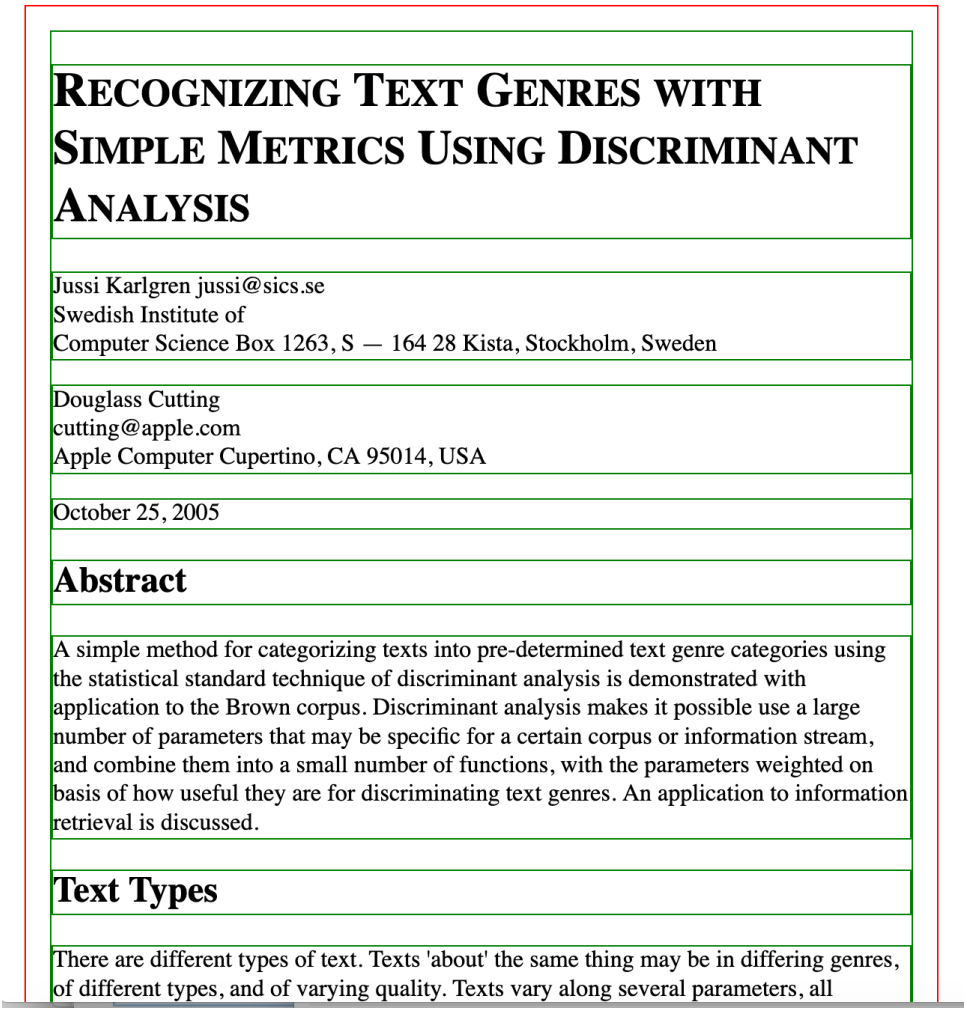
Förutom lösningarna på rad 13 och rad 7 har vi här också illustrerat hur ett block-elements egenskaper `padding`, `border` och `margin` kan sättas (padding är ytan innanför ramen).

Jag vill betona att `border` är här satt för att just illustrera den grafiska yta som ett block-element gör anspråk på. När du arbetar med formgivningen kan det vara

praktiskt att sätta en ram på alla element på följande sätt (rad 14).

```
1  /.../
2      <head>
3          <title>Recognizing Text Genres with Simple Metrics Using
4           Discriminant Analysis &mdash; Jussi Karlgren,
5           Douglass Cutting</title>
6          <style type="text/css">
7              body { width : 36em ;
8              margin : 1em 1em 1em 1em ;
9              padding : 1em 1em 1em 1em ;
10             font-size : 12pt
11             }
12             h1 { font-variant : small-caps;
13             }
14             * { border : solid red 1px ;
15             }
16         </style>
17     </head>
18 /.../
```

* känner du som trunkeringstecken i många databaser, här, som selektor, refererar det till alla förekommande html-element. Regeln kan enkelt raderas när du är färdig. I figuren nedan kan du också observera och dra slutsatsen att default för padding är 0, men att det inte gäller för margin-top och margin-bottom för alla element.



Recognizing Text Genres with Simple Metrics Using Discriminant Analysis

Jussi Karlgren jussi@sics.se
Swedish Institute of
Computer Science Box 1263, S — 164 28 Kista, Stockholm, Sweden

Douglass Cutting
cutting@apple.com
Apple Computer Cupertino, CA 95014, USA

October 25, 2005

Abstract

A simple method for categorizing texts into pre-determined text genre categories using the statistical standard technique of discriminant analysis is demonstrated with application to the Brown corpus. Discriminant analysis makes it possible use a large number of parameters that may be specific for a certain corpus or information stream, and combine them into a small number of functions, with the parameters weighted on basis of how useful they are for discriminating text genres. An application to information retrieval is discussed.

Text Types

There are different types of text. Texts 'about' the same thing may be in differing genres, of different types, and of varying quality. Texts vary along several parameters, all

3 Elementen div och span, samt attributen class och id

I inledningen kunde du se hur vi bäddade in allting i `body` i ett inneliggande `div`. div är ett block-element men saknar annan semantik än att det just skapar ett avsnitt (division) med ett visst innehåll. Elementet `span` är en motsvarighet för inline-element, dvs beter sig som t ex `em` eller `code` beter sig.

Vad ska det vara bra för då? Ska inte uppmärkningen vara semantiskt meningsfull, säga något om innehållets karaktär?

Det är här vi kan lite grann förbigå html-repertoarens begränsningar. Om det inte finns ett html-element för t ex ett 'abstract' så kan du skapa en klass av `div`-element för ändamålet.

```
<div class="abstract"><h2> Abstract </h2>
                <p> A simple method for categorizing texts
into pre-determined text genre categories
using the statistical standard technique of
dscriminant analysis is demonstrated
with application to the Brown corpus.
Discrimanant analysis makes it possible use a
large number of parameters that may be specific
for a certain corpus or information
stream, and combine them into a small number of
functions, with the parameters
weighted on basis of how useful they are for
discriminating text genres. An
application to information retrieval is discussed.
                </p>
</div>
```

eller

```
<h2 class="abstract"> Abstract </h2>
                <p> A simple method for categorizing texts
into pre-determined text genre categories
using the statistical standard technique of
dscriminant analysis is demonstrated
with application to the Brown corpus.
Discrimanant analysis makes it possible use a
large number of parameters that may be specific
for a certain corpus or information
stream, and combine them into a small number of
functions, with the parameters
weighted on basis of how useful they are for
discriminating text genres. An
application to information retrieval is discussed.
                </p>
```

Attributet `class` kan alltså åsättas vilket element som helst och därmed bidra med semantisk tydlighet till innehållet i en text. En sådan klass av element kan sedan hanteras med CSS genom en speciell selektor som inleds med en punkt (se rad 16).

Till exempel

```
/.../
    <head>
        <title>Recognizing Text Genres with Simple Metrics Using
         Discriminant Analysis &mdash; Jussi Karlgren,
         Douglass Cutting</title>
        <style type="text/css">
            body { width : 36em ;
            margin : 1em 1em 1em 1em ;
            padding : 1em 1em 1em 1em ;
            font-size : 12pt
            }
            h1 { font-variant : small-caps;
            }
            * { border : solid red 1px ;
            }
            .abstract { font-weight : bold ;
            text-align : center ;
            }
        </style>
    </head>
/.../
```

Prova nu själv och se effekten med de båda varianterna av lösning för html-kodningen.

När du nu fått grepp om klasser och elementet `div` så kan vi ta hand om och strukturera innehållet direkt efter artikeltiteln som ju innehåller väsentliga metadata som kanske skulle vara värdefulla att algoritmiskt kunna extrahera. Här är en av många möjliga lösningar som också använder `span`.

```
/.../
<h1>Recognizing Text Genres with Simple Metrics
    Using Discriminant Analysis </h1>
    <div class="opening">
      <div class="person" id="p1"><span class="name">Jussi
       Karlgren</span> <span class="e-mail">jussi@sics.se
       </span>
      <span class="adress">Swedish Institute of
       Computer Science Box 1263, S &mdash; 164 28 Kista,
       Stockholm, Sweden </span>
      </div>
      <div class="person" id="p2"><span class="name">Douglass
      Cutting</span>
      <span class="e-mail">cutting@apple.com  </span>
      <span class="adress"> Apple Computer  Cupertino,
      CA 95014, USA </span></div>
      <p class="date">October 25, 2005 </p>
    </div>
/.../
```

Här ser du också exempel på attributet `id` som identifierar exakt en instans av ett element

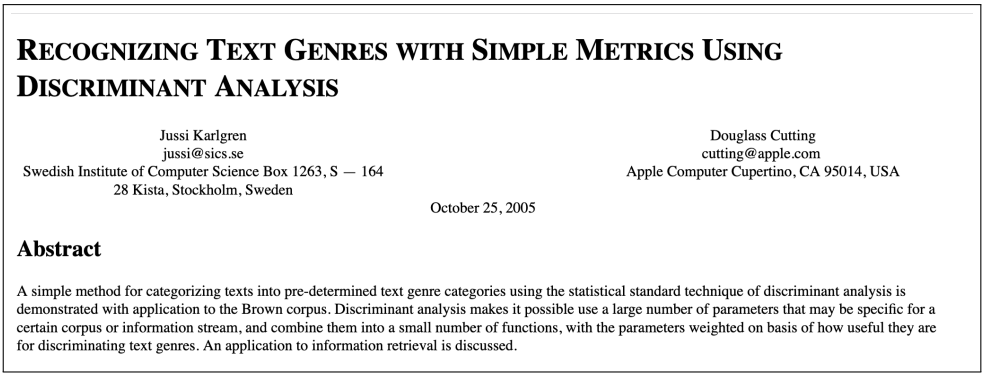
> *NB Här vore det mer på sin plats att göra bruk av Microdata eller RDFa, se t ex Wikipedias artikel på temat:* `https://en.wikipedia.org/wiki/RDFa`*. Men vi låter det vara lite enklare här. Du har alltid möjligheten att fördjupa dig på egen hand, om intresse finns.*

På det här viset kan vi också formge dessa metadata med CSS. Kom dock ihåg att det inte finns någon poäng med att replikera förlagan exakt!

```
/.../
     .opening { text-align : center}
     #p1 {float : left }
     #p2 {float : right }
     .name:after {content: '\A';
        white-space: pre;}
     .e-mail:after {content: '\A';
        white-space: pre;}
     .date {clear : both }
     .person { width : 40% }

/.../
```

med följande resultat



RECOGNIZING TEXT GENRES WITH SIMPLE METRICS USING DISCRIMINANT ANALYSIS

Jussi Karlgren
jussi@sics.se
Swedish Institute of Computer Science Box 1263, S — 164
28 Kista, Stockholm, Sweden

Douglass Cutting
cutting@apple.com
Apple Computer Cupertino, CA 95014, USA

October 25, 2005

Abstract

A simple method for categorizing texts into pre-determined text genre categories using the statistical standard technique of discriminant analysis is demonstrated with application to the Brown corpus. Discriminant analysis makes it possible use a large number of parameters that may be specific for a certain corpus or information stream, and combine them into a small number of functions, with the parameters weighted on basis of how useful they are for discriminating text genres. An application to information retrieval is discussed.

Selektorn `#p1` refererar alltså till elementet med `id="p1"`.

`.name:after` gör bruk av en pseudoselektor och avser efterföljande utrymme. Det används ofta för att infoga text med CSS-egenskapen `content` som här har en radbrytning efter elementet, vilket då sin tur fordrar att egenskapen `white-space` ges värde `pre`.

`float` gör att elementet i fråga inte längre beter sig som ett vanligt block-element utan flyter, antingen till vänster eller höger och kan därmed låta innehåll renderas till höger eller vänster.

Här var det många komplexa detaljer. Observera att du får se det som exempel och behöver inte lära dig CSS på den nivån inom ramen för den här kursen.