DD2418 Språkteknologi, KTH
Johan Boye

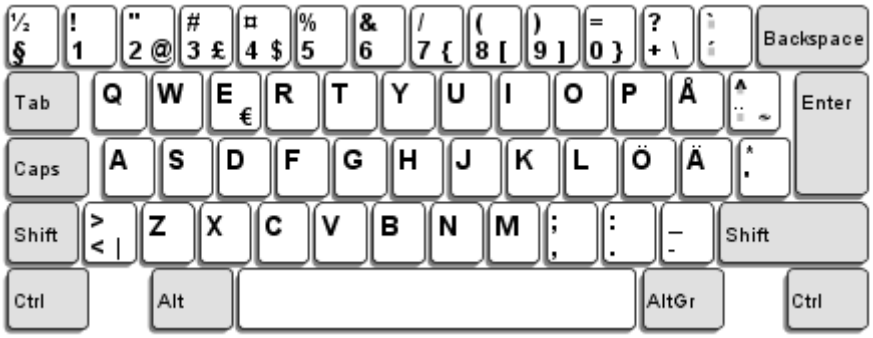
Laboration 5: Dolda markovmodeller

Uppgift 0: *Läs om markovmodeller och Viterbi-algoritmen (boken sid 207-220).*

När man skriver på ett tangentbord finns alltid en viss risk att man råkar trycka på fel tangent, t.ex. att man råkar skriva ett "H" i stället för ett "G". I denna uppgift ska du försöka rätta sådana fel med hjälp av en s.k. *dold markovmodell* (eng. "Hidden Markov Model", eller HMM).

Sannolikheten för att en viss (felaktig) tangent trycks ned i stället för en annan (korrekt) tangent är inte likformigt fördelad, utan det är vanligare att man förväxlar intilliggande tangenter än tangenter om ligger långt ifrån varandra på tangentbordet. I denna labb gör vi följande förenklade antagande: **Sannolikheten att trycka ned en felaktig tangent är 0.1 om den felaktiga tangenten ligger intill den korrekta och 0 annars.** Om man till exempel menade att trycka ned ett "A" så kan "Q", "W", "S" eller "Z" produceras i stället med sannolikhet 0.1 vardera. Sannolikheten att man verkligen lyckas träffa "A" när man avser att trycka på "A" blir då 1-4·0.1= 0.6.

Som en ytterligare förenkling kommer vi bortse från alla icke-bokstäver på tangentbordet (vi låtsas att de inte finns). Enda undantaget är mellanslagtangenten, som vi antar att vi alltid lyckas träffa med sannolikhet 1, och dessutom att mellanslagstangenten aldrig trycks ned av misstag.



Det "random keyboard" vi ska ägna oss åt i denna labb är en implementation av en sådan feltryckningsmodell. Detta speciella tangentbord lägger in slumpmässiga fel i texten enligt sannolikhetsfördelningen som just beskrivits.

Gör detta först: Kopiera alla filer på kurskatalogen `/info/DD2418/sprakt14/HMMlab/` till en lokal katalog under din hemkatalog eller på din egen dator. Katalogen ska innehålla 3 st Java-klasser och två filer med n-gram-statistik.

Titta i filen `bigramstats.txt`. Filen innehåller bokstavsbigramstatistik för svenska språket. En rad i filen ser typiskt ut så här:

```
0 0 -7.71
```

vilket ska tolkas som att (naturliga) logaritmen av den uppskattade betingade sannolikheten P("aa"|”a”) är -7.71. Filen innehåller sådan bigramstatistik för alla bokstäver i engelska språket (a-z, kod 0-25), samt de svenska bokstäverna "ö" (26), "ä" (27) och "å" (28), samt mellanslag/ meningsavskiljare (29). Raden ”`29 0 -2.71`” uttrycker alltså den betingade sannolikheten för bigrammet ”mellanslag/meningsbörjan följt av ’a’”, medan ”`0 29 -1.48`” uttrycker den betingade sannolikheten för bigrammet ” ’a’ följt av mellanslag/meningsslut”.

Uppgift 1: *Förklara för labbassistenten varför det är lämpligt att logaritmera sannolikheterna, snarare än att representera dem som decimaltal mellan 0 och 1.*

Kompilera alla Java-klasser med

```
javac *.java
```

och kör sedan programmet med

```
java TypeWriter
```

Om allt fungerar kommer du se ett fönster med 3 st textfält. I det översta skriver du in en mening (**OBS! använd bara de gemena (”små”) bokstäverna a-ö samt mellanslag**). Allt eftersom du skriver kommer den förvrängda texten produceras i mellanfältet. När du skrivit klart, tryck på "Decode". Då visas avkodningen av den förvrängda texten i det tredje fältet. Just nu kommer denna sträng enbart bestå av en mängd "a". Detta ska vi snart avhjälpa.

Uppgift 2: *Förklara för labbassistenten hur feltryckningsmodellen som beskrivits ovan kan representeras som en HMM. Beskriv speciellt vilka tillstånden är, vilka observationerna är, och hur man kan definiera sannolikhetsmatriserna A (för tillståndsövergångar) och B (för observationer).*

Uppgift 3: *Om vi ser strängen "qööq", vilket svenskt ord har antagligen skrivits in, givet feltryckningsmodellen och bigramsstatistiken för svenska som finns i filen "bigramstats.txt"? Rita upp grafen (trellis) för strängen "qööq" och förklara.*

Uppgift 4: *Komplettera Java-koden i klassen* `Decoder.java` *(leta efter "YOUR CODE HERE"), så att klassen korrekt implementerar Viterbi-algoritmen. Tips: Implementationen håller sig nära pseudokoden i boken (s 220). En skillnad är att boken räknar sina tidssteg från 1 och uppåt, medan Java-programmet räknar från 0.*

Om din implementation är korrekt så bör resultatet i det nedersta textfältet när du trycker "Decode" nästan alltid bli bättre än den förvängda texten i mellanfältet.

Här är några testmeningar du kan skriva in i **mellanfältet** och trycka "Decode":

Testmening	**Ska bli**	**Korrekt mening**
bkmbwf lxy hfanatef	bomber och branater	bomber och granater
wjzmnz qrtoilsr dpe zwjqtw oädnint	skanna artlilar för senate lärnint	skanna artiklar för senare läsning
tlitwns lwmpq öhded	flitens lampa änder	flitens lampa lyser