Lösningsförslag till tentamen i DH2418 Språkteknologi 2011-10-19

1. **Lexikal flertydighet**: ”Amazon” kan vara en flod eller en webbplats. Detta kan ställa till problem för sökmotorer (svårt att få bra precision). **Syftningsflertydighet** kan till exempel ställa till det för översättningsprogram; ”He ate his ice-cream” kan översättas till ”Han åt sin glass” eller ”Han åt hans glass” beroende på sammanhanget.

2. Täckning: andel fel som upptäcks. Precision: andel av larm som är riktiga fel. Täckning och precision kan uppskattas med hjälp av en felkorpus, där alla fel är kända från början. Språkteknologen bör alltså börja med att anskaffa eller konstruera en felkorpus.

3. Till exempel: (1) Om ordet kan bildas som en sammansättning där efterledet finns i tagglexikonet används efterledets taggning. (2) Använd statistik för hur ord med samma suffix som det okända taggas. (3) Tagga ordet som egennamn om det börjar med versal.

4. En vektorrumsmodell avbildar sökfrågan och varje dokument som vektorer, och mäter likhet med hjälp av vinklar i detta vektorrum. Fördelen med detta är att sökmotorn kan hitta dokument som inte innehåller alla dokument i sökfrågan, att dokumenten kan rankas efter vinkellikhet, samt att det inte krävs någon utbildning för att kunna använda sökmotorn. En Boolesk modell ser sökfrågan som ett Booleskt uttryck (t.ex. ord x och y ska finnas i dokumentet, men inte z). Fördelen med en Boolesk modell är att det är matematiskt exakt definierat vilka dokument som bör returneras för en given sökfråga.

5. Blandat initiativ innebär i enklaste fallet att systemet ställer frågor och användaren svarar, men att användaren kan ge mer information än vad det frågas efter. Till exempel: ”Vart vill du åka?” ”Jag vill åka från Stockholm till Göteborg på torsdag morgon”. Detta kan åstadkommas med en formulärbaserad dialoghantering.

6. Först måste uppsättningen kategorier bestämmas, i samråd med experter på Skatteverket. Eftersom det antagligen är mycket svårt att försöka göra en grammatisk analys av innehållet i godtyckliga e-brev, skulle en realistisk approach vara att bygga en kategoriserare enbart baserat på vilka ord som förekommer i brevet (en *”bag-of-word approach”*) . Naive Bayes brukar vara en metod som funkar bra på dylika problem. Innan e-breven kan kategoriseras måste de dock tokeniseras, stavfelen rättas, och helst bör alla ord lemmatiseras. Man kan utvärdera systemet genom att kolla hur många e-brev som kategoriseras rätt, och jämföra med hur experter på Skatteverket lyckas utföra samma uppgift. För att systemet ska accepteras får det inte vara mycket sämre (och helst bättre) än mänskliga experter.

7. (a) Ta fram statistik för bokstavsfyrgram, där ordbörjan/ordslut behandlas som en egen bokstav. I den tredje ordningens markovmodellen har vi ett tillstånd för varje bokstavstrigram (som förekommer). Generera tänkbara korrigeringar på vanligt sätt och se vilken produktsannolikhet varje korrigering får i markovmodellen (eventuellt med hänsyn taget till ordlängden). Rangordna förslagen efter produktsannolikheterna och använd en tröskel för vilka förslag som ska presenteras.

(b) Eftersom rättelseförslagen inte filtreras genom en ordlista är risken att ord som inte är äkta ord föreslås. Detta sänker systemets trovärdighet eftersom användare troligen förväntar sig att ett rättstavningssystem ska kunna stava.

8. (a) Sture har övertränat kategoriseraren och testat den på träningsmaterialet, vilket har gett orealistiskt bra resultat. Han har inte tagit med några exempel i klassen **Någon Annan**, vilket leder till att denna klass aldrig kommer uppträda. Dessutom verkar han inte ha klargjort för sin kund vad produkten har för användningsområde.
   (b) Slumpen är en olämplig baslinje i detta fall. En kategoriserare som alltid chansar på den största klassen (CL) skulle i detta fall ha rätt i nästan 47 % av fallen (7/15) om träningsmaterialet är representativ t för testmängden. En lämplig baslinje i detta fall kan alltså vara 47 %, och en kategoriserare som presterar sämre måste anses dålig. En ännu bättre baslinje kanske kan fås genom att kolla hur bra mänskliga experter skulle prestera på samma problem.

9. För att automatisera uppgiften måste man ha tillgång till samma text på två olika språk, t.ex. ett antal böcker på svenska och dess engelska översättningar (detta brukar kallas en ”parallellkorpus”). Först så paras varje svensk mening ihop med sin översättning. En heuristik man kan använda är meningslängd: En lång mening översätts ofta till en annan lång mening. Detta innebär att om man utgår från en lång svensk mening, och ser en lång engelsk mening ungefär på rätt ställe, så kan man anta att denna är en översättning av den svenska långa meningen. När man har parat ihop varje svensk mening med en engelsk (och slängt de meningar man inte kan para ihop), försöker man para ihop ord i de bägge meningarna. Detta kan göras med hjälp av ett lexikon och en morfologikomponent för att analysera böjda ord. Om man lyckas para ihop alla svenska ord utom ett (t.ex. ”fett”) med engelska ord, och bara ett engelskt ord återstår (t.ex. ”cool”), så är dessa kandidater för en ny översättningsvariant.

10. Vi måste utgå från ett lexikon där vissa ord är märkta med en smiley som markerar emotiv laddning: t.ex. ”lycklig” är märkt ☺, medan ”olycklig” är märkt ☹. Eftersom programmet ska returnera ”?” om inputmeningen är ogrammatisk kan vi knappast använda en rent statistisk approach. Det bästa verkar vara att skriva en grammatik, där varje grammatikregel associeras med en formel (inom mängdklamrar nedan) för hur man räknar ut rätt smiley. Vi antar nedan att vi kan negera smileys, så att –☹ = ☺ och –☺ = ☹ och –😐=😐. Några exempelregler:

ADJP → JJ	{smiley(AdjP) = smiley(JJ)}
ADJP → inte ADJP	{smiley($AdjP_1$) = −smiley($AdjP_2$)}
ADJP → knappast ADJP	{smiley($AdjP_1$) = −smiley($AdjP_2$)}
NP → PN	{smiley(NP) = smiley(PN)}
VP → V	{smiley(VP) = smiley(V)}
PN → han	{😐}
V → blev	{😐}
JJ → sjuk	{☹}
S → NP VP ADJP	{smiley(S) = smiley(ADJP)}

Varje nod i syntaxträdet kan nu associeras med en smiley som räknas ut enligt formeln, så här: