setningsalignment med TCA2

jeg skal fortelle om et setningsalignment-program jeg har laget - TCA2.

hva er et setningsalignment-program?

et setningsalignment-program tar som input to filer, inneholdende hver sin tekst, skrevet på hvert sitt språk. den ene teksten er en oversettelse av den andre. programmet prøver å finne ut hvilke setninger i den ene teksten som er oversettelse av hvilke setninger i den andre. output fra programmet er igjen to filer - med hver sin tekst, der informasjon om hvordan setningene hører sammen er lagt inn.

ord-alignment

tekst kan også alignes på ord-nivå. sindre sørensen har laget et ord-alignment-program. men våre to programmer har ikke noe med hverandre å gjøre.

hva kan output brukes til?

...

bakgrunn, tilblivelse, ...

programmet er en videreføring av et program som knut hofland har skrevet. men mens hans program er et batch-program for automatisk alignment, skulle det nye programmet være interaktivt. med mitt program kan brukeren gjøre alignment dels automatisk, dels manuelt. brukeren kan f.eks kjøre automatisk et lite antall alignments om gangen, for så å sjekke og rette opp manuelt.

det var johan poppe som opprinnelig fikk jobben med å lage det nye programmet. han ble ikke ferdig, men la et solid fundament for min jobb. jeg har videreført programmet til en stand hvor det kan brukes. men det er fortsatt rom for forbedringer.

knuts program heter TCA, og mitt er døpt TCA2.

programmet er utviklet i java. det kjøres lokalt. siden det er skrevet i java, er det kanskje ikke så vanskelig å lage en versjon som går på web, men det spørs om det er noe særlig poeng.

hovedtrekkene i grensesnittet

STARTER PROG

programmet har et grensesnitt med en venstre-del, en midt-del og en høyre-del. til venstre kan brukeren se den ene teksten, til høyre den andre teksten.

grensesnittet er tredelt også andre veien. hver tekst viser over tre ruter. nederst står setninger som ennå ikke er alignet.

ÅPNER FILER

ÅPNER OGSÅ ANKERORDLISTE (men sier bare et dette er et hjelpemiddel jeg skal komme tilbake til)

når brukeren åpner de to input-filene, er det der teksten vil komme til syne. øverst står setninger som er ferdig alignet. i midten står setninger som er under arbeid, så å si, f.eks et alignment-forslag fra programmet som brukeren gjør endringer i. så tekstenes setninger vandrer nedenfra og oppover mens programmet og brukeren jobber med dem.

KJØRER AUTOMATISK ALIGNMENT (Suggest, men ikke Skip 1-1)

grensesnittet inneholder også endel andre elementer, slik som knapper som får ulike ting til å skje, og paneler som viser relevant informasjon. disse elementene har jeg prøvd å plassere de steder hvor de logisk eller tematisk hører hjemme, men i et par tilfeller var ikke det så lett.

settingsdialog.

xml-kodet tekst

for at programmet skal kunne gjøre jobben sin, må inputfilene ha koder som forteller hvor de enkelte setninger starter og slutter. nærmere bestemt må input være velformet xml, og setningselementene må ha id-attributt med en unik verdi.

setningselementene vil typisk ha koder **<s>**, men brukeren kan selv fortelle hvilken kode - eller hvilke koder - som er brukt for å markere setninger.

automatisk alignment

når brukeren ber programmet om å aligne automatisk, kan hun enten be om å få én alignment, eller at programmet fortsetter av seg selv, og gjør mange alignments, inntil visse betingelser er oppnådd.

i begge tilfeller benytter programmet en helt bestemt framgangsmåte for hver enkelt alignment.

metode for én enkelt automatisk alignment

metoden programmet bruker for å gjøre én alignment automatisk har diverse småingredienser, men hovedprinsippet er som følger:

programmet antar at følgende alignments er mulige:

1-1	1 setning i den første teksten samsvarer med 1 setning i den andre teksten
1-2	1 setning i den første teksten samsvarer med 2 påfølgende setninger i den andre teksten
2-1	2 påfølgende setninger i den første teksten samsvarer med 1 setning i den andre teksten
1-0	1 setning i den første teksten står alene, og har ikke noe samsvar med setninger i den andre teksten
0-1	1 setning i den andre teksten står alene, og har ikke noe samsvar med setninger i den første teksten

programmet prøver alle disse 5 alternativene etter tur, og gir poeng til hvert av dem. poengene kommer fra diverse metoder som ser på ord og lengder. disse metodene kommer jeg tilbake til. det vesentlige nå er at programmet har en måte å gi poeng til en mulig alignment.

men programmet utroper ikke umiddelbart som vinner det alternativet blant de 5 som har flest poeng. isteden prøver programmet å aligne seg videre fram i tekstene. for hvert av de 5 alternativene prøver programmet alle de 5 alternativene for neste alignment, og for hvert av disse 5x5=25 kombinasjonene prøver programmet alle de 5 alternativene for en tredje alignment, osv. programmet minner her om et sjakkprogram som prøver alle mulige kombinasjoner av trekk framover, med den hensikt å finne hvilket første-trekk som er det beste.

alignmentprogrammet utforsker på denne måten et stort tre av muligheter, og for hver sti i treet samler programmet opp poeng fra enkelt-stegene.

og akkurat som et godt sjakkprogram kan luke bort lite lovende stier, så kan alignmentprogrammet luke bort stier som ikke har sjansen til å vinne, og dermed hindre en eksplosiv økning av kombinasjoner.

til sist kårer programmet en vinner - den stien som har flest poeng - og velger det første steget (alignmenten) fra denne.

programmet utforsker altså muligheter langt framover i teksten, og det kan høres rart ut at det alltid bare plukker ut det første steget. dersom programmet likevel er blitt bedt om å gjøre flere alignments - burde det da ikke bestemme flere alignments med det samme? kanskje, men programmet tar faktisk vare på masse mellomresultater som det gjenbruker senere, så det regner ikke så mye på nytt som man kunne tro.

ord- og lengde-baserte metoder for å gi poeng til mulig alignment

underveis i prosessen med automatisk alignment sitter programmet mange ganger og skal gi poeng til enkeltalignments. det sitter altså med n setninger fra første tekst, og skal finne ut hvor godt disse samsvarer med m setninger fra andre tekst.

0-1 eller 1-0 gir alltid 0 poeng.

for de andre kombinasjonene har programmet først tre ordbaserte metoder som det bruker.

den første ordbaserte metoden går ut på å finne ord i de to tekstene som er oversettelser av hverandre, såkalte ankerord. programmet benytter da en såkalt ankerordliste - en liste over par av ord i de to språkene. metoden krever at noen på forhånd har satt opp en slik liste. programmet mitt tåler fraser i tillegg til enkeltord, og ord med wildcard, for å kunne ta bøyninger og sammensetninger.

den andre ordbaserte metoden ser etter ord med stor forbokstav, under den antakelse at slike ord er egennavn, og ser om ordene finnes i begge tekstene. mange egennavn vil være identiske i oversettelse.

den tredje og siste ordbaserte metoden ser etter ord som har liknende skrivemåte. mange ord er mer eller mindre internasjonale, med forholdsvis små forskjeller i skrivemåte fra språk til

språk. her er det mange tenkelige metoder, men i programmet benyttes et mål for likhet kalt Dice, hvor det telles opp identiske bokstavpar.

etter at programmet har gitt poeng ut fra disse ordbaserte metodene, ser den på tegnlengden av de to tekstbitene den sammenlikner. programmet øker poengsummen hvis lengden passer godt overens, og reduserer den hvis samsvaret er dårlig.

som en tilleggsregel krever programmet at alignments av type 1-2 og 2-1 ikke har for dårlig samsvar i lengde. dersom lengden stemmer, gir programmet så mye minuspoeng at den aktuelle stien falle ut av konkurransen.