Syntaktisk informert frasesamanstilling

Kevin Brubeck Unhammer

01/05, 2010

Innhald

List of Corrections

Kapittel 1

Introduksjon

Kapittel 2

Bakgrunn og relaterte metodar

Kapittel 3

Den ideelle frasesamanstillinga

Kapittel 4

Implementasjonen av `lfgalign`

For å finne ut av kor godt krava i forrige kapittel fungerer til å avgrense kva for lenkjer som er moglege, har eg implementert dei etter beste evne i eit Lisp[1]-program.

intro TODO, kanskje noko om kva eg faktisk har fått ut av implementasjonen

Ei implementering gjer det svært synleg om det finst manglar i eit formelt krav, eller om noko ikkje er godt nok spesifisert.

Programmet `lfgalign`[2] tek inn LFG-analysane av to setningar som me av uavhengige grunnar trur er omsetjingar av kvarandre. LFG-analysane må vere disambiguerte og i Prolog-formatet frå XLE[3]. Programmet les inn dei to filene og opprettar ein intern representasjon av LFG-analysen.

Me kan i tillegg gi programmet informasjon om kva for ord-omsetjingar me ser på som lingvistisk prediktable. Intensjonen er at dette kan vere informert av omsetjingstabellen frå eit automatisk ordsamanstillingsprogram, eller av handskrivne omsetjingsordbøker.

treng eg ein eigen del om LPT i dette kapittelet? Implementasjonen er jo veldig enkel iallfall.

Programmet byrjar lenkjinga med f-strukturane. Ei f-struktur*samanstilling* er ei mengd med *lenkjer* mellom individuelle f-strukturar. Resultatet av lenkjinga på dette nivået kan vere tvitydig: sidan det ofte finst fleire måtar å lenkje argument og adjunkt på, får me i første omgang mange samanstillingar mellom kjelde- og mål-f-strukturar.

Difor rangerer me f-struktursamanstillingane, og den beste sender me vidare til c-struktursamanstillinga. Denne delen av programmet gir ut éi, utvitydig mengd med mange-til-mange-lenkjer mellom c-strukturane (her treng me ingen rangering). Nodane i kvar av desse mange-til-mange-lenkjene definerer no den endelege frasesamanstillinga.

[1] Dette språkvalet kan gjere eventuell integrering med andre LFG-system lettare (Common Lisp er m.a. nytta i LFG Parsebanker (Rosén et al., 2009)).

[2] Tilgjengeleg frå http://github.com/unhammer/lfgalign som fri og open programvare under GNU General Public License.

[3] Formatet er dokumentert på http://www2.parc.com/isl/groups/nltt/xle/doc/xle.html. Importeringa til Lisp-strukturar handterer «pakka representasjonar» og kjenner igjen ekvivalensforhold (t.d. der fleire ϕ-variablar refererer til same f-struktur, eller fleire Prolog-variabler refererer til same analyseval); men filene eg har testa utnyttar ikkje det fulle spennet til formatet, så det finst ganske sikkert feil.

Nedanfor går eg gjennom detaljane rundt dei relevante delene av programmet.

4.1 Lenkjer mellom f-strukturar

Hovudalgoritmen for lenkjing mellom f-strukturar er vist i kodefigur 1. Funksjonen `f-align` returnerer ei mengd med moglege samanstillingar. Kvar samanstilling er ei mengd med par av f-strukturar[4]. Eit par (F_s, F_t) representerer ei lenkje frå ein f-struktur på kjeldespråket, til ein f-struktur på målspråket. Me føreset at dette paret har LPT-korrespondanse[5], dette blir sjekka før alle kall på `f-align`. Der me ikkje har informasjon om LPT-korrespondanse mellom to ord (orda er ukjende), er lenkjing lov. Pro-element og substantiv kan alltid lenkjast med kvarandre.

Hjelpefunksjonen `argalign` (som igjen kallar `argalign-p`, vist i kodefigur 2) gir alle moglege «argumentpermutasjonar», dvs. moglege kombinasjonar av lenkjer mellom argumenta til F_s og F_t som tilfredsstiller kravet om LPT-korrespondanse, men utan å sjekke at desse argumenta igjen kan samanstillast. Funksjonen prøver å lenkje kvart argument til eit argument eller eit adjunkt, men gir ingen lenkjer mellom to adjunkt (sjå del 4.1.1 nedanfor om dette). Funksjonen gir heller ikkje kombinasjonar der minst eitt argument ikkje er lenkja – alle kombinasjonane må inkludere alle argument frå F_s og F_t, jf. krav (iii) og (iv) i Dyvik et al. (2009, s. 75). Elles er krav (i) er tautologisk oppfylt, medan me som nemnt føreset at krav (ii) er oppfylt før alle kall på `f-align`.

Eit døme: viss F_s har argumenta SUBJ og OBJ og ingen adjunkt, og F_t har argumentet SUBJ og eitt adjunkt ADJ, der alle ord-omsetjingar er moglege, vil `argalign` gi dei to samanstillingane $\{(\text{SUBJ}, \text{SUBJ}), (\text{OBJ}, \text{ADJ})\}$ og $\{(\text{SUBJ}, \text{ADJ}), (\text{OBJ}, \text{SUBJ})\}$. Viss adjunktet til F_t ikkje fantest, eller ikkje hadde LPT-korrespondanse med nokon av argumenta til F_s, ville me ikkje fått nokon samanstillingar; medan viss paret $(\text{SUBJ}, \text{SUBJ})$ ikkje hadde LPT-korrespondanse og alt anna var likt, ville me berre fått den siste samanstillinga.

Funksjonen `f-align` går så gjennom kvar lenkje i kvar argumentpermutasjon, og prøver å kalle `f-align` på alle lenkjene. Sidan lenkjene som `argalign` gir har LPT-korrespondanse, vil alle f-strukturane i dei rekursive kalla i `f-align` ha LPT-korrespondanse. Eit rekursivt kall kan gi nye samanstillingar i dei indre f-strukturane, viss dei relevante krava er oppfylte.

Det er mogleg at ei lenkje frå éi samanstilling kan finnast i andre samanstillingar, me unngår dobbeltarbeid ved å lagre alle delvise samanstillingar i tabellen *aligntable*. Dette føreset at `f-align`(s,t) er uavhengig av konteksten rundt; t.d. må mengda av samanstillingar som kjem ved å lenkje subjektet til F_s mot subjektet

[4] Eigentleg eit slags avgjerdstre; kvart element er eit par, kor første element er lenkja mellom dei yttarste f-strukturane, og andre element er dei moglege samanstillingane for dei indre strukturane. Denne strukturen kan vere nyttig for å rangere samanstillingar, og `f-align` blir mykje meir oversiktleg av å jobbe med eit slikt tre. Ein funksjon `flatten` omformar det ferdige treet til ei enkel liste med samanstillingar, kor kvar samanstilling er ei flat liste med lenkjer mellom f-strukturar.

[5] Når eg her skriv at to f-strukturar har LPT-korrespondanse, meiner eg sjølvsagt at ordformen til PRED-verdien til kvar f-struktur har LPT-korrespondanse.

til F_t vere uavhengig av om objektet til F_s er lenkja mot eit objekt eller eit adjunkt osb. av F_t.

TODO: nemne føresetnaden om uavhengnad i kapittel 3

```
alignments ← ∅ ;
forall the argperm in argalign(F_s, F_t) do
    p ← ∅ ;
    forall the A_s, A_t in argperm do
        if not(aligntable[A_s,A_t]) then
            aligntable[A_s,A_t] ← f-align(A_s, A_t);
        if aligntable[A_s,A_t] then
            add aligntable[A_s,A_t] to p;
        else
            add (A_s,A_t) to p
    end
    add p to alignments ;
    forall the adjperm in adjalign(argperm, F_s, F_t) do
        a ← copy-of(p) ;                    // optional adjunct links
        forall the A_s, A_t in adjperm do
            if not(aligntable[A_s,A_t]) then
                aligntable[A_s,A_t] ← f-align(A_s, A_t);
            if aligntable[A_s,A_t] then
                add aligntable[A_s,A_t] to a;
            else
                add (A_s,A_t) to a
        end
        add a to alignments ;
    end
end
 // loop through adjalign if no arguments exist
if alignments = ∅ then return ∅ ;  // Fail
else return ((F_s,F_t),alignments) ;
```

Funksjon 1: f-align(F_s, F_t)

Sjølv om det er krav om LPT-korrespondanse mellom kvart argument og eit argument/adjunkt for å lenkje F_s og F_t, er det ikkje noko krav om at alle para i ein argumentpermutasjon tilfredsstiller alle lenkjingskrava. Viss `f-align`(OBJ, ADJ) frå dømet over gir null, og ikkje kan lenkjast (t.d. fordi ADJ hadde eitt argument, og OBJ ingen argument/adjunkt), medan `f-align`(SUBJ, SUBJ) kan lenkjast, vil `f-align` likevel returnere samanstillinga som inneheld (OBJ, ADJ) og (SUBJ, SUBJ). Me kan sjå i *aligntable* for å finne ut av om kvar av f-strukturane kunne lenkjast; i dette tilfellet vil *aligntable*[OBJ, ADJ] vere tom.

Om me i tillegg krev at substrukturar kan samanstillast kan me utelukke len-

TODO: forskjellen mellom LPT-krav og rekursjons-krav på argument må inn i kapittel 3

usage: Kalt av argalign slik:
argalign-p(arguments(F_s), adjuncts(F_s), arguments(F_t), adjuncts(F_t))

$a \leftarrow \emptyset$;
if $args_s$ **then**
  $s \in args_s$;
  forall the $t \in args_t$ ***where*** *LPT(s,t)* **do**
    forall the $p \in$ *argalign-p*($args_s - \{s\}$, $adjs_s$, $args_t - \{t\}$,$adjs_t$) **do**
    add $\{(s,t)\} \bigcup p$ to a;
  end
  forall the $t \in adjs_t$ ***where*** *LPT(s,t)* **do**
    forall the $p \in$ *argalign-p*($args_s - \{s\}$, $adjs_s$, $args_t$,$adjs_t - \{t\}$) **do**
    add $\{(s,t)\} \bigcup p$ to a;
  end
  return a;
else if $args_t$ **then**
  if $adjs_s$ **then**
    $s \in adjs_s$;
    forall the $t \in args_t$ ***where*** *LPT(s,t)* **do**
      forall the $p \in$ *argalign-p*($args_s$, $adjs_s - \{s\}$, $args_t - \{t\}$,$adjs_t$) **do** add $\{(s,t)\} \bigcup p$ to a;
    end
    return a;
  else
    return $\emptyset$; `// Fail`
else
  return $\{\emptyset\}$; `// End`

Funksjon 2: argalign-p($args_s$, $adjs_s$, $args_t$, $adjs_t$)

kjing av f-strukturane F_s og F_t i (1) under:

(1) a. F_s [PRED '**planlegge**<*eg*,[1:*gi*]>' XCOMP $_1$[PRED '*gi (opp)*']]

b. F_t [PRED '**plan**<*I*,[2:*give*]>' XCOMP $_2$[PRED **give**<*I*,*him*,*it*>']]

Men det kan vere at me ikkje *vil* krevje dette i alle moglege tilfelle. Ei tryggare løysing er å rangere ulike løysingar i etterkant, ved å spørje etter dei argumentsamanstillingane som har flest medlem i *aligntable*, dette kjem eg tilbake til i 4.2 nedanfor.

4.1.1 Overflødige adverbial

Argumentpermutasjonane frå `argalign` prøver som nemnt ikkje reine adjunkt-adjunkt-lenkjer, sidan me ikkje vil forkaste lenkjing av F_s og F_t berre på grunn av at ikkje alle adjunkt kunne lenkjast. Men når me har prøvd ein argumentpermutasjon, kan me lage ein kopi av denne som i tillegg inneheld lenkjer mellom «overflødige» adverbial, altså dei adjunkt-adjunkt-lenkjene som `argalign` ikkje prøver. Hjelpefunksjonen `adjalign` (ikkje vist her TODO: implementere :->) konstruerer moglege permutasjonar av lenkjer mellom adjunkt som ikkje er inkludert i *argperm*, og `f-align` prøver desse rekursivt på same måte som med argumentlenkjene. Lenkjene blir lagt til ein *kopi* av argumentpermutasjonane, sidan det ikkje er sikkert at me ønskjer å lenkje alle adjunktdøtre. Viss me har to overflødige adjunkt på kvar side, og kravet om LPT-korrespondanse er dekkja for alle fire moglege par, får me seks moglege permutasjonar, sidan me inkluderer dei fire permutasjonane der eitt adjunktpar er ulenkja.

og så er det spørsmålet om me kan lenkje adjunkt på ulike nivå i f-strukturane

Viss F_s og F_t ikkje hadde argument i det heile teke, går me au gjennom moglege permutasjonar av adjunktdøtre, på same måte (ikkje vist i kodefigur 1).

4.1.2 SKRIV Når f-lenkjene ikkje er 1-1

kausativ

preposisjonsobjekt

Dette må 1. spesifiserast (kap.3), og 2. implementerast...

"sigaretten" og "sigaretze" er ikkje på same nivå i dei respektive f-strukturane nedanfor:

```
0[ PRED vedde<28,29,27,30>
   29[ PRED sigarett<> ] ]

0[ PRED da-najleveba<37,10,46>
   ADJUNCT { 2 }
```

```
2[ ze<5>
   OBJ 5[ sigareti ] ] ]
```

Sjå au del **??**.

notat :ROTETE:

filene

```
((tab_s (open-and-import "dev/TEST_argadj_s.pl"))
 (tab_t (open-and-import "dev/TEST_argadj_t.pl")))
```

viser at me kan trenge samanføying av pred på ulike nivå.

4.1.3 Kan me gjere f-struktursamanstillinga bottom-up?

Ein alternativ metode for lenkjing av f-strukturane er å byrje med alle logisk moglege permutasjonar av LPT-korrespondansar, og så sile ut dei som ikkje svarer til krava. Prosessen ville nok blitt mykje meir oversiktleg på denne måten, sidan det då berre er snakk om å sjekke krav for kvar enkelt lenkje. Men ein slik metode er vanskeleg i praksis; når avskjeringa skjer så seint, blir det alt for mange moglege kombinasjonar for lengre setningar med mange ukjende ord til at ein vanleg datamaskin kan halde styr på dei.

Me må i alle tilfelle vere klar for ei setning der alle ord er ukjende (me har ingen informasjon om LPT-korrespondanse), slik at kvart kjeldeord kan lenkjast til kvart målord. Viss båe setningane er 4 ord, får me 16 moglege samanstillingar der alle ord er med i nøyaktig éi lenkje (2^l, kor l er setningslengd). Men ofte har me null-lenkjer, me må altså i tillegg tillate samanstillingar der minst eitt ord er ulenkja, utan at me treng å vite kva for ord det er; med desse kortare listene inkludert får me endå fleire moglege samanstillingar per setning (4 ord gir 26, 8 ord gir 2186 moglege samanstillingar). Sjølv om me heile tida vel dei samanstillingane som lenkjar flest ord, ville maskinen raskt fått problem. I tillegg har me problemet med 1-mange-lenkjer, som skaper endå fleire moglege samanstillingar.

Ein sideverknad av å byrje med ytre lenkjer og gå innover (prosessen skildra i del 4.1) er at me automatisk unngår å prøve «kryssande» lenkjer, t.d. å lenkje F_s med XCOMP av F_t, og XCOMP av F_s med F_t (denne kombinasjonen av lenkjer vil jo vere ein del av alle logisk moglege permutasjonar). Me får au prioritert å lenkje ytre element, som jo er sikrare lenkjer: gitt to f-strukturar for setningar der alt me veit om lenkjinga er at *setningane* er omsetjingar av kvarandre, vil dei to ytre f-strukturane ha størst sjanse for å korrespondere med kvarandre. For kvart steg du går innover må du multiplisere inn sjansen for å trå feil i argumentpermutasjonane.

4.2 SKRIV Rangering

Rangering er ikkje implementert enno (akkurat no gir rank(f-alignments) berre ut første samanstilling.) Rangering foregår etter ulike kriterium. Her er eit par forslag:

4.2.1 rekursivt lenkja > ulenkja, men LPT

aligntable seier om noko er rekursivt lenkja eller ikkje, plusspoeng viss me har klart å lenkje rekursivt.

4.2.2 argument-argument > argument-adjunkt

Plusspoeng for argument-argument-lenkjer, burde vere eit bra kriterium, men me får sjølvsagt problem viss LPT ikkje seier noko i døme **??** med
`da-najleveba<Abrams,Browne,regne>` adjunkt: `sigarett`
`bet<Abrams,sigarett,regne>` adjunkt: `Browne`

4.2.3 arg1-arg1 arg2-arg2 > arg1-arg2 arg2-arg1 (følgje)

Dette kjem til å gi problem når me vil lenkje «behage» og «like», viss me ikkje har motstridande LPT-informasjon (og argumentfølgje i leksikon ikkje er basert på semantikk, men syntaks). Men elles er det vel OK.

Enklaste implementasjon: Levenshtein-avstand. Men burde visse argument vektast? (T.d. vekte subjekt om alt anna er likt.)

Andre forslag: http://en.wikipedia.org/wiki/Edit_distance)

4.2.4 flest lenkja adjunkt

Usikker på dette... avheng av om me tillèt lenkjer på tvers av f-strukturar.

4.2.5 Prioritet på rangeringskriterium

Dette bør sjølvsagt testast empirisk, blir kanskje utanfor denne oppgåva (diskusjonsdel?), men kan jo prøve meg litt rundt.

4.3 Lenkjing av c-strukturnodar

Samanstilling mellom f-strukturar treng i `lfgalign` ikkje informasjon om c-strukturen, medan lenkjing av c-strukturnodar skjer på grunnlag av f-struktursamanstillinga. Programmet utfører difor samanstilling av c-strukturar sist.

Funksjonen `c-align` har som inndata c-strukturanalysane av kjelde- og målsetninga, og éi f-struktursamanstilling; utdata er ei mengd med lenkjer. Ei lenkje er eit par der første element er ei mengd c-strukturnodar på kjeldespråket, og andre element ei mengd nodar på målspråket. Det er ingen overlapp mellom medlem av lenkjer (ein node er aldri med i meir enn eitt par).

I Dyvik et al. (2009, s. 77) er kravet for å lenkje to c-strukturnodar er at dei dominerer same mengd med ordlenkjer[6]. Ein node *n* dominerer ei mengd lenkjer

[6]Dette er ein litt enklare måte å definere kravet på; ei *lenkje* refererer til både kjelde og mål, dimed blir det mogleg å seie at ein node på kjeldespråket kan dominere same mengd som ein node på målspråket.

l viss unionen av lenkjene dominert av døtrene til n er lik l. I `lfgalign` opererer eg ikkje med *ordlenkjer* i seg sjølv; f-struktursamanstillinga er basert på LPT-korrespondansar, som definerer moglege ordlenkjer utan å sjå på kontekst, og f-struktursamanstillinga avgrenser vidare moglege ordlenkjer gitt f-strukturinformasjon. Preterminale nodar er dei mest ordnære nodane som kan ha ei f-strukturlenkje (ved ϕ); når formålet er å lenkje c-strukturnodar kan me nytte f-strukturlenkja til den preterminale noden i staden for ordlenkjer.

To problem (kva vil me ha med?)
1. me får *ikkje* med LPT-korrespondansar som er OK, men ikkje med i $f-alignment$;
2. me *får* med LPT-korrespondansar som er med i $f-alignment$ men ikkje *aligntable* (ikkje er rekursivt lenkja).

c-alignments $\leftarrow \emptyset$;
$splits_s \leftarrow$ new table ;
add-links(f-alignment, $tree_s, splits_s$) ;
$splits_t \leftarrow$ new table ;
add-links(f-alignment, $tree_t, splits_t$) ;
forall the *links being the keys in* $splits_s$ **do**
  if *(links in* $splits_t$*)* **then**
    add $(splits_s[links], splits_t[links])$ to c-alignments ;
end
return c-alignments ;

Funksjon 3: c-align(f-alignment, $tree_s$, $tree_t$)

Hjelpeprosedyren `add-links` utfører hovudjobben. Inndata er rotnoden til c-strukturtreet for eitt av språka, og f-samanstillinga. Prosedyren kappar opp treet i nodemengder, kor kvar nodemengd dominerer same lenkjemengd (som definert over). Nodemengdene blir lagra i ein tabell, indeksert på lenkjemengdene. Prosedyren går rekursivt gjennom treet frå rot til lauv; lenkjemengden for kvar node er unionen av lenkjemengdene returnert av `add-links` kalt på kvar av døtrene. Viss ein node dominerer ei lenkjemengd *links*, legg me til denne noden i tabellen $splits[links]$.

$links \leftarrow \emptyset$;
if *node* **then**
  if *preterminal?(node)* **then**
    let $link \in$ f-alignment s.t. $\phi(node) \in link$;
    if *link* **then** $links \leftarrow \{link\}$
  else
    links $\leftarrow$add-links(f-alignment, left-branch(*node*)) $\bigcup$
    add-links(f-alignment, right-branch(*node*)) ;
  add *node* to $splits[links]$;
return *links* ;

Funksjon 4: add-links(f-alignment, $node, splits$)

Sidan `c-align` kallar `add-links` for kvar av sidene, får me to tabellar $splits_s$ og $splits_t$. Me hentar så ut alle dei lenkjemengdene som er i båe tabellane (dvs.

snittet av oppslagsnøklene til tabellen); nodane som er lagra med mengd med f-strukturlenkjer skal lenkjast på c-strukturnivå. Alle desse mange-til-mange-lenkjene blir til slutt returnert av `c-align`.

Prosessen er no ferdig, mange-til-mange-lenkjene mellom c-strukturnodar definerer frasesamanstillinga

til diskusjonsdel:*Det er ikkje berre ei N-gramsamanstilling; sidan lenkjene er mellom c-strukturnodar kor kvar node dominerer ein konstituent, kunne me kalt det ei* konstituentsamanstilling..

4.3.1 SKRIV viss me har LPT, men ikkje rekursiv f-lenkje

Dette bør kanskje vere valfritt i programmet, for å sjå kva det fører til: vil du ta med LPT-korrespondansar som ikkje har f-lenkjer i `add-links`?

Og omvendt, finst det LPT-korrespondansar som ikkje kjem med i f-alignment i det heile teke, men som likevel burde ha noko å seie for c-strukturlenkjinga? (Men burde dei ikkje då vere med i f-alignment au?)

Kan me **fjerne** visse f-samanstillingar pga. c-strukturinfo? dvs. disambiguere... (dette er vel heller stoff for diskusjonsdelen?)

Kapittel 5

Diskusjon, resultat av å automatisk samanstille norske og georgiske setningar

Kapittel 6

Avslutning

Referansar

Alsina, A., Bresnan, J. & Sells, P. (red.). (1997). *Complex predicates*. Stanford, CA, USA: Center for the Study of Language and Information. Paperback.

Aronson, H. (1990). *Georgian. A Reading Grammar. Corrected Edition*. Columbus, OH: Slavica Publishers.

Bresnan, J. (2001). *Lexical-Functional Syntax*. Oxford, UK: Blackwell Publishers. Tilgjengeleg frå http://books.google.com/books?id=7elu0CcxQWkC (ISBN: 0631209743)

Brown, P.F., Della Pietra, S.A., Della Pietra, V.J. & Mercer, R.L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*(2), 263–311. Tilgjengeleg frå http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8919

Butt, M., Dyvik, H., King, T., Masuichi, H. & Rohrer, C. (2002). The Parallel Grammar Project. I *COLING-02 on Grammar engineering and evaluation* (vol. 15, s. 1–7). Morristown, NJ: Association for Computational Linguistics. Tilgjengeleg frå http://portal.acm.org/citation.cfm?id=1118783.1118786

Cheung, L., Lai, T., Luk, R., Kwong, O., Sin, K., Tsou, B. et al. (2002). Some Considerations on Guidelines for Bilingual Alignment and Terminology Extraction. , 1–5. Tilgjengeleg frå http://www.aclweb.org/anthology-new///W/W02/W02-1802.pdf

Cyrus, L., Feddes, H. & Schumacher, F. (2004). Annotating predicate-argument structure for a parallel treebank. *LISBON*, *2004*, 39. Tilgjengeleg frå http://arxiv.org/abs/cs/0407002

Dyvik, H., Meurer, P., Rosén, V. & Smedt, K.D. (2009). Linguistically motivated parallel parsebanks. I M. Passarotti, A. Przepiórkowski, S. Raynaud & F.V. Eynde (red.), *Proceedings of the eighth international workshop on treebanks and linguistic theories* (s. 71–82). Milan, Italy: EDUCatt. Tilgjengeleg frå http://tlt8.unicatt.it/allegati/Proceedings_TLT8.pdf#page=83

Giegerich, H. (2006). Attribution in English and the distinction between phrases and compounds'. *Englisch in Zeit und Raum-English in Time and Space: Forschungsbericht für Klaus Faiss. Trier: Wissenschaftlicher Verlag Trier*. Tilgjengeleg frå http://www.englang.ed.ac.uk/people/attributioninenglish.pdf

Hearne, M., Ozdowska, S. & Tinsley, J. (2008). Comparing Constituency and Dependency Representations for SMT Phrase-Extraction. I *Actes de la 15e Conférence Annuelle sur le Traitement Automatique des Langues Naturelles (TALN '08)*. Avignon, France. Tilgjengeleg frå http://www.computing.dcu.ie/~mhearne/publications.html

Koehn, P., Och, F. & Marcu, D. (2003). Statistical phrase-based translation. I *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* (s. 48–54). Morristown, NJ, USA. Tilgjengeleg frå http://www.iccs.inf.ed.ac.uk/~pkoehn/publications/phrase2003.pdf

Kruijff-Korbayova, I., Chvatalova, K. & Postolache, O. (2006). Annotation guidelines for Czech-English word alignment. , 1256–1261. Tilgjengeleg frå http://www.mt-archive.info/LREC-2006-Kruijff.pdf

Maier, R.M. (2009). *Structural Interference from the Source Language: A psycholinguistic investigation of syntactic processes in non-professional translation*. Upublisert akademisk avhandling, University of Edinburgh. Tilgjengeleg frå http://linguistlist.org/issues/20/20-1786.html

Meurer, P. (2008, March). *A Computational Grammar for Georgian.* Tilgjengeleg frå http://maximos.aksis.uib.no/~paul/articles/Tbilisi2007-LNAI.pdf

Munday, J. (2001). *Introducing Translation Studies: Theories and Applications*. London: Routledge.

Piao, S. & McEnery, T. (2001). Multi-word Unit Alignment in English-Chinese Parallel Corpora. I P. Rayson, A. Wilson, T. McEnery, A. Hardie & S. Khoja (red.), *Proceedings of the Corpus Linguistics 2001 Conference* (s. 466–475). Lancaster, UK. Tilgjengeleg frå http://personalpages.manchester.ac.uk/staff/scott.piao/research/papers/mwu_align4.pdf

Pullum, G. & Scholz, B. (2001). On the Distinction between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. *Logical Aspects of Computational Linguistics: 4th International Conference, Lacl 2001, Le Croisic, France, June 27-29, 2001, Proceedings*. Tilgjengeleg frå http://portal.acm.org/citation.cfm?id=645668.665062

Riezler, S. & Maxwell, J. (2006). Grammatical Machine Translation. I M. Butt, M. Dalrymple & T.H. King (red.), *Intelligent Linguistic Architecture: Variations on themes by Ronald M. Kaplan* (s. 35–52). Stanford, CA: CSLI Publications. Tilgjengeleg frå http://www.parc.com/research/publications/details.php?id=5675

Rosén, V., Meurer, P. & Smedt, K. de. (2009). LFG Parsebanker: A Toolkit for Building and Searching a Treebank as a Parsed Corpus. I F.V. Eynde, A. Frank, G. van Noord & K.D. Smedt (red.), *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories (TLT7)* (s. 127–133). Utrecht: LOT. Tilgjengeleg frå http://ling.uib.no/~desmedt/papers/tlt7rosen-submitted.pdf

Samuelsson, Y. & Volk, M. (2006). Phrase Alignment in Parallel Treebanks. I *Proceedings of Treebanks and Linguistic Theories (TLT '06).* Prague. Tilgjengeleg frå http://ling16.ling.su.se:8080/new_PubDB/doc_repository/229_align.pdf

Samuelsson, Y. & Volk, M. (2007). Automatic Phrase Alignment: Using Statistical N-Gram Alignment for Syntactic Phrase Alignment. I *Proceedings of Treebanks and Linguistic Theories (TLT '07).* Bergen, Norway.

Thunes, M. (2003). *Ekserpering av leksikalske oversettelsekorrespondanser fra parallelltekst.* Tilgjengeleg frå http://www.hf.uib.no/i/LiLi/SLF/ans/Dyvik/marthaex.pdf

Tinsley, J., Hearne, M. & Way, A. (2007). Exploiting Parallel Treebanks to Improve Phrase-Based Statistical Machine Translation. I *Proceedings of Treebanks and Linguistic Theories (TLT '07).* Bergen, Norway.

XPar. (2008). *XPAR: Language diversity and parallel grammars.* (Submitted to the Research Council of Norway.)