Vloeiendheid in generatie

Natuurlijke Taalverwerking I - Week7

Een voorproefje...

- Hoe kiezen we de meest vloeiende zin uit een verzameling van zinnen?
  - *pas in 1842 werd er een geregelde kabinetsvergadering in het leven geroepen*
  - *er werd pas in 1842 een geregelde kabinetsvergadering in het leven geroepen*
  - *[...]*
  - *'n geregelde kabinet-vergadering werd in het leven er pas in 1842 geroepen*

Overzicht van vandaag

- Generatie
  - Toepassingen
  - Chart generatie
- Fluency ranking
  - N-gram modellen
  - Feature-gebaseerde modellen
- Evaluatie van fluency ranking
- Tentamen

GENERATIE

Parsing/generatie

- **Parsing:** geef de logische vorm van een zin.
- **Generatie:** bouw alle mogelijk zinnen gegeven een logische vorm.
- **Logische vorm:** abstracte representatie van een zin, zoals dependency structure, of een verzameling van predikaten.

Een voorbeeld

- **Parsing:** *Jan liep snel* -->
  - lopen(l), vt(l), snel(l), arg1(l,j), name(j,Jan)
- **Generatie:** lopen(l), vt(l), snel(l), arg1(l,j), name(j,Jan) -->
  - *Jan liep snel*
  - *Jan liep vlot*

Toepassingen

- **Controleren van een grammatica:** als een grammatica te veel toestaat (niet voldoet aan het ‘only’ principe), dan zal een generator ongrammaticale zinnen maken.
- **Parafrasering:** herschrijven van een zin die niet vloeiend is, door een zin te ontleden en vanaf de resulterende logische vorm te genereren.
- **Zinsfusie:** het combineren van de semantiek van twee of meer zinnen.
- **Zinscompressie:** verwijderen van elementen van een zin die weinig informatiewaarde hebben.
- **Machine translation:** genereren van een zin in een andere taal.

Parafraseren

- Zinnen parafraseren om ze vloeiender te maken.
- Zinnen parafraseren om informatie te coderen (watermarking): Topkara et al., 2005
- Verbeteren van statistische machinevertaling met behulp van parafraseringen: Callison-Burch et al., 2006
- Parafraseren van vragen voor websites die gebruikersondersteuning bieden (STEVIN DAISY project).

Zinsfusie

- Marsi & Krahmer 2005: intersection fusion en union fusion
- Bijvoorbeeld:
  - *Christina Aguilera has confirmed, in the American magazine Glamour, that she is pregnant.*
  - *Christina Aguilera has finally asserted what the whole world already knew: she is expectant.*
- **Union fusion:** *Christina Aguilera has finally confirmed, in the American magazine Glamour, what the whole world already knew: she is pregnant.*
- **Intersectie-fusie:** *Christina Aguilera has confirmed that she is pregnant.*
- Combineer de semantiek en genereer een zin.

Zinscompressie

- Comprimeren van zinnen door het verwijderen van constituenten met een lage informatiewaarde.
- James Clarke & Mirella Lapata, 2008:
  - *The Clinton administration recently unveiled a new means to encourage brownfields redevelopment in the form of a tax incentive proposal.*
  - *The Clinton administration unveiled a means to encourage brownfields redevelopment in a tax incentive proposal.*
- Meer dan simpelweg het verwijderen van woorden: generatie.
- Bruikbaar voor het maken van samenvattingen en ondertiteling.

CHART GENERATIE

Bottom-up parsing

- Begin bij de input (woorden);
- bepaal de categorieën van woorden,
- combineer categorieën tot zinsdelen,
- combineer zinsdelen tot grotere zinsdelen.

Bottom-up generatie

- Begin bij de semantiek;
- bepaal de woorden en categorieën die een deel van de semantiek uitdrukken;
- combineer categorieën tot zinsdelen met inachtneming van de semantiek;
- combineer zinsdelen tot grotere zinsdelen met inachtneming van de semantiek;
- loop de gegenereerde bomen top-down, left-right door, en extraheer de woorden.

Chart generatie

- Zet lexical items (woorden + categorieën) die de semantiek vertegenwoordigen op de agenda.
- Voor elke agenda item:
  - Kijk of er een grammaticaregel in te vullen is met dit item, en eventueel een item op de chart. Combineer de semantiek van de dochters, en controleer of je een stuk semantiek niet te vaak introduceert. Plaats alle mogelijke ingevulde regels op de agenda.
  - Verplaats de item van de agenda naar de chart.

Chart generatie (voorbeeld)

- *Jan liep snel*
- lopen(l), vt(l), snel(l), arg1(l,j), name(j,Jan)

Chart generation (example)

Lexicon:

Word	Category	Semantics
Jan	np(x)	name(x,Jan)
liep	vp(x,y)	lopen(x), vt(x), arg1(x,y)
snel	adv(x)	snel(x)
vlot	adv(x)	snel(x)

Grammar:

```
s(x)     -> np(y), vp(x,y)
vp(x,y) -> vp(x,y), adv(x)
```

N	Surface	Cat	Sem
1	Jan	np(j)	name(j,Jan)
2	liep	vp(l,y)	lopen(l), vt(r), arg1(r,x)
3	snel	adv(x)	snel(x)
4	vlot	adv(x)	snel(x)
5 (1,2)	Jan liep	s(l)	lopen(l), vt(l), arg1(l,j), name(j,Jan)
6 (2,3)	liep snel	vp(l,y)	lopen(l), vt(l), arg1(l,y), snel(r)
7 (2,4)	liep vlot	vp(l,y)	lopen(l), vt(l), arg1(l,y), snel(l)
8 (1,6)	Jan liep snel	s(l)	lopen(l), vt(l), arg1(l,j), name(j,Jan), snel(l)
9 (1,7)	Jan liep vlot	s(l)	lopen(l), vt(l), arg1(l,j), name(j,Jan), snel(l)

FLUENCY RANKING

De noodzaak van fluency ranking

- Een grammatica staat vaak toe dat er meerdere zinnen (realisaties) gegenereerd worden die aan de semantiek voldoen.
- Niet elke realisatie is even vloeiend.
- Een voorbeeld met behulp van de Alpino chart generator:
  - *omdat zijn rol toen echt wel uit was gespeeld*
  - *omdat echt wel toen z'n rol was uit gespeeld*
  - *omdat wel zijner rol echt waart uit gespeeld toen*
- Voor een set van 3688 logische vormen uit Eindhoven corpus met gemiddeld 15.7 lexicale items, was het gemiddelde aantal realisaties 38.5.
- Dus: het is noodzakelijk de meest vloeiende zin te kiezen.

Fluency als waarschijnlijkheid

- We kunnen de vloeiendheid van een zin beschouwen als de waarschijnlijkheid waarmee die zin in een taal voorkomt: de zin *y* die *P(y)* maximaliseert is de meest vloeiende.
- *P(y)* kunnen we bepalen met een waarschijnlijkheidsmodel.
- Een voor de hand liggend model schat de waarschijnlijkheid van de opeenvolging van woorden.

$$P(y) = P(w_1, w_2 ... w_n) = \frac{C(w_1, w_2 ... w_n)}{N}$$

Gebruik van een tekstcorpus

- Schatten van waarschijnlijkheden met een tekstcorpus.
- Nederlandstalige Wikipedia (2008): 109 miljoen woorden, 7 miljoen zinnen.
- Probleem: aantal mogelijke zinnen is oneindig, Wikipedia bevat een eindig aantal zinnen, waardoor de waarschijnlijkheid van veel zinnen *0.0* zal worden:

```
$ grep -c 'Bevat Wikipedia alle mogelijke zinnen \?' \
nlwikipedia.txt
0
```

Chain rule

- We kunnen de waarschijnlijkheid P(w_1,w_2...w_n) herschrijven met behulp van de kettingregel (chainrule):

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1})$$

- Voorbeeld:

$$P(w_3|w_1^2) = \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)}$$

- Schieten we hier iets mee op?

N-grammen

- *n-gram:* n opeenvolgende woorden of karakters uit een reeks woorden of karakters
- *De kat zit op de mat*
- Woord bigrammen (2-grammen): *de kat*, *kat zit*, *zit op*, *op de*, *de mat*
- Woord trigrammen (3-grammen): *de kat zit*, *kat zit op*, *zit op de*, *op de mat*

N-grammen (2)

- Hoewel zinnen niet in een corpus voor zullen komen, komen korte n-grammen meestal wel voor:
  - *Bevat Wikipedia*: 0
  - *Wikipedia alle*: 1
  - *alle mogelijke*: 458
  - *mogelijke zinnen*: 3
  - *zinnen ?*: 3

Bigram model

- We kunnen een benadering maken van de waarschijnlijkheid van het volgende woord met behulp van een bigram:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

- Of voor een hele zin:

$$P(w_1^n) \approx P(w_1|start)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1}), P(end|w_n)$$

- Waarin:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

- Formeel: *Markov property* - de volgende staat is alleen afhankelijk van de huidige staat.

Simpel voorbeeld

- Corpus:
  - What is the meaning of life ?
  - The meaning of life is 42 .
- Zin:
  - What is 42 .
- Uitwerking op het bord...

Trigram model

- De gebruikte n-gram lengte in een model is een trade-off:
  - Langere n-grammen bevatten meer contextuele informatie.
  - Maar hoe langer de n-gram, hoe schaarser je trainingsmateriaal.
- In de praktijk blijken trigrammen een goede balans te vormen tussen deze trade-offs.

Smoothing

- Hoewel het gebruik van n-grammen het schaarste-probleem kleiner maakt, is het nog niet opgelost:
  - Nieuwe woorden.
  - Domein-specifieke zinnen of trainingsmateriaal.
  - Te weinig trainingsmateriaal.
- *[Bevat Wikipedia] alle mogelijke zinnen?*
- *P(y)* wordt berekend door waarschijnlijkheden van n-grammen te vermenigvuldigen.
- Als een n-gram de waarschijnlijkheid *0* heeft, dan *P(y) = 0*.

Smoothing (2)

- Om een zin toch een waarschijnlijkheid > *0* te kunnen geven, moeten we de waarschijnlijkheid van de n-grammen die niet in de trainingsdata voorkwamen schatten: smoothing.
- Veel verschillende methoden, zoals linear interpolation smoothing en back-off smoothing.
- Frequente benadering: als een n-gram niet voorkomt in een corpus, komt een (n-1)-gram misschien wel voor.

Problemen n-gram model

- Alleen kennis van de oppervlakte van de zin (opeenvolging van woorden).
- Het constructieproces van de zin wordt niet in acht genomen.

FEATURE-GEBASEERDE MODELLEN

Oppervlakte-eigenschappen

Sommige fenomenen die niet vloeiend zijn, kunnen we vinden door naar een zin te kijken:

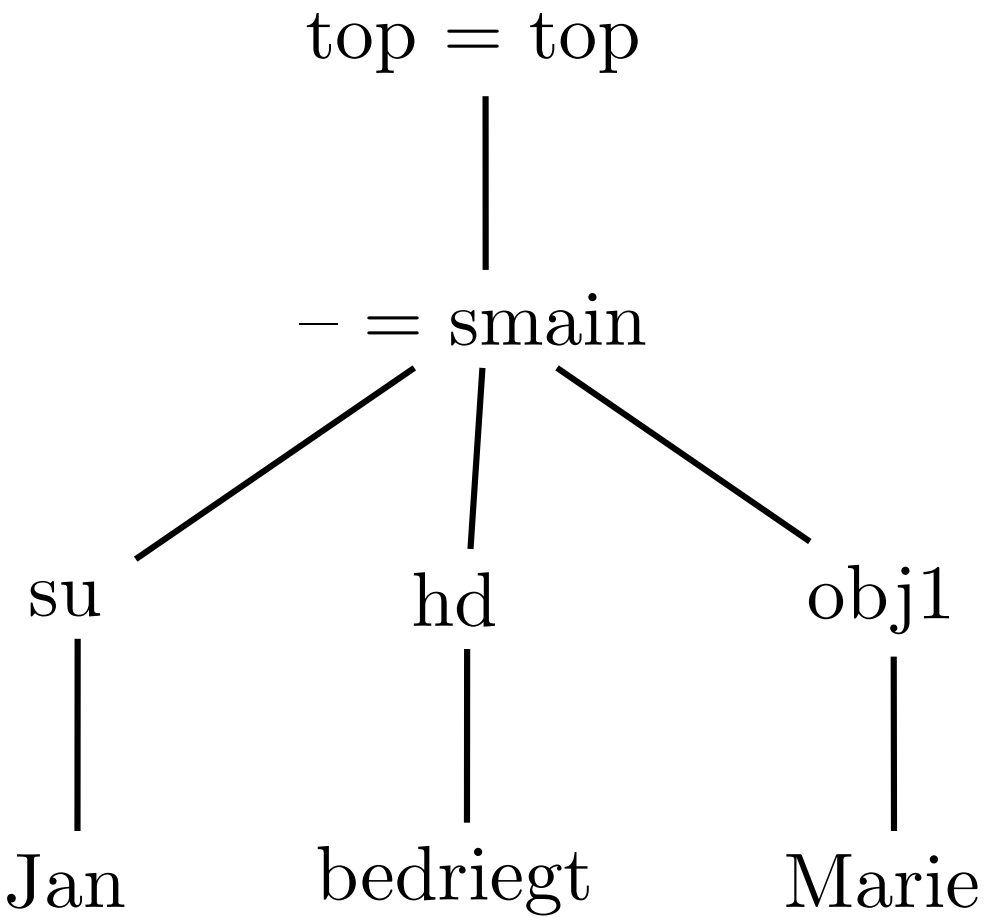
- Archaïsche woorden: *was*/*waar*, *zijn*/zijner
- Volgordes in coordinaties: *de man en zijn kind*/*zijn kind en de man*
- Idiomen: *van horen en zeggen*/*van zeggen en horen*
- Partikels: *omdat ik hem opbel*/*omdat ik hem op bel*

Structurele eigenschappen

Maar soms hebben we structurele informatie nodig:

- Barack Obama won de verkiezingen
- de verkiezingen won Barack Obama
- Jan bedriegt Marie
- Marie bedriegt Jan

Subject/direct object-volgorde



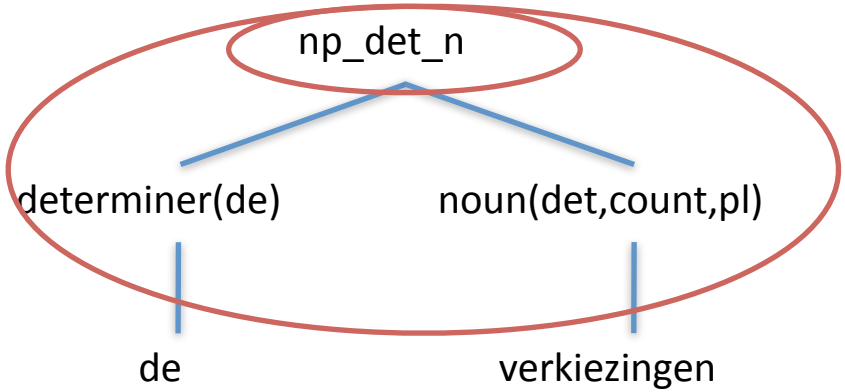
Features

- Nodig: een raamwerk waarin we allerlei kenmerken van vloeiende/niet-vloeiende zinnen kunnen opnemen.
- Maximum entropy models.
- Interessante informatie voor fluency ranking kunnen gemodelleerd worden als features:
  - *Features op basis van de genereerde zin*
  - *Features op basis van de derivatieboom*
- Features kunnen arbitraire waarden hebben, zoals frequenties of scores.

Features gegenereerde zin

- *Barack Obama won de verkiezingen*
  - ngram_lm → 64.3 (- log $P(w_0..w_n)$)
- *proper_name verb determiner noun*
  - ngram_tag → 11.9 (- log $P(t_0..t_n)$)

Features derivatieboom



- r1(np_det_n) → 1
- r2(np_det_n,1,determiner(de))
- r2(np_det_n,2,noun(det,count,pl)
- r1(n_adj_n) → 0

Automatisch extraheren features

- In de praktijk worden bijna alle features automatisch gegenereerd met behulp van feature templates.
- Bijv., extraheer *r1(Rulename)* voor elke knoop in een derivatieboom.
- De meeste fluency ranking modellen zijn groot (tienduizenden tot honderdduizenden features).

Het bouwen van een model

- We willen een model dat de realiteit zo goed mogelijk weerspiegeld.
- We gebruiken een trainingscorpus als een geannoteerde sample van de realiteit.
- Een trainingscorpus bestaat uit:
  - Logische vormen
  - Voor elke logische vorm een set van realisaties, met kwaliteitsscore.
- Bestaande software kan op basis van een trainingscorpus de gewichten voor elk feature schatten.

Fluency ranking

$$score(y|x) = \sum_i \lambda_i f_i(x, y)$$

- De score van een zin (*y*) gegeven een logische vorm (*x*).
- $f_i(x,y)$: Frequentie van feature *i* in realizatie *y* voor logische vorm *x*.
- λ_i : Gewicht van feature *i*.

Effectieve features fluency ranking

Feature	Polariteit (vloeiend/niet-vloeiend)
ngram_lm	+
ngram_tag	+
r1(vp_v_mod)	+
r1(optpunct(e))	+
s1(subj_np_topic)	+
r2(vp_mod_v,3,vproj_vc)	+
r2(vpx_vproj,1,vp_arg_v(np))	+
r2(non_wh_topicalization(modifier),1,mod2)	+
r2(vp_arg_v(pp),2,vproj_vc)	+
r2(vp_arg_v(pred),2,vproj_vc)	+

EVALUATIE

Doel evaluatie

- We willen een fluency ranking model kunnen evalueren:
  - Leveren veranderingen aan het model betere resultaten?
  - Hoe goed werkt model X vergeleken met model Y?

Benodigdheden

- Voor een gegeven logische vorm hebben we nodig:
  - De zin die de fluency ranker suggereerde.
  - Handmatig gekozen meest vloeiende zin (gold standard).
- Evaluatiedata bestaat uit een verzameling van logische vormen plus gewenste zinnen.
- Om eerlijk te kunnen evalueren moeten we de evaluatiedata in twee sets verdelen:
  - De test set: wordt tijdens het ontwikkelen van het model gebruikt om te evalueren.
  - De evaluatie set: alleen voor de uiteindelijke evaluatie, om specifieke optimalisatie voor de evaluatieset te voorkomen.

Evaluatie als goed/fout ratio

- Een simpele evaluatiemethode:
  - Tel het aantal gevallen waarin de fluency ranker de meest vloeiende zin koos, gegeven een logische vorm.
  - Dit aantal delen we door het totale aantal logsiche vormen die gebruikt worden voor de evaluatie.
  - Bijv: we hebben 100 logische vormen waarmee we evalueren, en in 60 gevallen koos de fluency ranker de meest vloeiende zin. Dan is de score: *60/100 = 0.6*.

Evaluatie als goed/fout ratio

- Een realisatie die vrijwel goed is, draagt niet bij aan de score. Neem bijvoorbeeld de volgende referentie (*gold standard*) en realisatie:
  - `R: hij is rood aangelopen`
  - `C: hij is rood aan gelopen`
- Dit wordt nog problematischer als we bedenken dat realisaties die bijna vloeiend zijn dezelfde score krijgen als realisaties die helemaal niet vloeiend zijn. Bijvoorbeeld:
  - `R:  ik ben nog steeds op zoek`
  - `C1: op zoek ben ik nog steeds`
  - `C2: opzoek zijt nog steeds ik`

Evaluatie als goed/fout ratio

- Conclusie: evalueren op basis van het aantal correcte zinnen is vaak te grof.
- We hebben een verfijndere maat nodig die een score aan een realisatie toe kan kennen.

Wat te meten?

- Zijn de verwachte woorden aanwezig?
- Zijn de woorden in de verwachte volgorde?

ROUGE

- Verzameling van evaluatiematen voor machinevertaling (komt de vertaling overeen met de verwachte vertaling?)
- Neemt aanwezigheid en volgorde van woorden op verschillende manieren in beschouwing.
- Voor vandaag: ROUGE-N en ROUGE-L.

ROUGE-N

$$ROUGE - N = \frac{|ngrams(R) \cap ngrams(C)|}{|ngrams(R)|}$$

ROUGE-2

- Gold standard: *ik heb de trein gemist*
- Realisatie: *de trein heb ik gemist*
- Gold standard 2-grams: *ik heb*, *heb de*, *de trein*, *trein gemist*
- Realisatie 2-grams: *de trein*, *trein heb*, *heb ik*, *ik gemist*
- ROUGE-2: *1/4 = 0.25*

Longest common subsequence

- Een *subsequence* is een opeenvolging van elementen uit een reeks die dezelfde volgorde als de reeks aanhoudt. Subsequences in *ABC* zijn bijvoorbeeld: *A*, *B*, *C*, *AB*, *AC*, *BC*, en *ABC*.
- De *Longest Common Subsequence* (LCS) is de langste subsequence die twee reeksen delen. De LCS van *ABCD* en *ABDE* is bijvoorbeeld *ABD*.

LCS (voorbeelden)

```
1:  de politie was ter plaatse
2:  ter plaatse was de politie
LCS: de politie, ter plaatse

1:  mooi weer met hier en daar een bui
2:  mooi weer met ’n bui hier en daar
LCS: mooi weer met hier en daar

1:  de NS zet bussen in
2:  bussen zet de NS in
LCS: de NS in
```

ROUGE-L

$$R_{lcs} = \frac{LCS(R, C)}{length(R)}$$

$$P_{lcs} = \frac{LCS(R, C)}{length(C)}$$

$$ROUGE - L = \frac{2 \cdot R_{lcs} P_{lcs}}{R_{lcs} + P_{lcs}}$$

ROUGE-L voorbeeld

R: de NS zet bussen in

C: bussen zet de NS in

LCS: de NS in

- $R_{lcs} = 3 / 5 = 0.6$
- $P_{lcs} = 3 / 5 = 0.6$
- $F_{lcs} = (2* 0.6 * 0.6) / (0.6 + 0.6) = 0.6$

ROUGE-L voorbeeld (2)

```
R:   ik wilde haar zien of opbellen
C:   ik wilde d'r op bellen of zien
LCS: ik wilde zien
```

- $R_{lcs} = 3 / 6 = 0.5$
- $P_{lcs} = 3 / 7 = 0.43$
- $F_{lcs} = (2* 0.5 * 0.43) / (0.5 + 0.43) = 0.46$

Opdracht van deze week

- Klein setje van logische vormen en realisaties annoteren.
- Toepassing van evaluatiemethoden, per geval en voor een heel corpus.
- Foutanalyse.