Sentence generation

Daniël de Kok <d.j.a.de.kok@rug.nl>

Introduction

- **Parsing:** build all possible logical forms for a given sentence.
- **Generation:** build all possible sentences given a logical form.
- **Logical form:** abstract representation of meaning.

An example

- Parsing: John ran fast -->
  - *run(r), past(r), fast(r), arg1(r,j), name(j,John)*

- Generation: *run(r), past(r), fast(r), arg1(r,j), name(j,John)* -->
  - *John ran fast*
  - *John ran quickly*

This hour's topics

- Introduction
  - Applications
  - Chart generation
- Fluency ranking
  - Features
  - Maximum entropy models

INTRODUCTION

Applications

- **Checking a grammar:** if a grammar is too permissive, using it with a generator will create ungrammatical sentences.
- **Paraphrasing:** rewriting a (non-fluent) sentence by parsing the sentence and generating from the resulting semantics.
- **Sentence fusion:** combining the semantics of two (or more sentences.
- **Sentence compression:** removing non-salient elements of a sentence.
- **Machine translation:** generating a sentence in a different language (interlingua or transfer-based MT).

Paraphrasing

- Rephrase sentences to make them more fluent.
- Rephrase sentences to encode information (watermarking): Topkara et al., 2005
- Improving statistical machine translation using paraphrases: Callison-Burch et al., 2006
- Paraphrasing of questions for user support sites (STEVIN DAISY project).

Sentence fusion

- Marsi & Krahmer 2005: intersection fusion and union fusion
- Consider:
  - *Christina Aguilera has confirmed, in the Amercian 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.*
- Intersection fusion: *Christina Aguilera has confirmed that she is pregnant.*
- Fuse semantics & generate a sentence from semantics.
- Creating more reliable or more extensive news text.

Sentence compression

- Compress sentences by removing non-salient constituents.
- 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 unveiled a means to encourage brownfields redevelopment in a tax incentive proposal.*
- To go beyond simple word removal generation is required.
- Useful in summarization and subtitling

Chart generation

- Build a tree bottom-up using lexical items and grammar rules.
- Do not specify positions in lexical items, and let the grammar decide on allowed word orders.
- Keep track of semantics during the generation, and only allow for items which semantics subsume a part of the goal semantics.
- Higher complexity than chart parsing (exponential rather than cubic).

FLUENCY RANKING

The need for fluency ranking

- A grammar will often allow for more than one surface sentence (realization) to be generated.
- But not every realization is equally fluent.
- One example generated with the 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*
- For a set of 7657 sentences from Wikipedia from 5 to 15 words, the average number of realizations, allowing minimal punctuation was 83.8
- So, we need methods to rank realizations, and pick the most fluent realization.

Surface characteristics

Some influencies can be detected by looking at the generated sentence, for example:

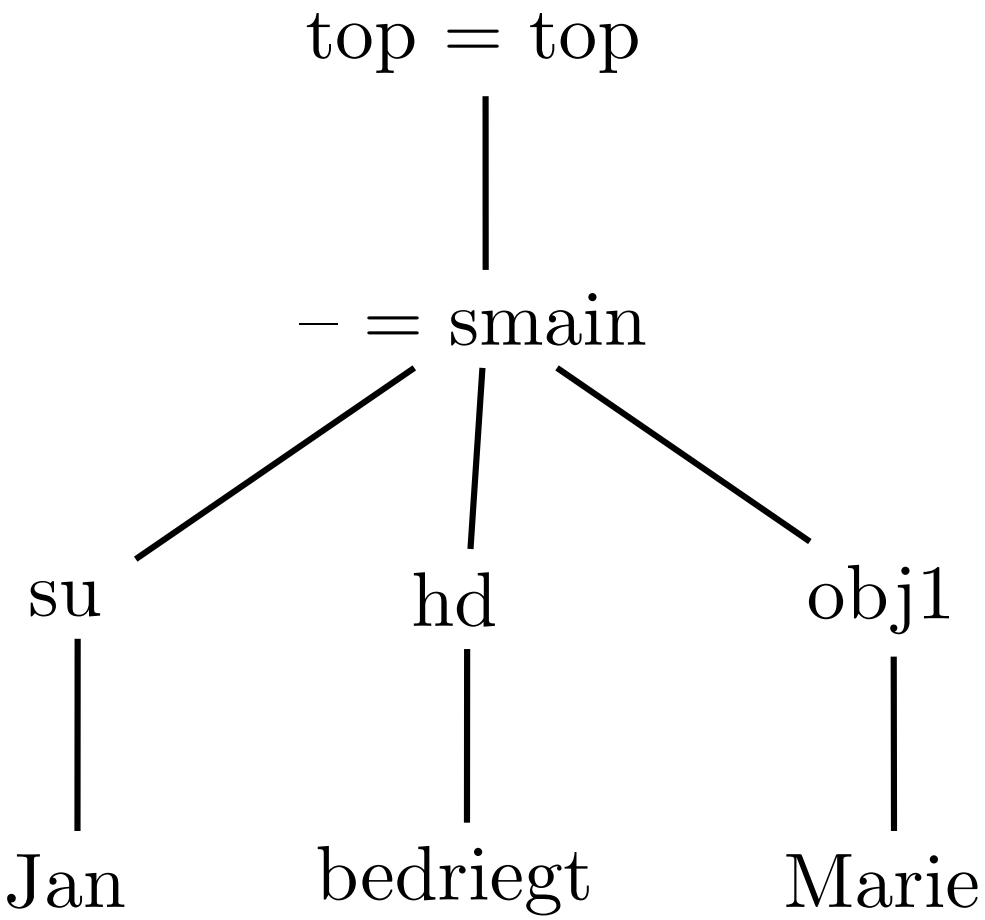
- Archaic words: *was*/*waar*, *zijn*/zijner
- Preferred coordination orders: *de man en zijn kind*/*zijn kind en de man*
- Idioms: *van horen en zeggen*/*van zeggen en horen*
- Particles: *omdat ik heb opbel*/*omdat ik hem op bel*

Structural characteristics

But some influencies can only be detected by looking at structure. For example:

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

Subject/direct object-order



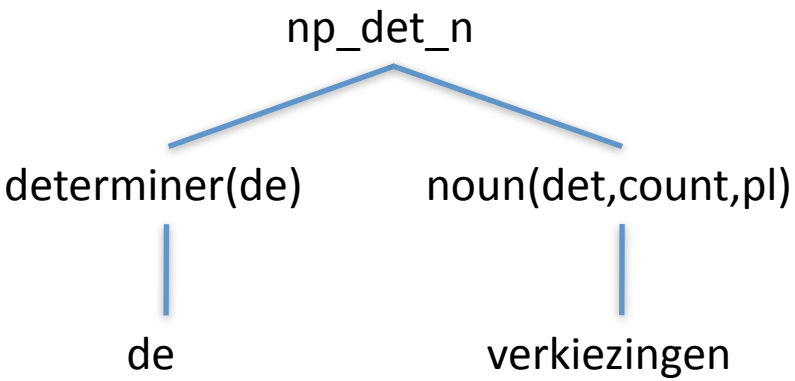
Features

- We need a framework where we can integrate all kinds of information.
- Interesting information for fluency ranking can be modeled as features:
  - Output features (generated sentence/surface)
  - Process features (derivation tree)
- Features can have arbitrary values, such as frequencies or scores.

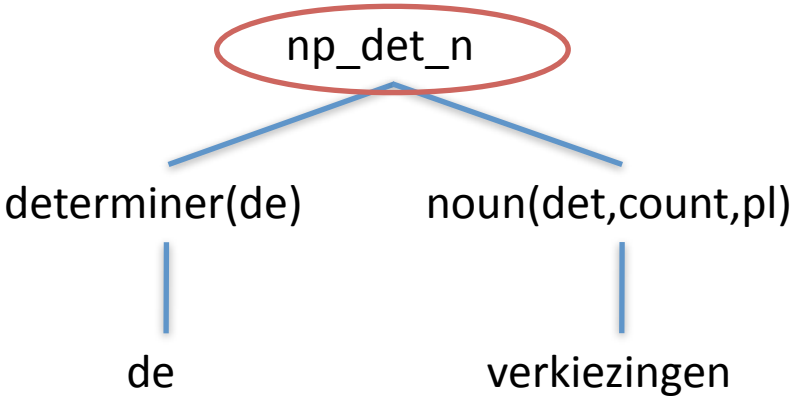
Output features

- *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 $P(t_0..t_n)$)

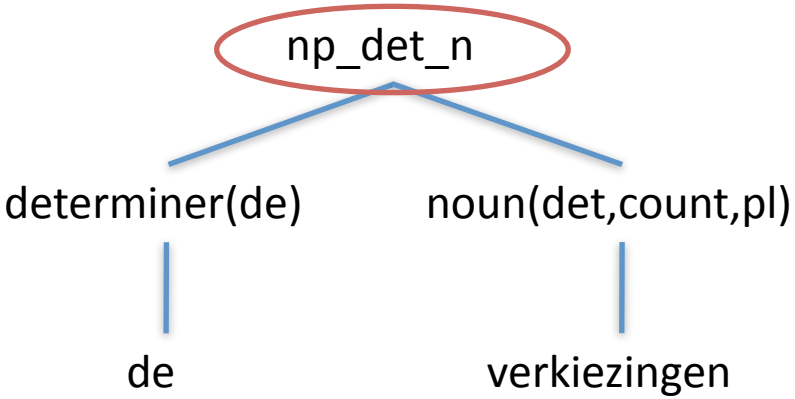
Derivation tree features



Derivation tree features

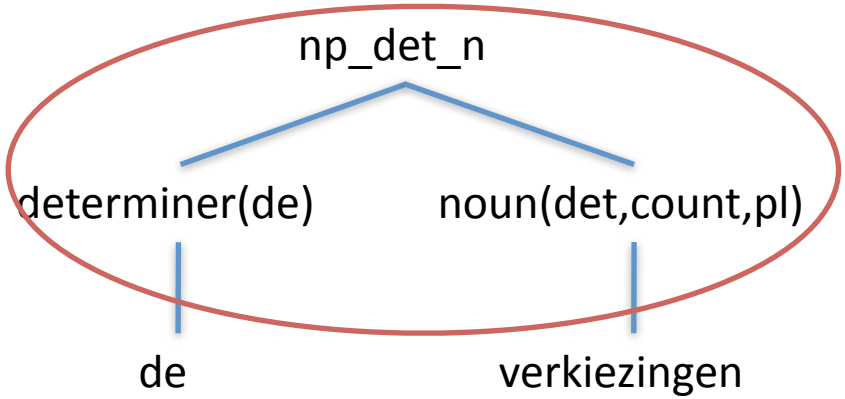


Derivation tree features



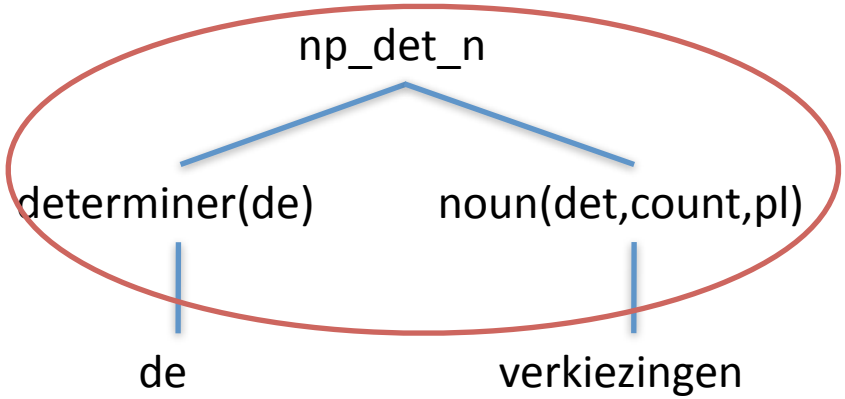
- r1(np_det_n) → 1

Derivation tree features



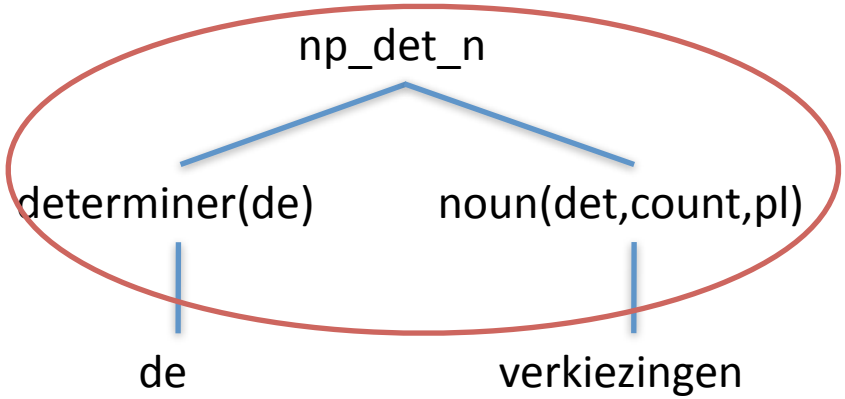
- r1(np_det_n) → 1

Derivation tree features



- r1(np_det_n) → 1
- lds(np_det_n,[determiner(de),noun(det,count,pl)],[]) → 1

Derivation tree features



- r1(np_det_n) → 1
- lds(np_det_n,[determiner(de),noun(det,count,pl)],[]) → 1
- r1(n_adj_n) → 0

Extracting many features

- In practice, nearly all features are automatically generated using feature templates.
- E.g., extract *lds(Node,Daughters,OptParents)* for every node in a derivation tree.
- Most fluency models are huge (hundreds of thousand features).

Building a model

- We want a model that reflects reality as closely as possible.
- We use a training corpus as a (annotated) ‘sample’ of reality.
- A training corpus consists of:
  - Logical forms (contexts)
  - For each logical form a set of realizations (events)
  - Context and event probabilities.

Coin tossing

- p(head) = 0.4, p(tails) = 0.6
- Play: 1 Euro
- Win (head): get 40 cents
- Lose (tail): lose 1 Euro
- Fair game? Calculate the expected value!
- $E(profit) = -1 \cdot 0.6 + 1.4 \cdot 0.4 = -0.04$

Expected feature values

- Just as in coin tossing, we can calculate the expected value of a feature:

$$E_{\tilde{p}}(f) = \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

- And the expected model feature value:

$$E_{\sigma}(f) = \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y)$$

- Where *x* is a context, and *y* an event.

Building a model (2)

- The expected value of a given feature in a model can be constrained to that of the training data:

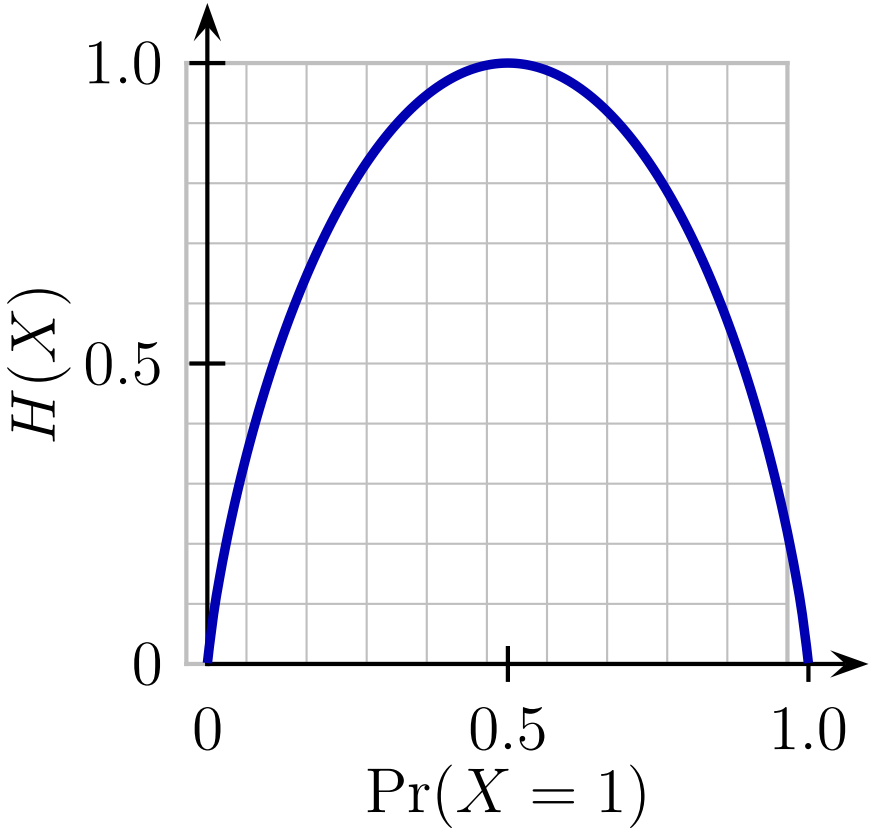
$$E_\sigma(f) = E_{\tilde{p}}(f)$$

- However, normally there are many possible models given a set of constraints on expected feature values.
- Which one to pick?

The maximum entropy principle

- We want to pick the model that has as few assumptions as possible, other than the feature expectation constraints.
- Occam's Razor: *"Of two equivalent theories or explanations, all other things being equal, the simpler one is to be preferred"*
- Given no constraints, only the uniform model has no assumptions.
- We can find the most uniform model by maximizing entropy.
- So, we want to find the model that maximizes entropy.

Entropy (Shannon, 1948)



$$H(X) \equiv -\sum_{i=1}^{n} p(x_i)\ log\ p(x_i)$$

Parametric form

- To be able to learn a model, we add a weight λ_i to every feature f_i.
- We can now calculate the probability of a realization (*y*) given a logical form (*x*):

$$p_w(y|x) = \frac{1}{Z_\lambda(x)} exp(\sum_i \lambda_i f_i(x,y))$$

$$Z_\lambda(x) = \sum_y exp(\sum_i \lambda_i f_i(x,y))$$

Fluency ranking

$$score(y|x) = \frac{1}{Z_\lambda(x)} exp(\sum_i \lambda_i f_i(x, y))$$

Fluency ranking

$$score(y|x) = \frac{1}{Z_\lambda(x)} exp(\sum_i \lambda_i f_i(x, y))$$

- In ranking we are not interested in actual probabilities, just relative order.

Fluency ranking

$$score(y|x) = \frac{1}{Z_\lambda(x)} exp(\sum_i \lambda_i f_i(x, y))$$

- In ranking we are not interested in actual probabilities, just relative order.
- The normalization factor is constant for an input.

Fluency ranking

$$score(y|x) = \frac{1}{Z(x)} exp(\sum_i \lambda_i f_i(x, y))$$

- In ranking we are not interested in actual probabilities, just relative order.
- The normalization factor is constant for an input.

Fluency ranking

$$score(y|x) = \frac{1}{Z(x)} exp(\sum_i \lambda_i f_i(x, y))$$

- In ranking we are not interested in actual probabilities, just relative order.
- The normalization factor is constant for an input.
- $e^n > e^m$ iff $n > m$

Fluency ranking

$$score(y|x) = \frac{1}{Z(x)} exp(\sum_i \lambda_i f_i(x, y))$$

- In ranking we are not interested in actual probabilities, just relative order.
- The normalization factor is constant for an input.
- $e^n > e^m$ iff $n > m$

Fluency ranking

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

- In ranking we are not interested in actual probabilities, just relative order.
- The normalization factor is constant for an input.
- $e^n > e^m$ iff $n > m$

Fluency ranking for Dutch

- Wide-coverage Alpino grammar and parser for Dutch.
- Now includes a chart generator and fluency ranker.
- Good coverage of Alpino test-suites.
- Also pretty good (but slow) on other data.

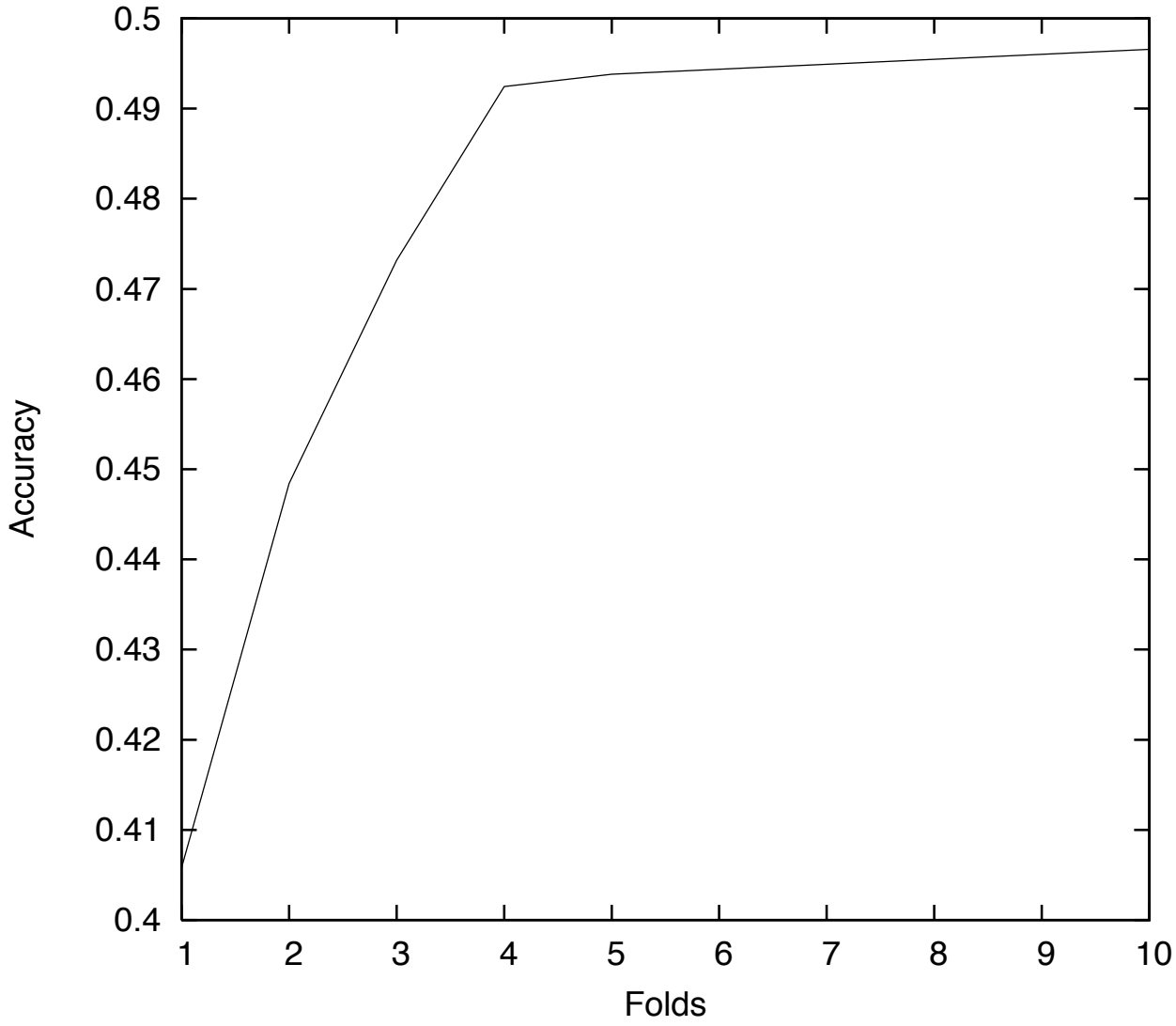
Evaluation

- Parse 10.000 sentences from Dutch Wikipedia of 5-15 words, and create logical forms.
- Generate from logical forms.
- Evaluate fluency ranking with 10 fold cross-validation:
  - Divide data in 10 parts
  - Evaluate 10 times
  - 9 folds for training, 1 fold for evaluation
- Best match accuracy: how often did the fluency component pick the best realization?

Results

Model	Best match accuracy
Random selection	0.012
N-gram language model	0.390
Maximum entropy model	0.522

Learning curve



Revisiting subject/direct object

- Surface-based model ranking:
  - [Marie]$_{obj1}$ bedriegt [Jan]$_{su}$
  - [Jan]$_{su}$ bedriegt [Marie]$_{obj1}$
- Maxent ranking:
  - [Jan]$_{su}$ bedriegt [Marie]$_{obj1}$
  - [Marie]$_{obj1}$ bedriegt [Jan]$_{su}$

Conslusion

- The Alpino parsing/generation system for Dutch: http://www.let.rug.nl/vannoord/alp/Alpino/
- Questions?

Chart generation algorithm outline

1. Initialize the agenda with lexical items that subsume parts of the goal semantics.
2. Process an item on the agenda:
   - Find all grammar rules that have the category of the current item as a daughter. Try to fill in all daughters with this item and (if required) another item from the chart where the combined semantics subsume (parts of) the goal semantics. Put all possible completions of rules with this item as items on the agenda.
   - Put the item on the chart.
3. Repeat step 2 until the agenda is empty.
4. Find all items with the start symbol, having the goal semantics.

Chart generation (example)

- *John ran fast*
- run(r), past(r), fast(r), arg1(r,j), name(j,John)

Chart generation (example)

Lexicon:

Word	Category	Semantics
John	np(x)	x: name(x,John)
ran	vp(x,y)	run(x), past(x), arg1(x,y)
fast	adv(x)	fast(x)
quickly	adv(x)	fast(x)

Grammar:

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

N	Surface	Cat	Sem
1	John	np(j)	name(j,John)
2	ran	vp(r,y)	run(r), past(r), arg1(r,x)
3	fast	adv(x)	fast(x)
4	quickly	adv(x)	fast(x)

N	Surface	Cat	Sem
1	John	np(j)	name(j,John)
2	ran	vp(r,y)	run(r), past(r), arg1(r,x)
3	fast	adv(x)	fast(x)
4	quickly	adv(x)	fast(x)
5 (1,2)	John ran	s(r)	run(r), past(r), arg1(r,j), name(j,John)

N	Surface	Cat	Sem
1	John	np(j)	name(j,John)
2	ran	vp(r,y)	run(r), past(r), arg1(r,x)
3	fast	adv(x)	fast(x)
4	quickly	adv(x)	fast(x)
5 (1,2)	John ran	s(r)	run(r), past(r), arg1(r,j), name(j,John)
6 (2,3)	ran fast	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)

N	Surface	Cat	Sem
1	John	np(j)	name(j,John)
2	ran	vp(r,y)	run(r), past(r), arg1(r,x)
3	fast	adv(x)	fast(x)
4	quickly	adv(x)	fast(x)
5 (1,2)	John ran	s(r)	run(r), past(r), arg1(r,j), name(j,John)
6 (2,3)	ran fast	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)
7 (2,4)	ran quickly	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)

N	Surface	Cat	Sem
1	John	np(j)	name(j,John)
2	ran	vp(r,y)	run(r), past(r), arg1(r,x)
3	fast	adv(x)	fast(x)
4	quickly	adv(x)	fast(x)
5 (1,2)	John ran	s(r)	run(r), past(r), arg1(r,j), name(j,John)
6 (2,3)	ran fast	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)
7 (2,4)	ran quickly	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)
8 (1,6)	John ran fast	s(r)	run(r), past(r), arg1(r,j), name(j,John), fast(r)

N	Surface	Cat	Sem
1	John	np(j)	name(j,John)
2	ran	vp(r,y)	run(r), past(r), arg1(r,x)
3	fast	adv(x)	fast(x)
4	quickly	adv(x)	fast(x)
5 (1,2)	John ran	s(r)	run(r), past(r), arg1(r,j), name(j,John)
6 (2,3)	ran fast	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)
7 (2,4)	ran quickly	vp(r,y)	run(r), past(r), arg1(r,j), fast(r)
8 (1,6)	John ran fast	s(r)	run(r), past(r), arg1(r,j), name(j,John), fast(r)
9 (1,7)	Jon ran quickly	s(r)	run(r), past(r), arg1(r,j), name(j,John), fast(r)