Novel Methods in Natural Language Generation for Spoken Dialogue Systems

Ondřej Dušek

Supervisor: **Filip Jurčíček**
Institute of Formal and Applied Linguistics
Charles University, Prague

Ph.D. thesis defense
June 12, 2017

Outline

1. Introduction to the problem
2. Surface Realization
3. A*/Perceptron Sentence Planning
4. Sequence-to-sequence Generation
5. Context-aware extensions (user adaptation/entrainment)
6. Generating Czech
7. Conclusions

NLG in Spoken Dialogue Systems

- converting a meaning representation (dialogue acts, DAs) to a sentence

inform(name=X,eattype=restaurant,food=Italian,area=riverside)

↓

X is an Italian restaurant near the river.

NLG in Spoken Dialogue Systems

- converting a meaning representation (dialogue acts, DAs) to a sentence

inform(name=X,eattype=restaurant,food=Italian,area=riverside)

↓

X is an Italian restaurant near the river.

- DA = act type (*inform, request…*) + slots (attributes) + values

NLG in Spoken Dialogue Systems

- converting a meaning representation (dialogue acts, DAs) to a sentence

inform(name=X,eattype=restaurant,food=Italian,area=riverside)

↓

X is an Italian restaurant near the river.

- DA = act type (*inform, request…*) + slots (attributes) + values

NLG in Spoken Dialogue Systems

- converting a meaning representation (dialogue acts, DAs) to a sentence

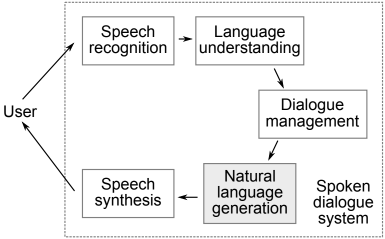
inform(name=X,eattype=restaurant,food=Italian,area=riverside)

↓

X is an Italian restaurant near the river.

  - DA = act type (*inform, request…*) + slots (attributes) + values

- input: from dialogue manager
- output: to TTS



Objectives

A) Create an NLG system easily adaptable for different domains

- fully trainable
- minimize required data annotation

Objectives

A) Create an NLG system easily adaptable for different domains
   - fully trainable
   - minimize required data annotation

B) Create an NLG system adaptable for different languages
   - we experiment with English and Czech

Objectives

A) Create an NLG system easily adaptable for different domains
- fully trainable
- minimize required data annotation

B) Create an NLG system adaptable for different languages
- we experiment with English and Czech

C) Create a generator that adapts to the user
- reuse users’ words/phrases – more natural

Objectives

A) Create an NLG system easily adaptable for different domains
   - fully trainable
   - minimize required data annotation

B) Create an NLG system adaptable for different languages
   - we experiment with English and Czech

C) Create a generator that adapts to the user
   - reuse users' words/phrases – more natural

D) Compare different NLG architectures
   - two-step pipeline / end-to-end joint setup

Objectives

A) Create an NLG system easily adaptable for different domains
- fully trainable
- minimize required data annotation

B) Create an NLG system adaptable for different languages
- we experiment with English and Czech

C) Create a generator that adapts to the user
- reuse users' words/phrases – more natural

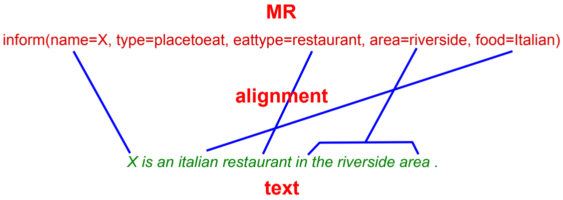
D) Compare different NLG architectures
- two-step pipeline / end-to-end joint setup

E) Create novel NLG datasets
- not many were available

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step


MR
inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)
alignment
X is an italian restaurant in the riverside area .
text

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly

MR

inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)

X is an italian restaurant in the riverside area .

text

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation

MR

inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)

X is an italian restaurant in the riverside area .

text

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
    - no error acummulation / manual annotation

Addressing data sparsity: Delexicalization

- Some/all slot values replaced with placeholders

inform(direction="Fulton Street", from_stop="Rockefeller Center", line=M11,
vehicle=bus, departure_time=11:02am)
Take line M11 bus at 11:02am from Rockefeller Center direction Fulton Street.

inform(name="La Mediterranée", good_for_meal=lunch, kids_allowed=no)
La Mediterranée is good for lunch and no children are allowed.

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation

Addressing data sparsity: Delexicalization

- Some/all slot values replaced with placeholders

inform(direction="Fulton Street", from_stop="Rockefeller Center", line=M11,
vehicle=bus, departure_time=11:02am)
Take line **M11 bus** at **11:02am** from **Rockefeller Center** direction **Fulton Street**.

inform(name="La Mediterranée", good_for_meal=lunch, kids_allowed=no)
La Mediterranée is good for **lunch** and no children are allowed.

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation

Addressing data sparsity: Delexicalization

- Some/all slot values replaced with placeholders

inform(direction="X-dir", from_stop="X-from", line=X-line,
    vehicle=X-vehicle, departure_time=X-departure)
Take line **X-line** **X-vehicle** at **X-departure** from **X-from** direction **X-dir**.

inform(name="X-name", good_for_meal=X-meal, kids_allowed=no)
X-name is good for **X-meal** and no children are allowed.

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation

Addressing data sparsity: Delexicalization

- Some/all slot values replaced with placeholders
- Different from full alignments – much easier to obtain

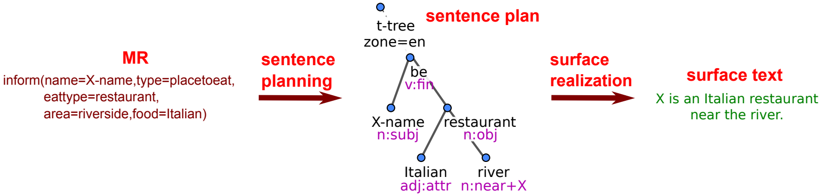
inform(direction=“X-dir”, from_stop=“X-from”, line=X-line,
vehicle=X-vehicle, departure_time=X-departure)
Take line **X-line X-vehicle** at **X-departure** from **X-from** direction **X-dir**.

inform(name=“X-name”, good_for_meal=X-meal, kids_allowed=no)
X-name is good for **X-meal** and no children are allowed.

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation



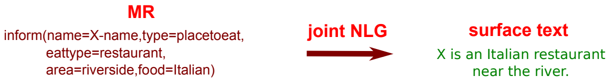
Pipeline / joint NLG

- traditional: sentence planning + surface realization

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

joint NLG

surface text

X is an Italian restaurant
near the river.

Pipeline / joint NLG

- traditional: sentence planning + surface realization
- newer: joint, end-to-end 1-step

Basic Ideas

Unaligned data

- earlier systems: manual alignments / preprocessing step
- we learn latent alignments jointly
  - no error acummulation / manual annotation

Addressing data sparsity: Delexicalization

- Some/all slot values replaced with placeholders
- Different from full alignments – much easier to obtain

Pipeline / joint NLG

- traditional: sentence planning + surface realization
- newer: joint, end-to-end 1-step
- we compare both, use *t-trees* as sentence plan

1. Introduction to the problem

2. Surface Realization

3. A*/Perceptron Sentence Planning

4. Sequence-to-sequence Generation

5. Context-aware extensions (user adaptation/entrainment)

6. Generating Czech

7. Conclusions

Our English Surface Realizer

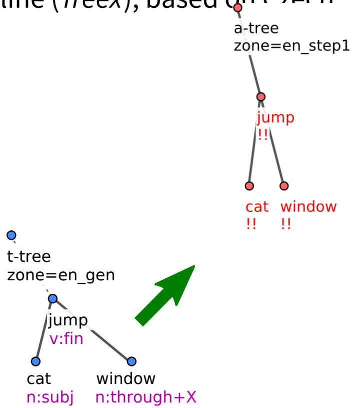
- Simple *t-tree* to text

Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech

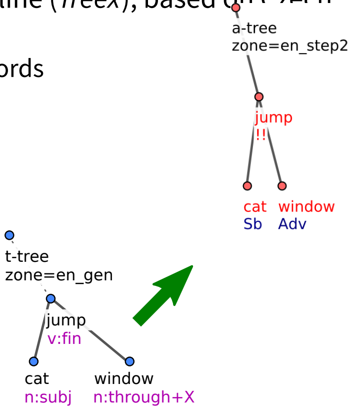
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
    - Copy t-tree



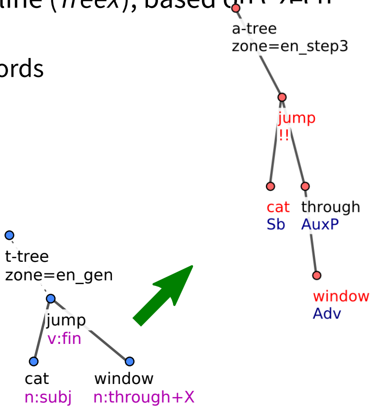
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
    - Copy t-tree
    - Add grammatical words



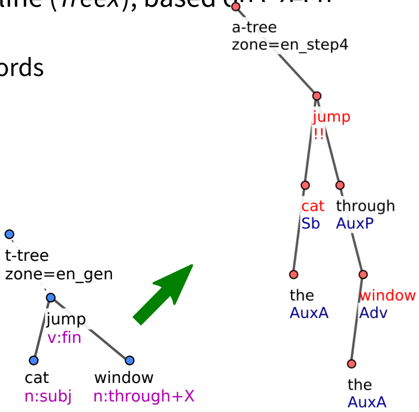
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words



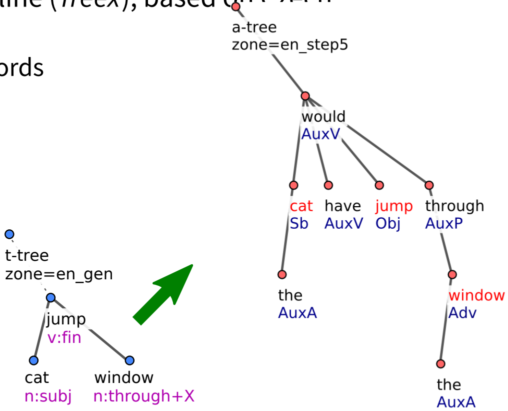
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words



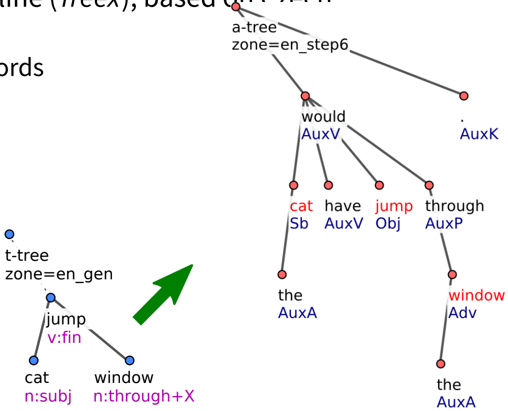
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words



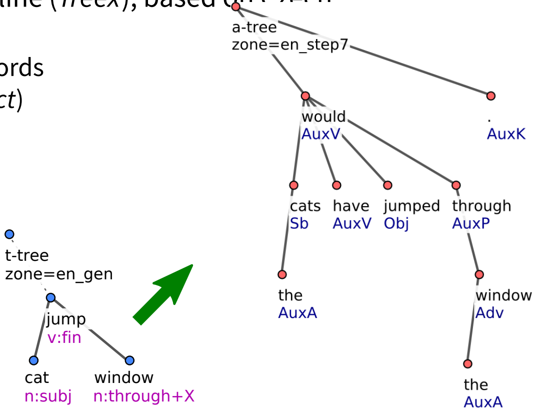
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words



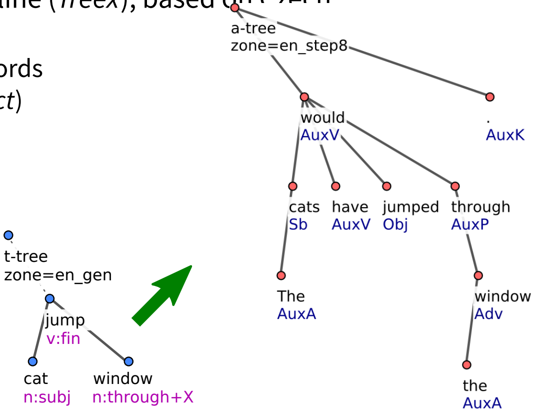
Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words
  - Word inflection (*Flect*)



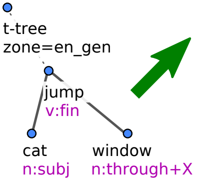
Our English Surface Realizer

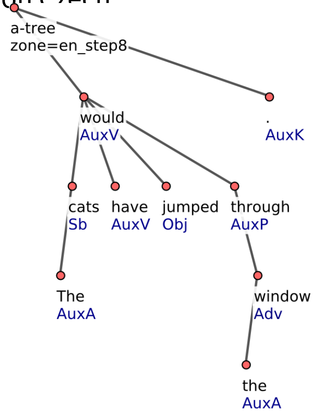
- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
    - Copy t-tree
    - Add grammatical words
    - Word inflection (*Flect*)



Our English Surface Realizer

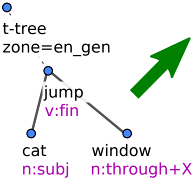
- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words
  - Word inflection (*Flect*)
- Used in our NLG systems & MT (*TectoMT*)

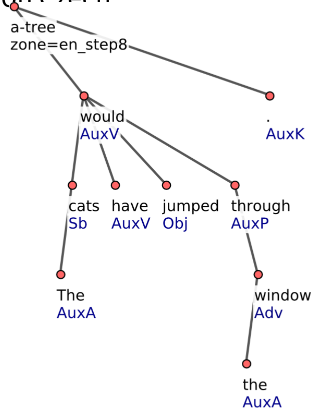




Our English Surface Realizer

- Simple *t-tree* to text
- Mostly rule-based pipeline (*Treex*), based on Czech
  - Copy t-tree
  - Add grammatical words
  - Word inflection (*Flect*)
- Used in our NLG systems & MT (*TectoMT*)





Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology

Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology
- Trained from corpora, generalizes to unseen words

Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology
- Trained from corpora, generalizes to unseen words
- Recast as multi-class classification

Wort

NN
Pl
Neut
Dat

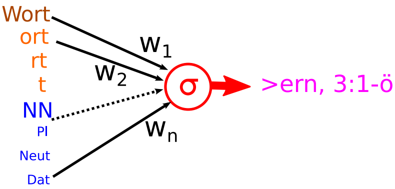
Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology
- Trained from corpora, generalizes to unseen words
- Recast as multi-class classification

Wort
ort
rt
t
NN
Pl
Neut
Dat

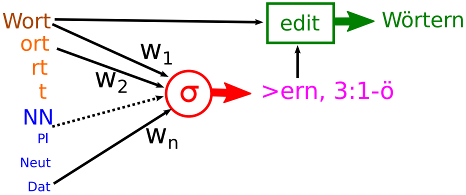
Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology
- Trained from corpora, generalizes to unseen words
- Recast as multi-class classification
    - Predict *edit script* (character diff lemma vs. form)



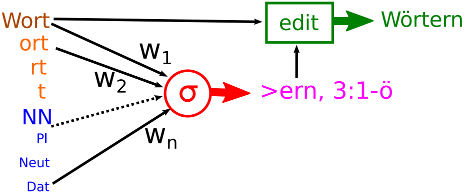
Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology
- Trained from corpora, generalizes to unseen words
- Recast as multi-class classification
    - Predict *edit script* (character diff lemma vs. form), then apply it



Flect – Statistical Word Inflection Generation

- Generate surface word form given lemma + morphology
- Trained from corpora, generalizes to unseen words
- Recast as multi-class classification
    - Predict *edit script* (character diff lemma vs. form), then apply it



- Evaluated on 6 languages, 96-99% accuracy

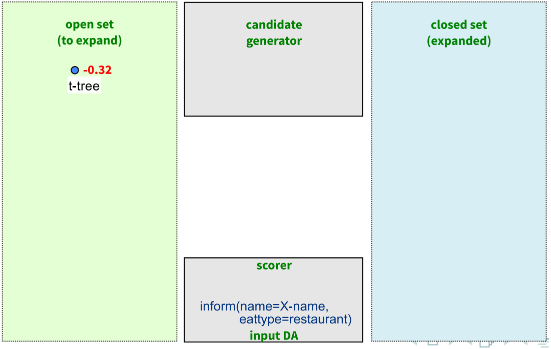
1. Introduction to the problem
2. Surface Realization
3. A*/Perceptron Sentence Planning
4. Sequence-to-sequence Generation
5. Context-aware extensions (user adaptation/entrainment)
6. Generating Czech
7. Conclusions

A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan

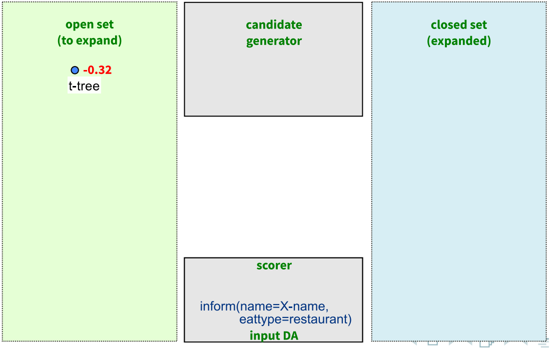
A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan



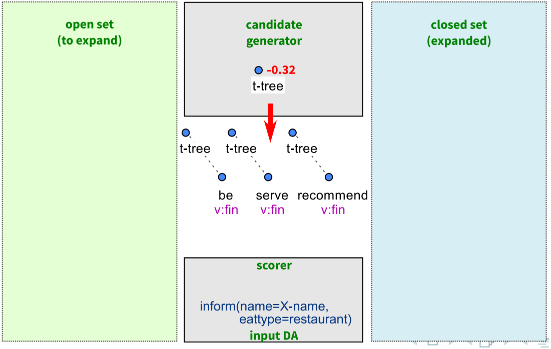
A*-Search/Perceptron Sentence Planner

- A*-style "path search": empty → full sentence plan
- Expanding candidate plan trees node-by-node



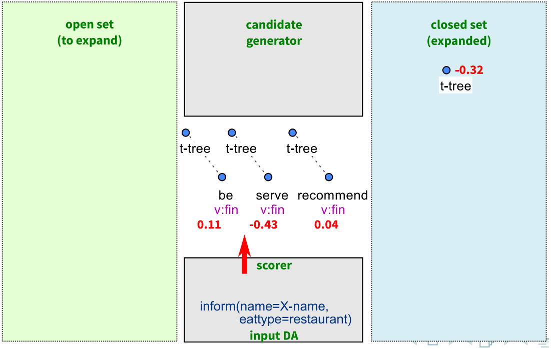
A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan
- Expanding candidate plan trees node-by-node



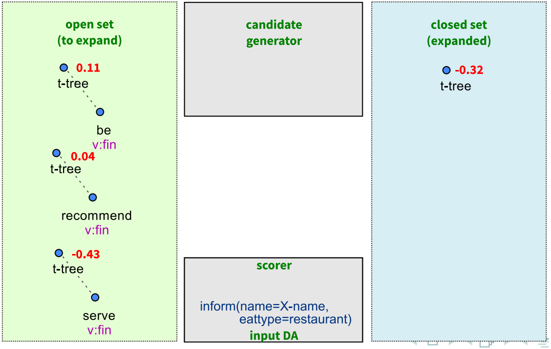
A*-Search/Perceptron Sentence Planner

- A*-style "path search": empty → full sentence plan
- Expanding candidate plan trees node-by-node
- Score = weights × features from tree & input DA



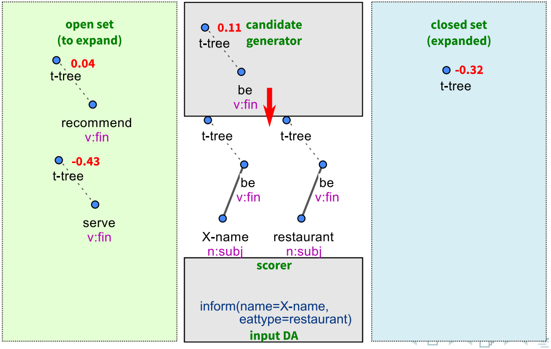
A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan
- Expanding candidate plan trees node-by-node
- Score = weights × features from tree & input DA



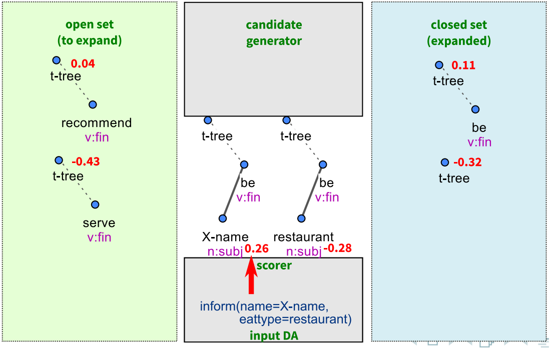
A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan
- Expanding candidate plan trees node-by-node
- Score = weights × features from tree & input DA



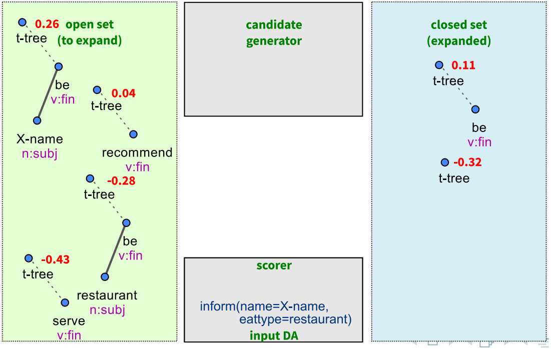
A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan
- Expanding candidate plan trees node-by-node
- Score = weights × features from tree & input DA



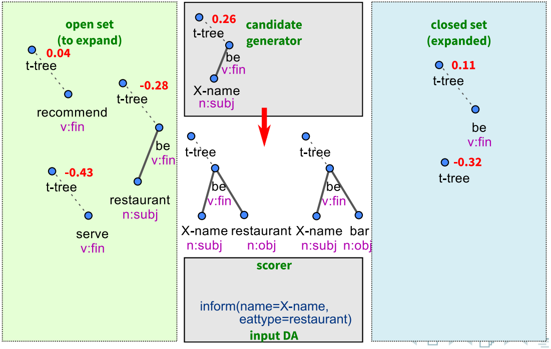
A*-Search/Perceptron Sentence Planner

- A*-style “path search”: empty → full sentence plan
- Expanding candidate plan trees node-by-node
- Score = weights × features from tree & input DA



A*-Search/Perceptron Sentence Planner

- A*-style "path search": empty → full sentence plan
- Expanding candidate plan trees node-by-node
- Score = weights × features from tree & input DA



A*/Perceptron Sentence Planning

Sentence Planner Details

- Output sentence plan processed by our realizer

A*/Perceptron Sentence Planning

Sentence Planner Details

- Output sentence plan processed by our realizer
- Perceptron ranker learning (Collins & Duffy, 2002)

A*/Perceptron Sentence Planning

Sentence Planner Details

- Output sentence plan processed by our realizer
- Perceptron ranker learning (Collins & Duffy, 2002)
- 1st NLG system learning from unaligned data

A*/Perceptron Sentence Planning

Sentence Planner Details

- Output sentence plan processed by our realizer
- Perceptron ranker learning (Collins & Duffy, 2002)
- 1st NLG system learning from unaligned data

Experiments

- BAGEL (404 sentences, restaurants) – 60% BLEU

A*/Perceptron Sentence Planning

Sentence Planner Details

- Output sentence plan processed by our realizer
- Perceptron ranker learning (Collins & Duffy, 2002)
- 1st NLG system learning from unaligned data

Experiments

- BAGEL (404 sentences, restaurants) – 60% BLEU
  - worse than orig. with alignments (67% BLEU) (Mairesse et al., 2010)

A*/Perceptron Sentence Planning

Sentence Planner Details

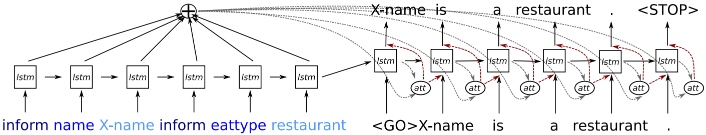
- Output sentence plan processed by our realizer
- Perceptron ranker learning (Collins & Duffy, 2002)
- 1st NLG system learning from unaligned data

Experiments

- BAGEL (404 sentences, restaurants) – 60% BLEU
  - worse than orig. with alignments (67% BLEU) (Mairesse et al., 2010)
- mostly fluent, but frequent errors (missed/added information)

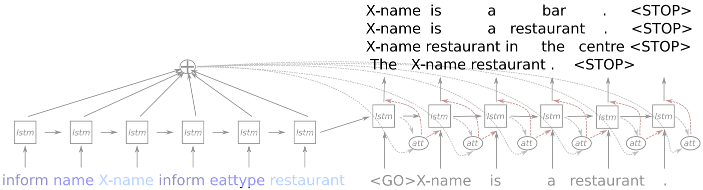
1. Introduction to the problem
2. Surface Realization
3. A*/Perceptron Sentence Planning
4. Sequence-to-sequence Generation
5. Context-aware extensions (user adaptation/entrainment)
6. Generating Czech
7. Conclusions

Sequence-to-sequence Generation: Our Model


X-name is a restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

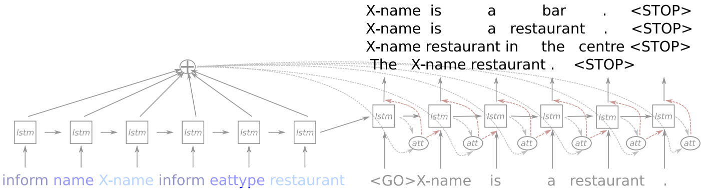
Sequence-to-sequence Generation: Our Model


X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

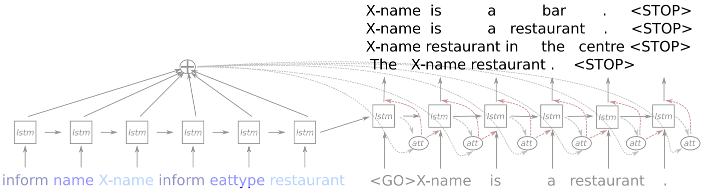
Sequence-to-sequence Generation: Our Model


X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs
+ *n*-best list **reranker**

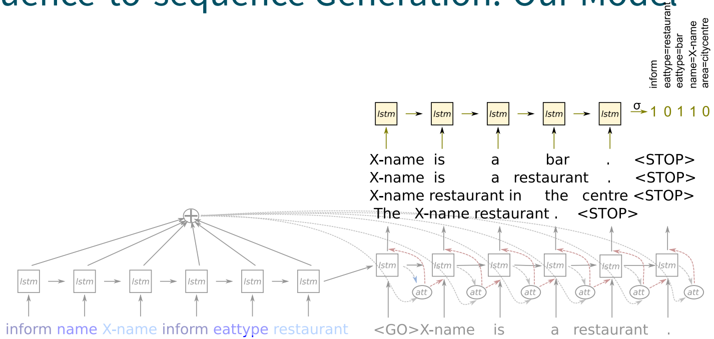
Sequence-to-sequence Generation: Our Model


X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs
+ *n*-best list **reranker**
  - to penalize missing/superfluous information in outputs

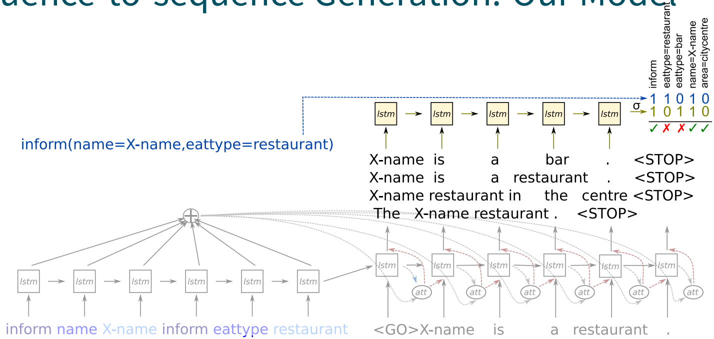
Sequence-to-sequence Generation: Our Model


inform
eattype=restaurant
eattype=bar
name=X-name
area=citycentre
lstm
σ
1 0 1 1 0
X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs
+ *n*-best list **reranker**
  - to penalize missing/superfluous information in outputs
  - classify DA from output

Sequence-to-sequence Generation: Our Model



inform(name=X-name,eattype=restaurant)

inform
eattype=restaurant
eattype=bar
name=X-name
area=citycentre

σ

1 1 0 1 0

1 0 1 1 0

✓ ✗ ✗ ✓ ✓

X-name is a bar . <STOP>

X-name is a restaurant . <STOP>

X-name restaurant in the centre <STOP>

The X-name restaurant . <STOP>

inform name X-name inform eattype restaurant

<GO> X-name is a restaurant .

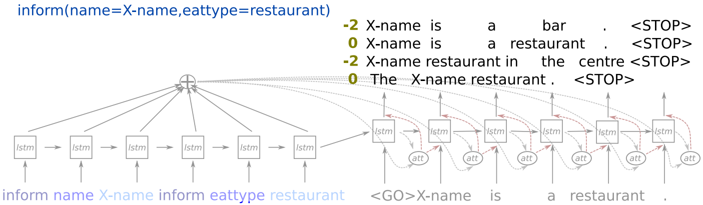
- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

+ *n*-best list **reranker**

  - to penalize missing/superfluous information in outputs
  - classify DA from output, compare to input DA

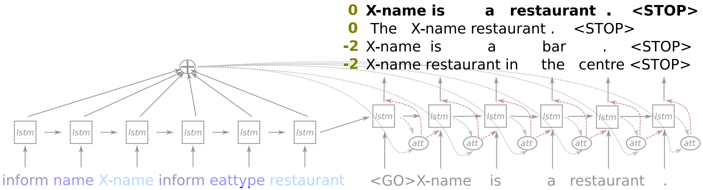
Sequence-to-sequence Generation: Our Model


inform(name=X-name,eattype=restaurant)
-2 X-name is a bar . <STOP>
0 X-name is a restaurant . <STOP>
-2 X-name restaurant in the centre <STOP>
0 The X-name restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs
+ *n*-best list **reranker**
    - to penalize missing/superfluous information in outputs
    - classify DA from output, compare to input DA

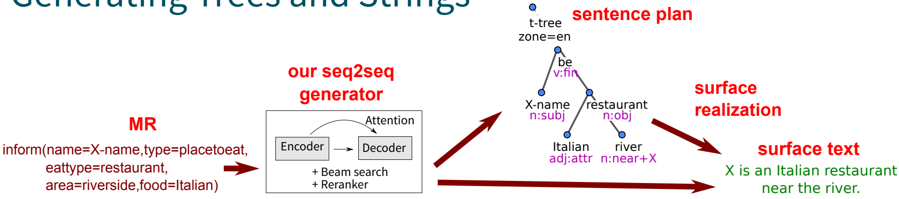
Sequence-to-sequence Generation: Our Model


0 X-name is a restaurant . <STOP>
0 The X-name restaurant . <STOP>
-2 X-name is a bar . <STOP>
-2 X-name restaurant in the centre <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs
+ *n*-best list **reranker**
    - to penalize missing/superfluous information in outputs
    - classify DA from output, compare to input DA

Generating Trees and Strings



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

our seq2seq generator

+ Beam search
+ Reranker

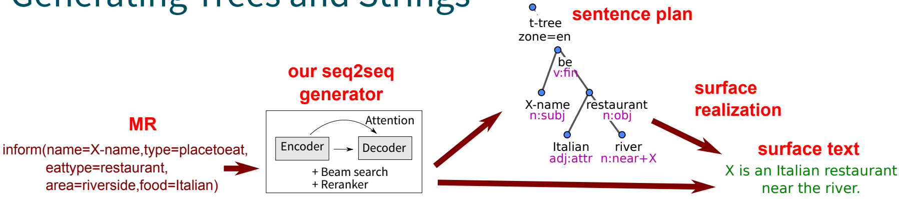
sentence plan

surface realization

surface text

X is an Italian restaurant near the river.

Generating Trees and Strings



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

our seq2seq generator

Encoder
Decoder
Attention
+ Beam search
+ Reranker

sentence plan

t-tree
zone=en
be
v:fin
X-name
n:subj
restaurant
n:obj
Italian
adj:attr
river
n:near+X

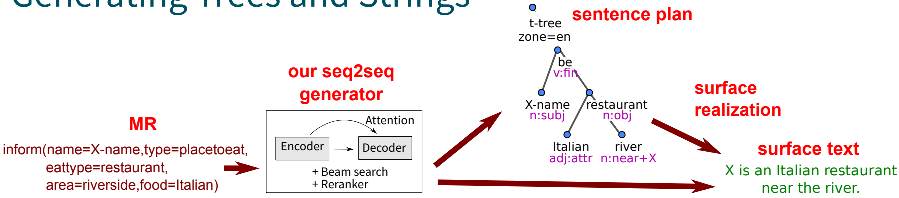
surface realization

surface text

X is an Italian restaurant
near the river.

- input: tokenized DAs

Generating Trees and Strings



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

our seq2seq generator

+ Beam search
+ Reranker

sentence plan

surface realization

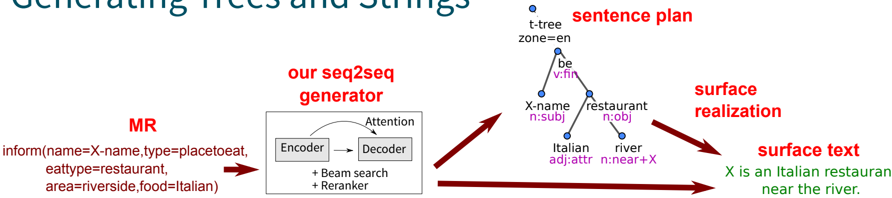
surface text

X is an Italian restaurant near the river.

- input: tokenized DAs
- output – 2 modes:

  joint mode – sentences

Generating Trees and Strings


MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

our seq2seq generator

+ Beam search
+ Reranker

sentence plan

surface realization

surface text

X is an Italian restaurant near the river.

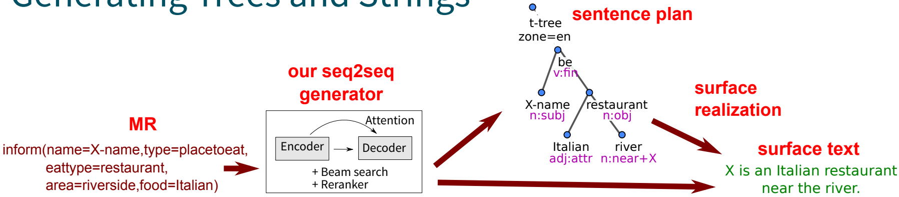
- input: tokenized DAs
- output – 2 modes:

  joint mode – sentences

  2-step mode – t-trees, in bracketed format (→ surface realizer)

(<root> <root> ((X-name n:subj) be v:fin ((Italian adj:attr) restaurant n:obj (river n:near+X))))

Generating Trees and Strings



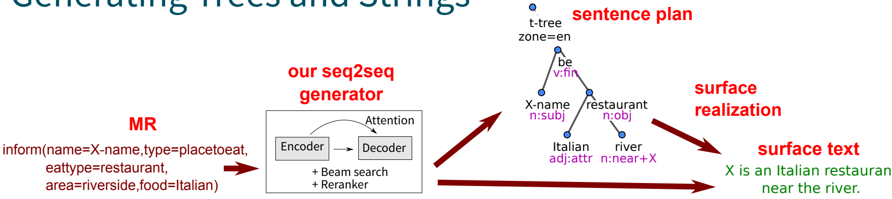
- input: tokenized DAs
- output – 2 modes:

  joint mode – sentences

  2-step mode – t-trees, in bracketed format (→ surface realizer)

- BAGEL – joint mode better:
  - BLEU joint 63% vs. trees 60%, same # of semantic errors

Generating Trees and Strings



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

our seq2seq generator

+ Beam search
+ Reranker

sentence plan

surface realization

surface text

X is an Italian restaurant near the river.

- input: tokenized DAs
- output – 2 modes:

  joint mode – sentences
  2-step mode – t-trees, in bracketed format (→ surface realizer)

- BAGEL – joint mode better:
  - BLEU joint 63% vs. trees 60%, same # of semantic errors
  - best without alignments (Mairesse et al. 2010: 67% BLEU)

1. Introduction to the problem
2. Surface Realization
3. A*/Perceptron Sentence Planning
4. Sequence-to-sequence Generation
5. Context-aware extensions (user adaptation/entrainment)
6. Generating Czech
7. Conclusions

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Our Seq2seq System & Entrainment

- Aim: condition generation on preceding context

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Our Seq2seq System & Entrainment

- Aim: condition generation on preceding context
  - data sparsity → just preceding utterance (biggest impact)

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Our Seq2seq System & Entrainment

- Aim: condition generation on preceding context
  - data sparsity → just preceding utterance (biggest impact)
- Context-aware data: new set collected via crowdsourcing

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Our Seq2seq System & Entrainment

- Aim: condition generation on preceding context
  - data sparsity → just preceding utterance (biggest impact)
- Context-aware data: new set collected via crowdsourcing
- Instance = DA + sentence

inform(from_stop="Fulton Street", vehicle=bus, direction="Rector Street",
departure_time=9:13pm, line=M21)

Go by the 9:13pm bus on the M21 line from Fulton Street directly to Rector Street

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Our Seq2seq System & Entrainment

- Aim: condition generation on preceding context
  - data sparsity → just preceding utterance (biggest impact)
- Context-aware data: new set collected via crowdsourcing
- Instance = DA + sentence + preceding utterance

NEW→*I'm headed to Rector Street*

inform(from_stop="Fulton Street", vehicle=bus, direction="Rector Street", departure_time=9:13pm, line=M21)

Go by the 9:13pm bus on the M21 line from Fulton Street directly to Rector Street

Entrainment in Trainable NLG: Data Needed

Entrainment in Dialogue

- speakers influenced by each other, reuse words & syntax
- natural, subconscious, helps success (Friedberg et al., 2012)
- NLG systems do not entrain (only limited, rule-based)

Our Seq2seq System & Entrainment

- Aim: condition generation on preceding context
  - data sparsity → just preceding utterance (biggest impact)
- Context-aware data: new set collected via crowdsourcing
- Instance = DA + context-aware sentence + preceding utterance

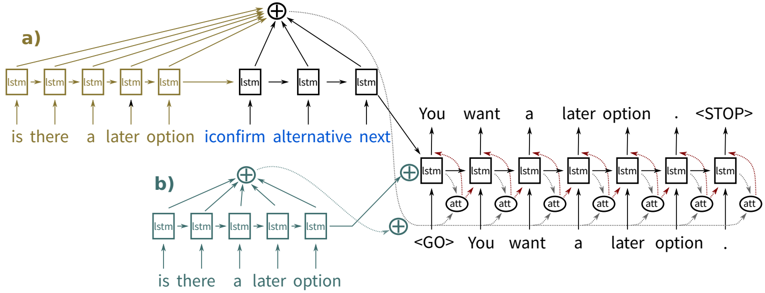
I'm headed to Rector Street

inform(from_stop="Fulton Street", vehicle=bus, direction="Rector Street",
departure_time=9:13pm, line=M21)

CONTEXT-AWARE → *Heading to Rector Street from Fulton Street, take a bus line M21 at 9:13pm.*

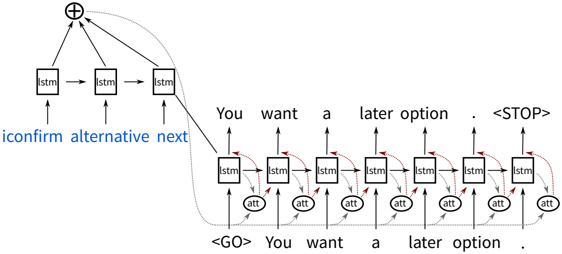
Context in our Seq2seq Generator

- Two direct context-aware extensions:


a)
lstm
is there a later option
iconfirm alternative next
b)
is there a later option
You want a later option . <STOP>
att
<GO> You want a later option .

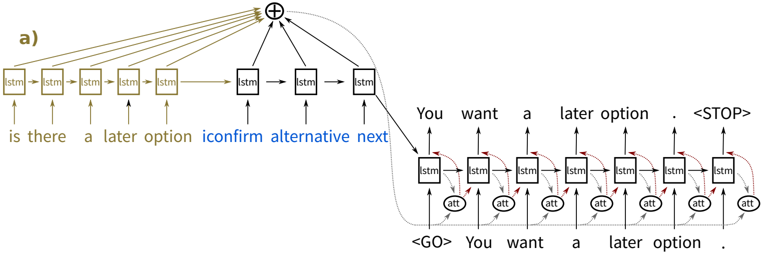
Context in our Seq2seq Generator

- Two direct context-aware extensions:



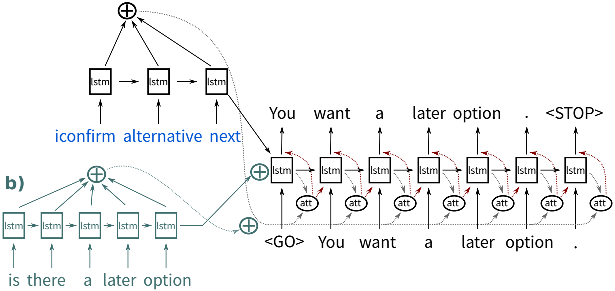
Context in our Seq2seq Generator

- Two direct context-aware extensions:
  a) preceding user utterance prepended to DA and fed into decoder


a)
lstm
att
⊕
is there a later option
iconfirm alternative next
You want a later option . <STOP>
<GO> You want a later option .

Context in our Seq2seq Generator

- Two direct context-aware extensions:
  a) preceding user utterance prepended to DA and fed into decoder
  b) separate context encoder, hidden states concatenated

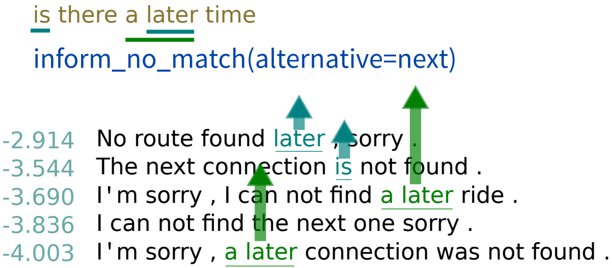


Context in our Seq2seq Generator

- Two direct context-aware extensions:
  - a) preceding user utterance prepended to DA and fed into decoder
  - b) separate context encoder, hidden states concatenated
- (One more) reranker: *n*-gram match

Context in our Seq2seq Generator

- Two direct context-aware extensions:
  - a) preceding user utterance prepended to DA and fed into decoder
  - b) separate context encoder, hidden states concatenated
- (One more) reranker: *n*-gram match
  - promote outputs having word/phrase overlap with context



is there a later time

inform_no_match(alternative=next)

-2.914	No route found later , sorry .
-3.544	The next connection is not found .
-3.690	I ' m sorry , I can not find a later ride .
-3.836	I can not find the next one sorry .
-4.003	I ' m sorry , a later connection was not found .

Context in our Seq2seq Generator

- Two direct context-aware extensions:
  a) preceding user utterance prepended to DA and fed into decoder
  b) separate context encoder, hidden states concatenated
- (One more) reranker: *n*-gram match
  - promote outputs having word/phrase overlap with context
- Evaluation (our set, 5.5k instances, public transport)

Context in our Seq2seq Generator

- Two direct context-aware extensions:
  a) preceding user utterance prepended to DA and fed into decoder
  b) separate context encoder, hidden states concatenated
- (One more) reranker: *n*-gram match
  - promote outputs having word/phrase overlap with context
- Evaluation (our set, 5.5k instances, public transport)
  - a) or b) + reranker best (66→69% BLEU)

Context in our Seq2seq Generator

- Two direct context-aware extensions:
  a) preceding user utterance prepended to DA and fed into decoder
  b) separate context encoder, hidden states concatenated
- (One more) reranker: *n*-gram match
  - promote outputs having word/phrase overlap with context
- Evaluation (our set, 5.5k instances, public transport)
  - a) or b) + reranker best (66→69% BLEU)
  - a) + reranker preferred by humans to baseline (52.5% cases, slight but significant)

1. Introduction to the problem

2. Surface Realization

3. A*/Perceptron Sentence Planning

4. Sequence-to-sequence Generation

5. Context-aware extensions (user adaptation/entrainment)

6. Generating Czech

7. Conclusions

Generating Czech

Motivation

- Statistical NLG tested almost exclusively on English

Generating Czech

Motivation

- Statistical NLG tested almost exclusively on English
  - no proper name inflection → easy delexicalization
  - little morphology, smaller lexicon

Generating Czech

Motivation

- Statistical NLG tested almost exclusively on English
  - no proper name inflection → easy delexicalization
  - little morphology, smaller lexicon

→ Czech is good choice (morphology, noun inflection)

Generating Czech

Motivation

- Statistical NLG tested almost exclusively on English
  - no proper name inflection → easy delexicalization
  - little morphology, smaller lexicon

→ Czech is good choice (morphology, noun inflection)

Czech NLG Data

- Virtually no non-English NLG datasets available

Generating Czech

Motivation

- Statistical NLG tested almost exclusively on English
  - no proper name inflection → easy delexicalization
  - little morphology, smaller lexicon

→ Czech is good choice (morphology, noun inflection)

Czech NLG Data

- Virtually no non-English NLG datasets available
- Crowdsourcing not usable → translating an English set (restaurants, Wen et al. 2015)

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected

Czech: Inflecting Proper Names

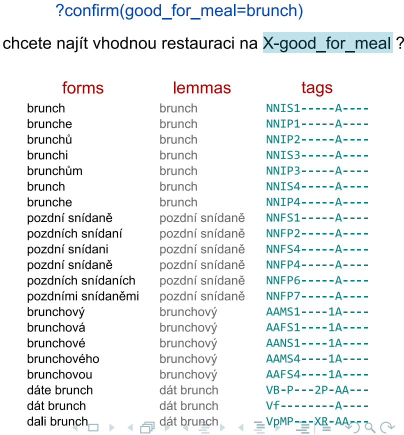
- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    ('lunch' as noun/verb)

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    (‘lunch’ as noun/verb)



?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    (‘lunch’ as noun/verb)
- Two baselines:

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
    - e.g., *obědvat* vs. *oběd*
      ('lunch' as noun/verb)
- Two baselines:
    a) random form

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    ('lunch' as noun/verb)
- Two baselines:
  a) random form
  b) most frequent form

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    ('lunch' as noun/verb)
- Two baselines:
  a) random form
  b) most frequent form
- Two LM-based approaches:

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    ('lunch' as noun/verb)
- Two baselines:
  a) random form
  b) most frequent form
- Two LM-based approaches:
  c) *n*-gram LM

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    ('lunch' as noun/verb)
- Two baselines:
  a) random form
  b) most frequent form
- Two LM-based approaches:
  c) *n*-gram LM
  d) RNN LM

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Czech: Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
    ('lunch' as noun/verb)
- Two baselines:
  a) random form
  b) most frequent form
- Two LM-based approaches:
  c) *n*-gram LM
  d) RNN LM
  - score options
    & select most probable

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

forms	lemmas	tags
brunch	brunch	NNIS1-----A----
brunche	brunch	NNIP1-----A----
brunchů	brunch	NNIP2-----A----
brunchi	brunch	NNIS3-----A----
brunchům	brunch	NNIP3-----A----
brunch	brunch	NNIS4-----A----
brunche	brunch	NNIP4-----A----
pozdní snídaně	pozdní snídaně	NNFS1-----A----
pozdních snídaní	pozdní snídaně	NNFP2-----A----
pozdní snídani	pozdní snídaně	NNFS4-----A----
pozdní snídaně	pozdní snídaně	NNFP4-----A----
pozdních snídaních	pozdní snídaně	NNFP6-----A----
pozdními snídaněmi	pozdní snídaně	NNFP7-----A----
brunchový	brunchový	AAMS1----1A----
brunchová	brunchový	AAFS1----1A----
brunchové	brunchový	AANS1----1A----
brunchového	brunchový	AAMS4----1A----
brunchovou	brunchový	AAFS4----1A----
dáte brunch	dát brunch	VB-P---2P-AA---
dát brunch	dát brunch	Vf--------A----
dali brunch	dát brunch	VpMP---XR-AA---

Generating Czech

Further Architecture Extensions

- Aimed at morphology

Generating Czech

Further Architecture Extensions

- Aimed at morphology

Evaluation

- BLEU & human (selected setups, WMT-style) on our dataset

Generating Czech

Further Architecture Extensions

- Aimed at morphology

Evaluation

- BLEU & human (selected setups, WMT-style) on our dataset
- Success, mostly good Czech

Generating Czech

Further Architecture Extensions

- Aimed at morphology

Evaluation

- BLEU & human (selected setups, WMT-style) on our dataset
- Success, mostly good Czech
- RNN lexicalization helps (better than baselines or *n*-grams)

Generating Czech

Further Architecture Extensions

- Aimed at morphology

Evaluation

- BLEU & human (selected setups, WMT-style) on our dataset
- Success, mostly good Czech
- RNN lexicalization helps (better than baselines or *n*-grams)
- Other extensions do not help

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL’15, ACL’16)

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL’15, ACL’16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT’15)
- Morphology generation (ACL-SRW’13)

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)
- Morphology generation (ACL-SRW'13)
- Seq2seq system adapted for Czech

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL’15, ACL’16)
- no need for fine-grained alignments

B✓ adapt easily to a different language
- English surface realizer from t-trees (WMT’15)
- Morphology generation (ACL-SRW’13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL’16)

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)
- Morphology generation (ACL-SRW'13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL'16)

- entrainment: generation conditioned on user utterances

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)
- Morphology generation (ACL-SRW'13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL'16)

- entrainment: generation conditioned on user utterances

D✓ show a comparison of different architectures (ACL'16)

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL’15, ACL’16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT’15)
- Morphology generation (ACL-SRW’13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL’16)

- entrainment: generation conditioned on user utterances

D✓ show a comparison of different architectures (ACL’16)

- generating strings / trees

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)
- Morphology generation (ACL-SRW'13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL'16)

- entrainment: generation conditioned on user utterances

D✓ show a comparison of different architectures (ACL'16)

- generating strings / trees

E✓ make novel datasets available

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)
- Morphology generation (ACL-SRW'13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL'16)

- entrainment: generation conditioned on user utterances

D✓ show a comparison of different architectures (ACL'16)

- generating strings / trees

E✓ make novel datasets available

- entrainment (with user utterances) (RE-WOCHAT'16)

Objectives: Our NLG systems…

A✓ adapt easily to different domains (ACL'15, ACL'16)

- no need for fine-grained alignments

B✓ adapt easily to a different language

- English surface realizer from t-trees (WMT'15)
- Morphology generation (ACL-SRW'13)
- Seq2seq system adapted for Czech

C✓ adapt to the user (SIGDIAL'16)

- entrainment: generation conditioned on user utterances

D✓ show a comparison of different architectures (ACL'16)

- generating strings / trees

E✓ make novel datasets available

- entrainment (with user utterances) (RE-WOCHAT'16)
- Czech

Thank you for your attention

Download my work

- Word Inflection Generator Code: bit.ly/flect
- A*+Seq2seq Generator Code: bit.ly/tgen_nlg
- Entrainment dataset: bit.ly/nlgdata
- Czech restaurant dataset: bit.ly/cs_rest

Contact me

Ondřej Dušek

odusek@ufal.mff.cuni.cz

References

Bahdanau, D. et al. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR*

Collins, M. & Duffy, N. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. *ACL*

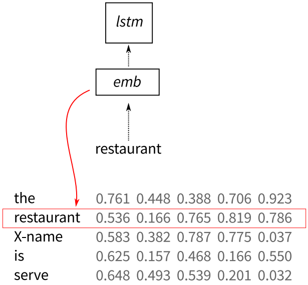
Friedberg, H. et al. 2012. Lexical entrainment and success in student engineering groups. *SLT*

Mairesse, F. et al. 2010. Phrase-based statistical language generation using graphical models and active learning. *ACL*

Wen, T. H. et al. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. *EMNLP*

Embeddings

- function: words $\rightarrow \mathbb{R}^n$
- equiv. to 1-hot encoding + fully connected layer
    - embedding values = weights in fully connected layer
- initialized randomly
- backpropagation during training
    - from output layer
    - through recurrent layers
    - to embedding layer



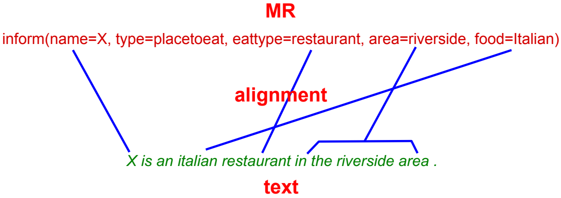
the	0.761	0.448	0.388	0.706	0.923
restaurant	0.536	0.166	0.765	0.819	0.786
X-name	0.583	0.382	0.787	0.775	0.037
is	0.625	0.157	0.468	0.166	0.550
serve	0.648	0.493	0.539	0.201	0.032

Detail: Generating from Unaligned Data

- earlier, NLG systems required:
  a) manual alignments
  b) alignment preprocessing step

Detail: Generating from Unaligned Data

- earlier, NLG systems required:
  - a) manual alignments
  - b) alignment preprocessing step


MR
inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)
alignment
X is an italian restaurant in the riverside area .
text

Detail: Generating from Unaligned Data

- earlier, NLG systems required:
  a) manual alignments
  b) alignment preprocessing step
- we learn alignments jointly

MR

inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)

X is an italian restaurant in the riverside area .

text

Detail: Generating from Unaligned Data

- earlier, NLG systems required:
  a) manual alignments
  b) alignment preprocessing step
- we learn alignments jointly
  - no error acummulation / manual annotation
  - alignment is latent (needs not be hard/1:1)

MR

inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)

X is an italian restaurant in the riverside area .

text

Detail: Generating from Unaligned Data

- earlier, NLG systems required:
  a) manual alignments
  b) alignment preprocessing step
- we learn alignments jointly
  - no error acummulation / manual annotation
  - alignment is latent (needs not be hard/1:1)

inform(name=X-name, type=placetoeat, **area=centre**, eattype=restaurant, near=X-near)
*The X restaurant is **conveniently** located near X, **right in the city center**.*

inform(name=X-name, type=placetoeat, **foodtype=Chinese_takeaway**)
*X serves **Chinese food** and has a **takeaway** possibility.*

inform(name=X-name, type=placetoeat, **pricerange=cheap**)
***Prices** at X are **quite cheap**.*

Detail: Delexicalization

- Way to address data sparsity

Detail: Delexicalization

- Way to address data sparsity
    - many slot values seen once or never in training

Detail: Delexicalization

- Way to address data sparsity
  - many slot values seen once or never in training
  - + they appear verbatim in the outputs
    - restaurant names, departure times

Detail: Delexicalization

- Way to address data sparsity
  - many slot values seen once or never in training
  - + they appear verbatim in the outputs
    - restaurant names, departure times
  - → replaced with placeholders for generation

Detail: Delexicalization

- Way to address data sparsity
  - many slot values seen once or never in training
  - + they appear verbatim in the outputs
    - restaurant names, departure times
  - → replaced with placeholders for generation
  - + added back in post-processing

Detail: Delexicalization

- Way to address data sparsity
  - many slot values seen once or never in training
  - + they appear verbatim in the outputs
    - restaurant names, departure times
  - → replaced with placeholders for generation
  - + added back in post-processing
- Still different from full semantic alignments
  - can be obtained by simple string replacement

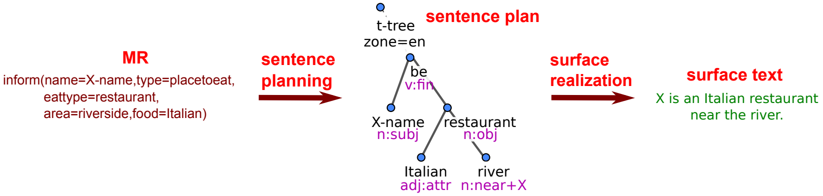
Detail: Delexicalization

- Way to address data sparsity
  - many slot values seen once or never in training
  - + they appear verbatim in the outputs
    - restaurant names, departure times
  - → replaced with placeholders for generation
  - + added back in post-processing
- Still different from full semantic alignments
  - can be obtained by simple string replacement
- Can be applied to some or all slots

enumerable:	food type, price range
non-enumerable:	restaurant name, phone number, postcode

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
  1. sentence planning – decide on the overall sentence structure
  2. surface realization – decide on specific word forms, linearize



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

sentence planning

sentence plan

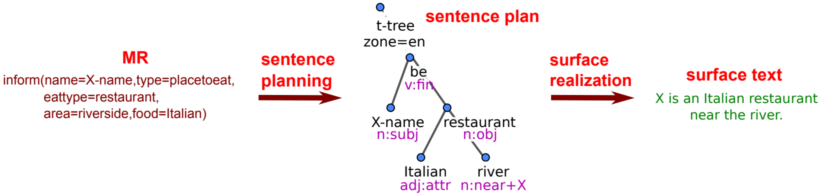
surface realization

surface text

X is an Italian restaurant
near the river.

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
  1. sentence planning – decide on the overall sentence structure
  2. surface realization – decide on specific word forms, linearize
- some NLG systems join this into a single step



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

sentence planning

sentence plan

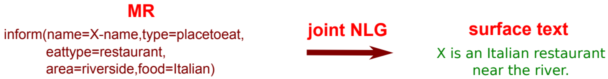
surface realization

surface text

X is an Italian restaurant
near the river.

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
  1. sentence planning – decide on the overall sentence structure
  2. surface realization – decide on specific word forms, linearize
- some NLG systems join this into a single step



MR

inform(name=X-name,type=placetoeat,
eattype=restaurant,
area=riverside,food=Italian)

joint NLG

surface text

X is an Italian restaurant
near the river.

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
    1. sentence planning – decide on the overall sentence structure
    2. surface realization – decide on specific word forms, linearize
- some NLG systems join this into a single step
- both aproaches have their merits

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
  1. sentence planning – decide on the overall sentence structure
  2. surface realization – decide on specific word forms, linearize
- some NLG systems join this into a single step
- both aproaches have their merits

  two-step: simpler structure generation (more abstract)

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
  1. sentence planning – decide on the overall sentence structure
  2. surface realization – decide on specific word forms, linearize
- some NLG systems join this into a single step
- both aproaches have their merits

two-step:	simpler structure generation (more abstract)
joint:	avoids error accumulation over a pipeline

Detail: Pipeline vs. Joint NLG

- NLG pipeline traditionally divided into:
  1. sentence planning – decide on the overall sentence structure
  2. surface realization – decide on specific word forms, linearize
- some NLG systems join this into a single step
- both aproaches have their merits

  two-step: simpler structure generation (more abstract)
  joint: avoids error accumulation over a pipeline
- we try both in one system + compare

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax

how bout the next ride

Sorry, I did not find a later option.

I'm sorry, the next ride was not found.

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success (Friedberg et al., 2012)

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success (Friedberg et al., 2012)
- typical NLG only takes the input DA into account

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success (Friedberg et al., 2012)
- typical NLG only takes the input DA into account
  - no way of adapting to user's way of speaking

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success (Friedberg et al., 2012)
- typical NLG only takes the input DA into account
  - no way of adapting to user’s way of speaking
- entrainment in NLG limited to rule-based systems so far

Detail: Entrainment

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success (Friedberg et al., 2012)
- typical NLG only takes the input DA into account
  - no way of adapting to user's way of speaking
- entrainment in NLG limited to rule-based systems so far
- our system is trainable and entrains/adapts

Detail: Why Multilingual NLG

- English: little morphology

Detail: Why Multilingual NLG

- English: little morphology
  - vocabulary size relatively small

Detail: Why Multilingual NLG

- English: little morphology
  - vocabulary size relatively small
  - (almost) no morphological agreement

Detail: Why Multilingual NLG

- English: little morphology
  - vocabulary size relatively small
  - (almost) no morphological agreement
  - no need to inflect proper names
  - → lexicalization = copy names from DA to output

Detail: Why Multilingual NLG

- English: little morphology
  - vocabulary size relatively small
  - (almost) no morphological agreement
  - no need to inflect proper names
  - → lexicalization = copy names from DA to output
- This does not work with rich morphology

			ě	é
Toto	se líbí	~~uživateli~~	Jana	Nováková.
This	*is liked by*	*user* [masc]	*(name)*	[fem]
		[dat]		[nom]

	e	u	
Děkujeme,	Jan	Novák,	vaše hlasování
Thank you,	*(name)*[nom]		bylo vytvořeno.
			your poll has been created

Detail: Why Multilingual NLG

- English: little morphology
  - vocabulary size relatively small
  - (almost) no morphological agreement
  - no need to inflect proper names

  → lexicalization = copy names from DA to output
- This does not work with rich morphology

  → Czech is a good language to try

Toto se líbí ~~uživateli~~ Jana ě Nováková é.

This is liked by user [masc] [dat] *(name)* [fem] [nom]

Děkujeme, Jan e Novák u, vaše hlasování bylo vytvořeno.

Thank you, (name) [nom] *your poll has been created*

Detail: Why Multilingual NLG

- English: little morphology
    - vocabulary size relatively small
    - (almost) no morphological agreement
    - no need to inflect proper names

    → lexicalization = copy names from DA to output
- This does not work with rich morphology

    → Czech is a good language to try
- Extensions to our generator to address this:
    - 3rd generator mode: generating lemmas & morphological tags

Detail: Why Multilingual NLG

- English: little morphology
    - vocabulary size relatively small
    - (almost) no morphological agreement
    - no need to inflect proper names
    - → lexicalization = copy names from DA to output
- This does not work with rich morphology
    - → Czech is a good language to try
- Extensions to our generator to address this:
    - 3rd generator mode: generating lemmas & morphological tags
    - inflection for lexicalization (surface form selection)

Surface Realizer in TectoMT

- BLEU scores for TectoMT translation within the QTLeap project

Task	Dutch-English		Czech-English	
	IT	news	IT	news
Phrase-based	25.57	23.50	19.03	24.03
TectoMT	27.09	19.40	20.53	13.04

Surface Realizer in TectoMT

(1) *Output:* One Council, how into that moment to do: carefully this page snatch and make from it bookmark.

Source: Jedna rada, jak se v tu chvíli zachovat: Opatrně tuhle stránku vytrhněte a udělejte si z ní záložku.

Reference: *A piece of advice on how to proceed at that moment: gently excise this page and make it your bookmark.*

(2) *Output:* Mr. Englund a historian is swedish and a journalist.

Source: Pan Englund je švédský historik a novinář.

Reference: *Mr. Englund is a Swedish historian and journalist.*

(3) *Output:* Their lives flikkeren as votiefkaarsen in a church; new is added to the altar other is been.

Source: Hun levens flikkeren als votiefkaarsen in een kerk; nieuwe worden toegevoegd aan het altaar terwijl andere worden uitgemaakt.

Reference: *Their lives flicker like votive candles in a church; new ones are added to the altar while others are put out.*

(4) *Output:* From the almost beginning, this is an inspiring book.

Source: Vrijwel vanaf het begin is dit een bezielend boek.

Reference: *Almost from the start, this is a moving book.*

Errors: source parsing, t-lemma translation, untranslated , formeme translation, article assignment, word ordering (transfer), word ordering (realizer), inflection (realizer)

Flect: The need for morphology in generation

- English – not so much:
  hard-coded solutions often work well enough

Flect: The need for morphology in generation

- English – not so much:
  hard-coded solutions often work well enough
- Languages with more inflection (e.g. Czech):
  even for the simplest things

Toto se líbí ~~uživateli~~ Jan~~a~~ě Novákov~~á~~é.
This is liked by user [masc] [dat] *(name)* [fem] [nom]

Děkujeme, Jan e Novák u, vaše hlasování bylo vytvořeno.
Thank you, (name) [nom] *your poll has been created*

Flect: The task at hand

word + NNS → words

Wort + $\text{NN}_{\text{Neut,Pl,Dat}}$ → Wörtern

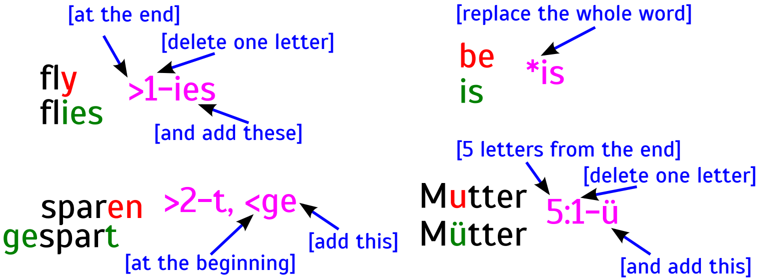
be + VBZ → is

ser + $\text{V}^{\text{gen=c,num=s,person=3,}}_{\text{mood=indicative,tense=present}}$ → es

- Input: Lemma (base form) or stem

  + morphological properties (POS, case, gender, etc.)
- Output: Inflected word form
- **Inverse to POS tagging**

Flect: Inflection patterns as multi-class classification



[at the end]
[delete one letter]
fly
flies
>1–ies
[and add these]

[replace the whole word]
be
is
*is

sparen
gespart
>2–t, <ge
[at the beginning]
[add this]

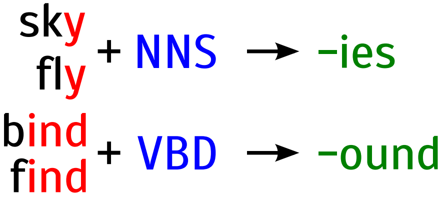
[5 letters from the end]
[delete one letter]
Mutter
Mütter
5:1–ü
[and add this]

Our inflection rules: *edit scripts*

- **A kind of diffs**: how to modify the lemma to get the form
- Based on Levenshtein distance

Flect: Features useful for morphology generation

- Same POS + same ending = (often) same inflection



sky

fly

+ NNS → –ies

bind

find

+ VBD → –ound

Flect: Features useful for morphology generation

- Same POS + same ending = (often) same inflection

sky
fly + NNS → -ies

bind
find + VBD → -ound

- **Suffixes = good features to generalize to unseen inputs**
- Machine learning should be able to deal with counter-examples

Flect: Features useful for morphology generation

- Same POS + same ending = (often) same inflection

sky
fly + NNS → –ies

bind
find + VBD → –ound

- **Suffixes = good features to generalize to unseen inputs**
- Machine learning should be able to deal with counter-examples
- **Capitalization: no influence on morphology**

Flect: Overall procedure

Wort

NN

Pl

Neut

Dat

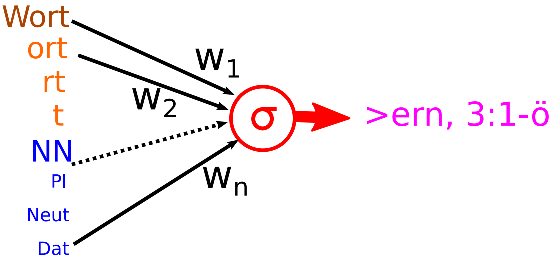
Flect: Overall procedure

1. Get **features** from lemma, POS, suffixes
   (+morph. properties & their combinations, possibly context)

Wort
ort
rt
t
NN
Pl
Neut
Dat

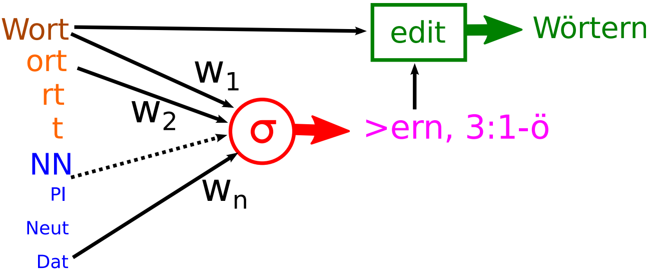
Flect: Overall procedure

1. Get **features** from lemma, POS, suffixes (+morph. properties & their combinations, possibly context)
2. Predict **edit scripts** using Logistic regression



Flect: Overall procedure

1. Get **features** from lemma, POS, suffixes (+morph. properties & their combinations, possibly context)
2. Predict **edit scripts** using Logistic regression
3. Use them as rules to obtain **form** from lemma

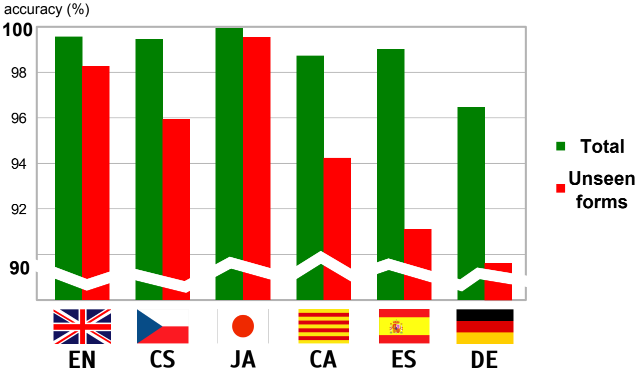


Testing *Flect* on 6 languages

- **CoNLL 2009 data**: varying morphology richness & tagsets

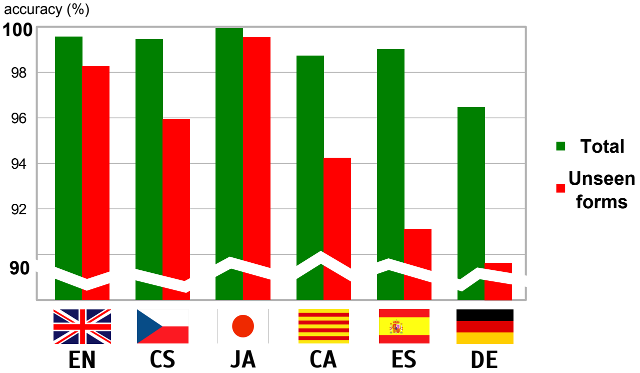
Testing *Flect* on 6 languages

- **CoNLL 2009 data**: varying morphology richness & tagsets



Testing *Flect* on 6 languages

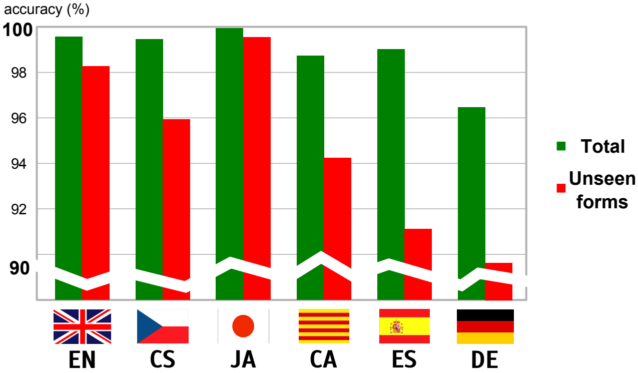
- **CoNLL 2009 data**: varying morphology richness & tagsets



- Works well even on unseen forms: suffixes help

Testing *Flect* on 6 languages

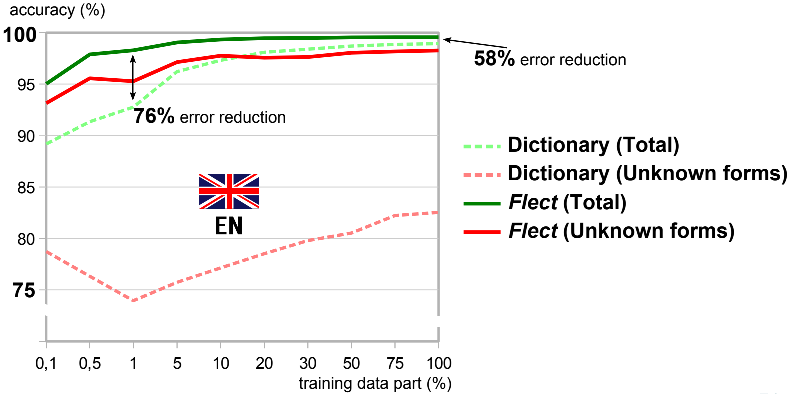
- **CoNLL 2009 data**: varying morphology richness & tagsets



- Works well even on unseen forms: suffixes help
  - over-generalization errors, e.g. torpedo + VBN = torpedone
  - German: syntax-sensitive morphology

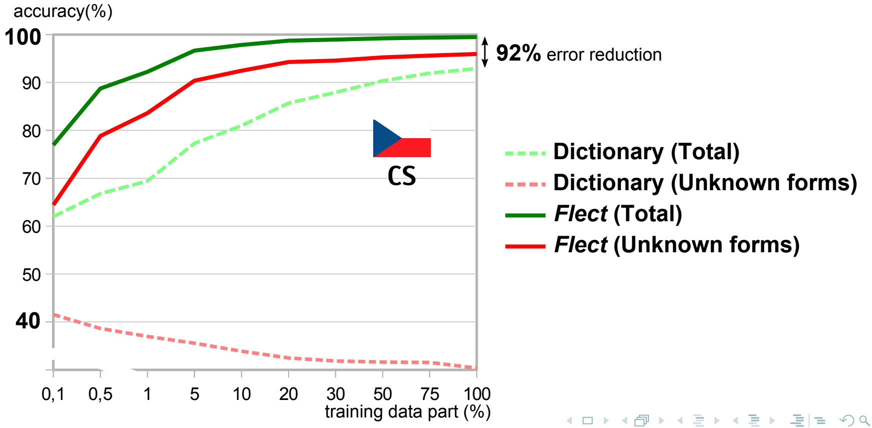
Flect vs. a dictionary from the same data

- English: Dictionary gets OK relatively soon



Flect vs. a dictionary from the same data

- English: Dictionary gets OK relatively soon
- Czech: Dictionary fails on unknown forms, our system works



Flect in English Surface Realization

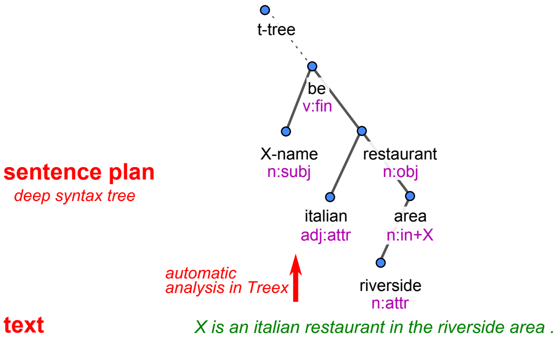
- TectoMT English Round-trip (PCEDT 2.0 Sect. 22+23)
  - analyzed and regenerated sentences compared to originals

Variant	**BLEU (%)**
Baseline (MorphoDiTa)	73.55
Flect alone	77.04
MorphoDiTa + Flect as a backoff	77.47

A*-search/Perceptron Sentence Planning

- Our generator learns alignments jointly
  - training from pairs: **MR + sentence**
  - with sentence planning (MR → deep syntax trees)

MR inform(name=X, type=placetoeat, eattype=restaurant, area=riverside, food=Italian)



A*/Perceptron: Overall workflow

A two-step setup:

A*/Perceptron: Overall workflow

MR

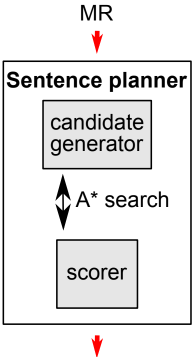
A two-step setup:

- *Input*: a meaning representation

A*/Perceptron: Overall workflow

A two-step setup:

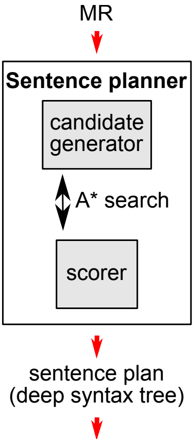
- *Input*: a meaning representation
- **1. – sentence planning**
    - statistical, our main focus
    - expanding + ranking candidate sentence plans
    - A*-like search



A*/Perceptron: Overall workflow

A two-step setup:

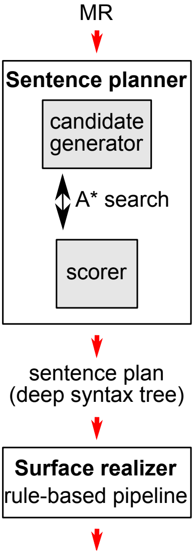
- *Input*: a meaning representation
- **1. – sentence planning**
  - statistical, our main focus
  - expanding + ranking candidate sentence plans
  - A*-like search
- *Intermediate*: sentence plan (deep syntax trees)



A*/Perceptron: Overall workflow

A two-step setup:

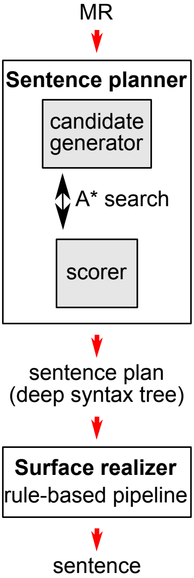
- *Input*: a meaning representation
- **1. – sentence planning**
  - statistical, our main focus
  - expanding + ranking candidate sentence plans
  - A*-like search
- *Intermediate*: sentence plan (deep syntax trees)
- **2. – surface realization**
  - reusing *Treex/TectoMT* realizer
  - (mostly) rule-based pipeline



A*/Perceptron: Overall workflow

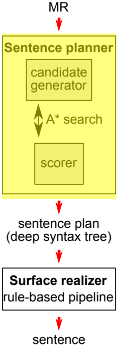
A two-step setup:

- *Input*: a meaning representation
- **1. – sentence planning**
  - statistical, our main focus
  - expanding + ranking candidate sentence plans
  - A*-like search
- *Intermediate*: sentence plan (deep syntax trees)
- **2. – surface realization**
  - reusing *Treex/TectoMT* realizer
  - (mostly) rule-based pipeline
- *Output*: plain text sentence



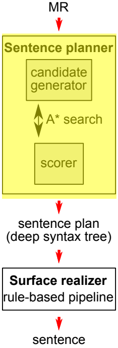
A*/Perceptron Sentence planner

- A*-style search
  - "finding the path" from empty tree to full sentence plan tree
  - expand the most promising candidate sentence plan in each step
  - stop when candidates don't improve for a while



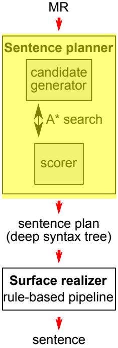
A*/Perceptron Sentence planner

- A*-style search
  - “finding the path” from empty tree to full sentence plan tree
  - expand the most promising candidate sentence plan in each step
  - stop when candidates don’t improve for a while
- Using two subcomponents:
  - **candidate generator**
    - churning out candidate sentence plan trees
    - given an incomplete candidate tree, add node(s)



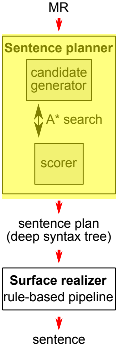
A*/Perceptron Sentence planner

- A*-style search
  - “finding the path” from empty tree to full sentence plan tree
  - expand the most promising candidate sentence plan in each step
  - stop when candidates don’t improve for a while
- Using two subcomponents:
  - **candidate generator**
    - churning out candidate sentence plan trees
    - given an incomplete candidate tree, add node(s)
  - **scorer**/ranker for the candidates
    - influences which candidate trees will be expanded



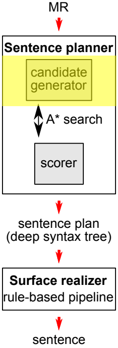
A*/Perceptron Sentence planner

- A*-style search
  - “finding the path” from empty tree to full sentence plan tree
  - expand the most promising candidate sentence plan in each step
  - stop when candidates don’t improve for a while
- Using two subcomponents:
  - **candidate generator**
    - churning out candidate sentence plan trees
    - given an incomplete candidate tree, add node(s)
  - **scorer**/ranker for the candidates
    - influences which candidate trees will be expanded
- Training data = MR + sentence plan tree pairs
  - trees obtained by automatic parsing in *Treex*



A*/Perceptron: Candidate generator

- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)


MR

Sentence planner

candidate generator

A* search

scorer

sentence plan (deep syntax tree)

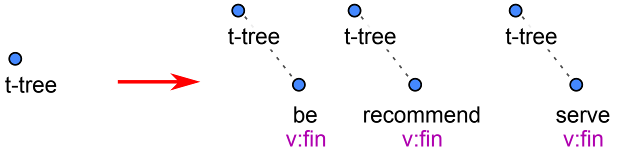
Surface realizer

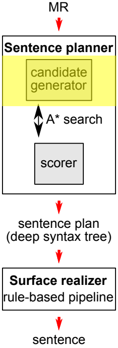
rule-based pipeline

sentence

A*/Perceptron: Candidate generator

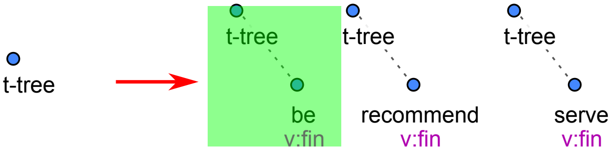
- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)

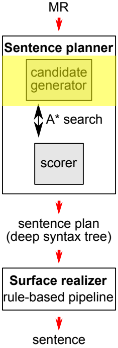




A*/Perceptron: Candidate generator

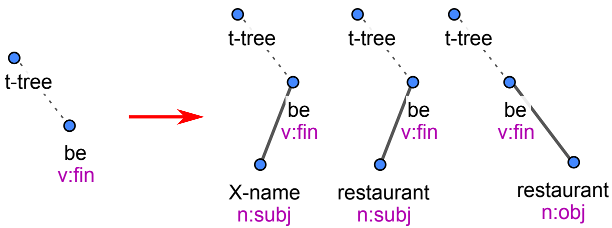
- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)

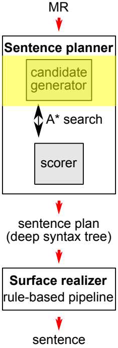




A*/Perceptron: Candidate generator

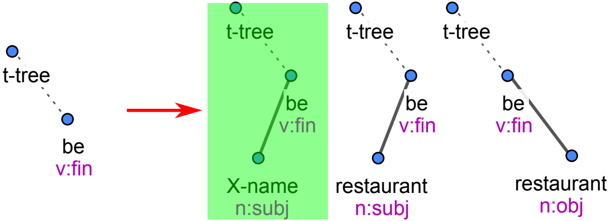
- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)

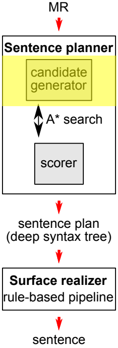




A*/Perceptron: Candidate generator

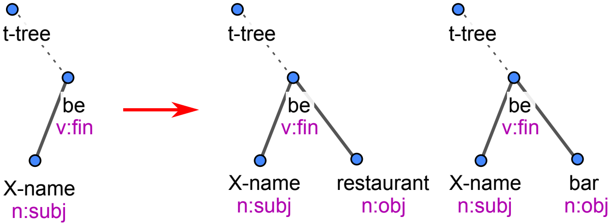
- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)

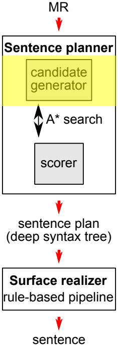




A*/Perceptron: Candidate generator

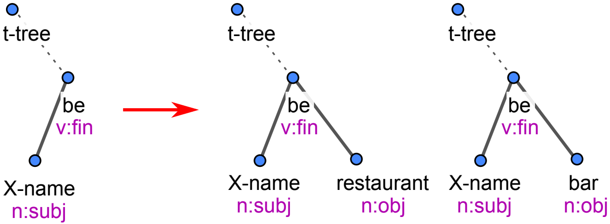
- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)



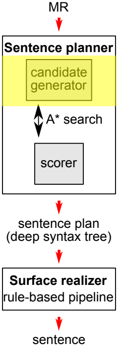


A*/Perceptron: Candidate generator

- Given a candidate plan tree, generate its successors by adding 1 node (at every possible place)



- Combinations explode even for small trees
- Limiting "possible places"
  - a few simple rules
  - based on context (elements of current MR, parent node)

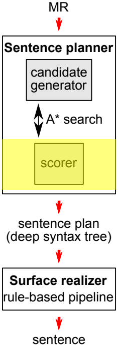


A*/Perceptron Sentence Planner: Scorer/Ranker

- a function:

  sentence plan tree + MR → real-valued score

  - describes the fitness of tree for MR



A*/Perceptron Sentence Planner: Scorer/Ranker

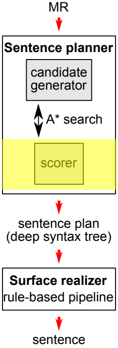
- a function:

  sentence plan tree + MR → real-valued score

  - describes the fitness of tree for MR

Linear perceptron scorer (Collins & Duffy, 2002)

- **score** = weights · features (from tree and MR)
  - features – elements of tree and MR
  - presence of nodes, slots, values + combination
  - tree size and shape, parent-child

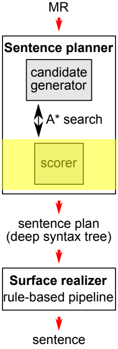


A*/Perceptron Sentence Planner: Scorer/Ranker

- a function:

  sentence plan tree + MR → real-valued score

  - describes the fitness of tree for MR



Linear perceptron scorer (Collins & Duffy, 2002)

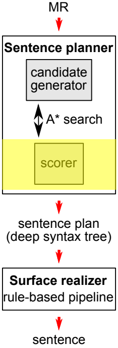
- **score** = weights · features (from tree and MR)
  - features – elements of tree and MR
  - presence of nodes, slots, values + combination
  - tree size and shape, parent-child
- **training** loop:
  - given MR, generate the best tree with current weights
  - update weights if generated tree ranks better than gold tree

A*/Perceptron Sentence Planner: Scorer/Ranker

- a function:

  sentence plan tree + MR $\rightarrow$ **real-valued score**

  - describes the fitness of tree for MR



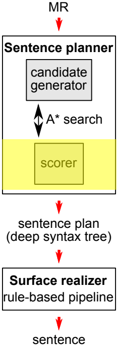
Linear perceptron scorer (Collins & Duffy, 2002)

- **score** = weights · features (from tree and MR)
  - features – elements of tree and MR
  - presence of nodes, slots, values + combination
  - tree size and shape, parent-child
- **training** loop:
  - given MR, generate the best tree with current weights
  - update weights if generated tree ranks better than gold tree
- **update** = $\alpha\cdot$ difference in features (gold$-$generated)
  - want gold to score better next time

A*/Perceptron Sentence Planner

Scoring problem

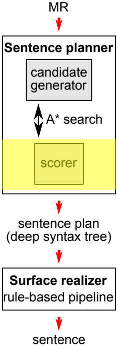
- Features are global over the whole sentence plan tree
  → bigger trees tend to score better



A*/Perceptron Sentence Planner

Scoring problem

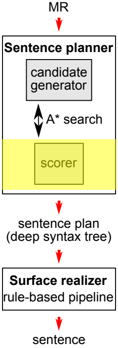
- Features are global over the whole sentence plan tree
  → bigger trees tend to score better
- But we score incomplete trees during the A* search
  - bigger incomplete trees are not always right
  - we need to promote “promising” incomplete trees



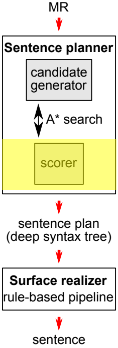
A*/Perceptron Sentence Planner

Scoring problem

- Features are global over the whole sentence plan tree
  → bigger trees tend to score better
- But we score incomplete trees during the A* search
  - bigger incomplete trees are not always right
  - we need to promote “promising” incomplete trees
- Scoring accuracy affects which paths are explored



A*/Perceptron Sentence Planner



Scoring problem

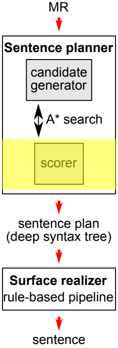
- Features are global over the whole sentence plan tree
  → bigger trees tend to score better
- But we score incomplete trees during the A* search
  - bigger incomplete trees are not always right
  - we need to promote “promising” incomplete trees
- Scoring accuracy affects which paths are explored

Our improvements to the scorer

- Differing tree updates
- Future promise

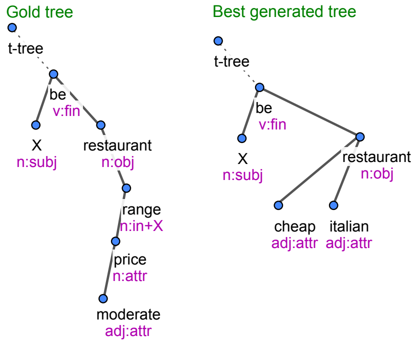
A*/Perceptron: Differing subtree updates

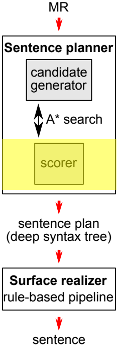
- Additional perceptron update
  - performed with the regular one
  - using pairs of differing subtrees of gold and generated tree (starting from common subtree)
  - promoting promising paths, demoting dead-ends



A*/Perceptron: Differing subtree updates

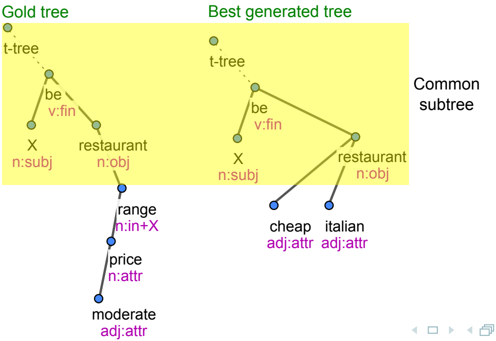
- Additional perceptron update
  - performed with the regular one
  - using pairs of differing subtrees of gold and generated tree (starting from common subtree)
  - promoting promising paths, demoting dead-ends

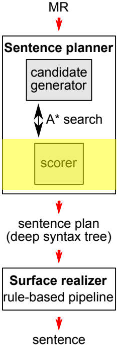




A*/Perceptron: Differing subtree updates

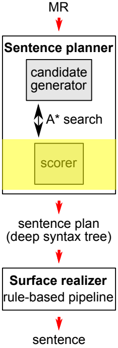
- Additional perceptron update
  - performed with the regular one
  - using pairs of differing subtrees of gold and generated tree (starting from common subtree)
  - promoting promising paths, demoting dead-ends

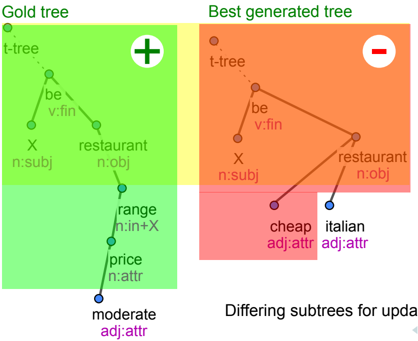




A*/Perceptron: Differing subtree updates

- Additional perceptron update
  - performed with the regular one
  - using pairs of differing subtrees of gold and generated tree (starting from common subtree)
  - promoting promising paths, demoting dead-ends

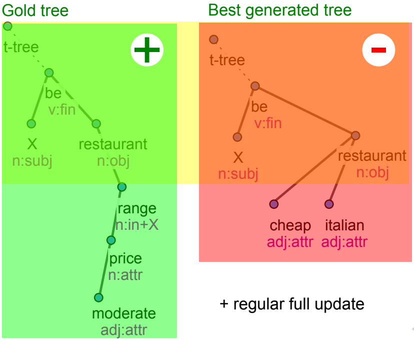




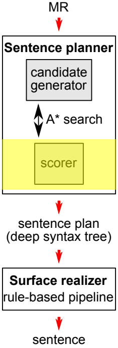
Differing subtrees for update

A*/Perceptron: Differing subtree updates

- Additional perceptron update
  - performed with the regular one
  - using pairs of differing subtrees of gold and generated tree (starting from common subtree)
  - promoting promising paths, demoting dead-ends

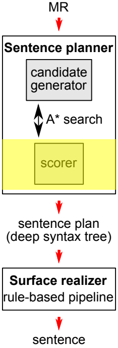


+ regular full update



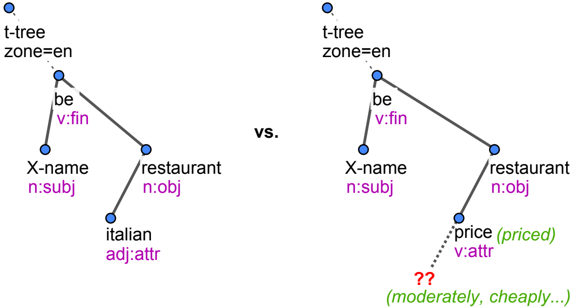
A*/Perceptron: Future promise estimate

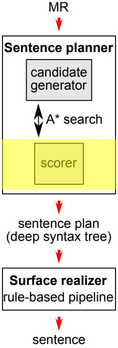
- Further score boost for incomplete trees



A*/Perceptron: Future promise estimate

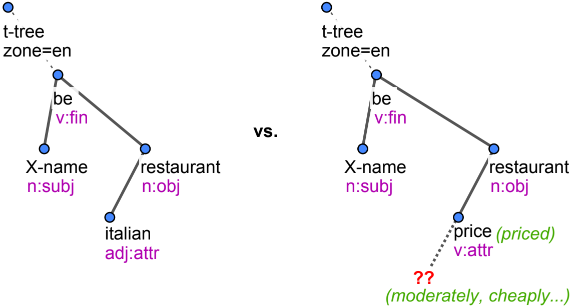
- Further score boost for incomplete trees
- Using the *expected number of children* of a node

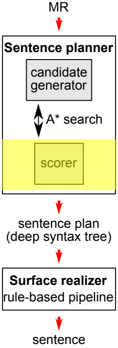




A*/Perceptron: Future promise estimate

- Further score boost for incomplete trees
- Using the *expected number of children* of a node





- **Future promise**:
  "how many children are missing to meet the expectation"
  - floored at zero, summed over the whole tree
- Added to scores, used to select next expansion path

A*/Perceptron Sentence Planner: Results

Setup	BLEU	NIST
perceptron scorer	54.24	4.643
+ differing subtree updates	58.70*	4.876
+ future promise	59.89*	5.231

- * both improvements statistically significant

A*/Perceptron Sentence Planner: Results

Setup	BLEU	NIST
perceptron scorer	54.24	4.643
+ differing subtree updates	58.70*	4.876
+ future promise	59.89*	5.231

- * both improvements statistically significant
- Overall, lower scores than Mairesse et al.'s ~ 67% BLEU

A*/Perceptron Sentence Planner: Results

Setup	BLEU	NIST
perceptron scorer	54.24	4.643
+ differing subtree updates	58.70*	4.876
+ future promise	59.89*	5.231

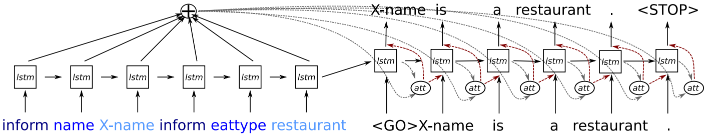
- * both improvements statistically significant
- Overall, lower scores than Mairesse et al.'s ~ 67% BLEU
- But our problem is harder:
  - we learn alignments jointly
  - our generator has to decide when to stop
    (whether all required information is included)

A*/Perceptron Example Outputs

Input DA	inform(name=X-name, type=placetoeat, pricerange=moderate, eattype=restaurant)
Reference	X is a restaurant that offers moderate price range.
Generated	X is a restaurant in the moderate price range.
Input DA	inform(name=X-name, type=placetoeat, area=X-area, pricerange=moderate, eattype=restaurant)
Reference	X is a moderately priced restaurant in X.
Generated	X is a restaurant in the X area.
Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, area=riverside, food=French)
Reference	X is a French restaurant on the riverside.
Generated	X is a French restaurant in the riverside area which serves French food.

- Mostly fluent and relevant
  - sometimes identical to reference, more often original
- Problems in some cases:
  - information missing / repeated / superfluous

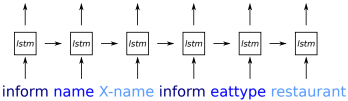
Sequence-to-sequence Generation: Our Model



X-name is a restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

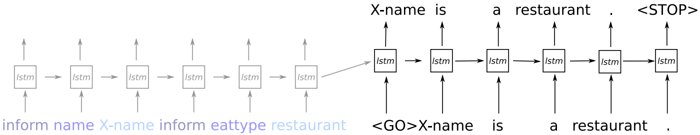
- Main generator: seq2seq with attention (Bahdanau et al., 2015)

Sequence-to-sequence Generation: Our Model



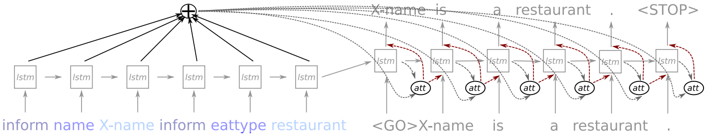
- Main generator: seq2seq with attention (Bahdanau et al., 2015)
    - Encoder LSTM RNN: encode DA into hidden states

Sequence-to-sequence Generation: Our Model


X-name is a restaurant . <STOP>
lstm
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)
  - Encoder LSTM RNN: encode DA into hidden states
  - Decoder LSTM RNN: generate output tokens

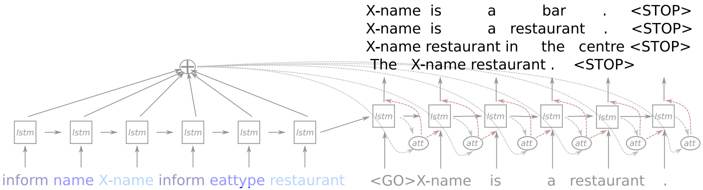
Sequence-to-sequence Generation: Our Model



X-name is a restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)
    - Encoder LSTM RNN: encode DA into hidden states
    - Decoder LSTM RNN: generate output tokens
    - attention model: weighing encoder hidden states

Sequence-to-sequence Generation: Our Model



X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>

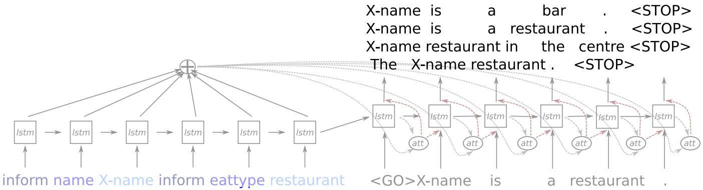
inform name X-name inform eattype restaurant

<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

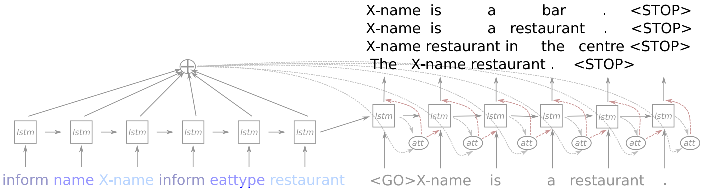
Sequence-to-sequence Generation: Our Model


X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs
+ *n*-best list **reranker**

Sequence-to-sequence Generation: Our Model



X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>

inform name X-name inform eattype restaurant

<GO>X-name is a restaurant .

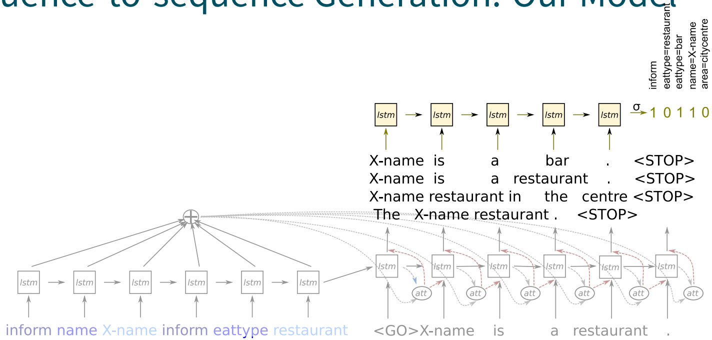
- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

+ *n*-best list **reranker**

  - to penalize missing/superfluous information in outputs

Sequence-to-sequence Generation: Our Model



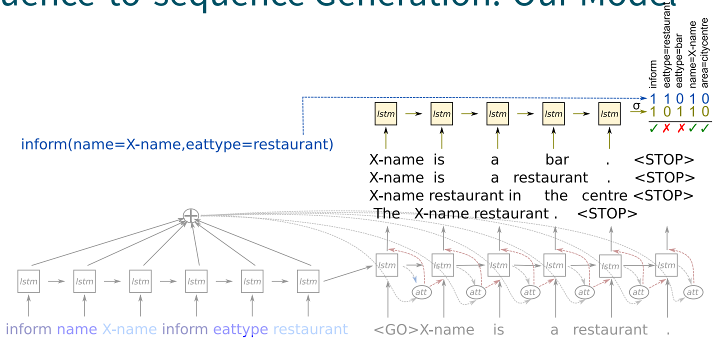
- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

+ *n*-best list **reranker**

  - to penalize missing/superfluous information in outputs
  - classify DA from output

Sequence-to-sequence Generation: Our Model


inform
eattype=restaurant
eattype=bar
name=X-name
area=citycentre
1 1 0 1 0
σ
1 0 1 1 0
✓ ✗ ✗ ✓ ✓
lstm
inform(name=X-name,eattype=restaurant)
X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

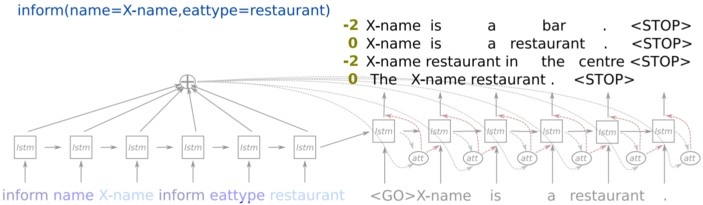
- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

+ *n*-best list **reranker**

  - to penalize missing/superfluous information in outputs
  - classify DA from output, compare to input DA

Sequence-to-sequence Generation: Our Model



inform(name=X-name,eattype=restaurant)

-2 X-name is a bar . <STOP>
0 X-name is a restaurant . <STOP>
-2 X-name restaurant in the centre <STOP>
0 The X-name restaurant . <STOP>

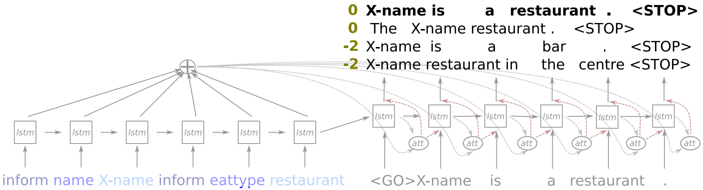
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

+ *n*-best list **reranker**
  - to penalize missing/superfluous information in outputs
  - classify DA from output, compare to input DA

Sequence-to-sequence Generation: Our Model



0 X-name is a restaurant . <STOP>
0 The X-name restaurant . <STOP>
-2 X-name is a bar . <STOP>
-2 X-name restaurant in the centre <STOP>
lstm
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .

- Main generator: seq2seq with attention (Bahdanau et al., 2015)

+ beam search, *n*-best list outputs

+ *n*-best list **reranker**

  - to penalize missing/superfluous information in outputs
  - classify DA from output, compare to input DA

Seq2seq Reranker Details

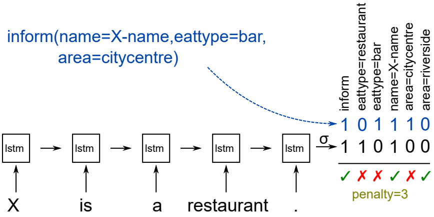
- generator may not cover the input DA perfectly
  - missing / superfluous information

Seq2seq Reranker Details

- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we want to penalize such cases

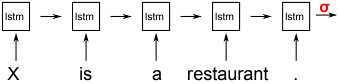
Seq2seq Reranker Details

- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we want to penalize such cases
- check whether output conforms to the input DA + rerank



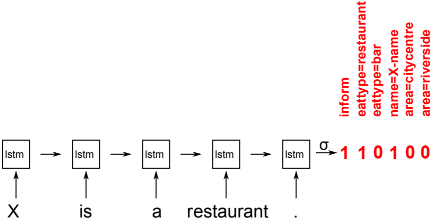
Seq2seq Reranker Details

- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we want to penalize such cases
- check whether output conforms to the input DA + rerank
  - LSTM RNN encoder + sigmoid classification layer



Seq2seq Reranker Details

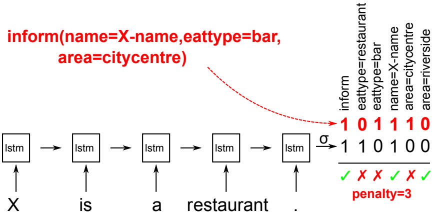
- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we want to penalize such cases
- check whether output conforms to the input DA + rerank
  - LSTM RNN encoder + sigmoid classification layer



- 1-hot DA representation

Seq2seq Reranker Details

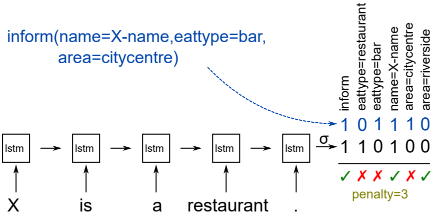
- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we want to penalize such cases
- check whether output conforms to the input DA + rerank
  - LSTM RNN encoder + sigmoid classification layer



- 1-hot DA representation
- penalty = Hamming distance from input DA (on 1-hot vectors)

Seq2seq Reranker Details

- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we want to penalize such cases
- check whether output conforms to the input DA + rerank
  - LSTM RNN encoder + sigmoid classification layer



- 1-hot DA representation
- penalty = Hamming distance from input DA (on 1-hot vectors)

Experiments on the BAGEL Set

- BAGEL dataset (Mairesse et al., 2010):
  202 DAs / 404 sentences, restaurant information

Experiments on the BAGEL Set

- BAGEL dataset (Mairesse et al., 2010):
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods

Experiments on the BAGEL Set

- BAGEL dataset (Mairesse et al., 2010):
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods
  - partially delexicalized (names, phone numbers → "X")

Experiments on the BAGEL Set

- BAGEL dataset (Mairesse et al., 2010):
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods
  - partially delexicalized (names, phone numbers → “X”)
  - manual alignment provided, but we do not use it

Experiments on the BAGEL Set

- BAGEL dataset (Mairesse et al., 2010):
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods
  - partially delexicalized (names, phone numbers → “X”)
  - manual alignment provided, but we do not use it
- 10-fold cross-validation
  - automatic metrics: BLEU, NIST

Experiments on the BAGEL Set

- BAGEL dataset (Mairesse et al., 2010):
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods
  - partially delexicalized (names, phone numbers → “X”)
  - manual alignment provided, but we do not use it
- 10-fold cross-validation
  - automatic metrics: BLEU, NIST
  - manual evaluation: semantic errors on 20% data (missing/irrelevant/repeated)

BAGEL Seq2seq Results

prev

Setup	BLEU	NIST	ERR
Mairesse et al. (2010) *– alignments*	$\sim$67	-	0
Our A*/perceptron	59.89	5.231	30

BAGEL Seq2seq Results

		Setup	**BLEU**	**NIST**	**ERR**
prev		Mairesse et al. (2010) – *alignments*	∼67	-	0
		Our A*/perceptron	59.89	5.231	30
our	*two-step*	Greedy with trees	55.29	5.144	20
		+ Beam search (beam size 100)	58.59	5.293	28
		+ Reranker (beam size 5)	60.77	5.487	24
		(beam size 10)	60.93	5.510	25
		+ Reranker (beam size 100)	60.44	5.514	**19**
	joint	Greedy into strings	52.54	5.052	37
		+ Beam search (beam size 100)	55.84	5.228	32
		+ Reranker (beam size 5)	61.18	5.507	27
		(beam size 10)	62.40	5.614	21
		+ Reranker (beam size 100)	**62.76**	**5.669**	**19**

Sample Outputs on the BAGEL set

Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, area=riverside, food=French)
Reference	X is a French restaurant on the riverside.
Greedy with trees	X is a restaurant providing french and continental and by the river.
+ Beam search	X is a restaurant that serves french takeaway. [riverside]
+ Reranker	X is a french restaurant in the riverside area.
Greedy into strings	X is a restaurant in the riverside that serves italian food. [French]
+ Beam search	X is a restaurant in the riverside that serves italian food. [French]
+ Reranker	X is a restaurant in the riverside area that serves french food.

Sample Outputs on the BAGEL set

Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, area=citycentre, near=X-near, food="Chinese takeaway", food=Japanese)
Reference	X is a Chinese takeaway and Japanese restaurant in the city centre near X.
Greedy with trees	X is a restaurant offering chinese takeaway in the centre of town near X. [Japanese]
+ Beam search	X is a restaurant and japanese food and chinese takeaway.
+ Reranker	X is a restaurant serving japanese food in the centre of the city that offers chinese takeaway.
Greedy into strings	X is a restaurant offering italian and indian takeaway in the city centre area near X. [Japanese, Chinese]
+ Beam search	X is a restaurant that serves fusion chinese takeaway in the riverside area near X. [Japanese, citycentre]
+ Reranker	X is a japanese restaurant in the city centre near X providing chinese food. [takeaway]

Sample Outputs on the BAGEL set

Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, near=X-near, food=Continental, food=French)
Reference	X is a French and Continental restaurant near X.
Greedy with trees	X is a french restaurant that serves french food and near X. [Continental]
+ Beam search	X is a french restaurant that serves french food and near X. [Continental]
+ Reranker	X is a restaurant serving french and continental food near X.
Greedy into strings	X is a french and continental style restaurant near X.
+ Beam search	X is a french and continental style restaurant near X.
+ Reranker	X is a restaurant providing french and continental food, near X.

Collecting Context-aware Data via CrowdFlower

1. Get natural user utterances in calls to a live dialogue system

Collecting Context-aware Data via CrowdFlower

1. Get natural user utterances in calls to a live dialogue system
   - record calls to live Alex SDS,
     task descriptions use varying synonyms

You want a connection – your departure stop is *Marble Hill*, and you want to go to *Roosevelt Island*. Ask how long the journey will take. Ask about a schedule afterwards. Then modify your query: Ask for a ride at six o'clock in the evening. Ask for a connection by bus. Do as if you changed your mind: Say that your destination stop is *City Hall*.

You are searching for transit options leaving from *Houston Street* with the destination of *Marble Hill*. When you are offered a schedule, ask about the time of arrival at your destination. Then ask for a connection after that. Modify your query: Request information about an alternative at six p.m. and state that you prefer to go by bus.

Tell the system that you want to travel from *Park Place* to *Inwood*. When you are offered a trip, ask about the time needed. Then ask for another alternative. Change your search: Ask about a ride at 6 o'clock p.m. and tell the system that you would rather use the bus.

Collecting Context-aware Data via CrowdFlower

1. Get natural user utterances in calls to a live dialogue system
    - record calls to live Alex SDS,
      task descriptions use varying synonyms
    - manual transcription + reparsing using Alex SLU

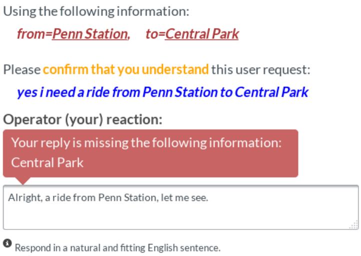
Collecting Context-aware Data via CrowdFlower

1. Get natural user utterances in calls to a live dialogue system
   - record calls to live Alex SDS,
     task descriptions use varying synonyms
   - manual transcription + reparsing using Alex SLU
2. Generate possible response DAs for the user utterances
   - using simple rule-based bigram policy

Collecting Context-aware Data via CrowdFlower

1. Get natural user utterances in calls to a live dialogue system
   - record calls to live Alex SDS,
     task descriptions use varying synonyms
   - manual transcription + reparsing using Alex SLU
2. Generate possible response DAs for the user utterances
   - using simple rule-based bigram policy
3. Collect natural language paraphrases for the response DAs

Collecting Context-aware Data via CrowdFlower



Using the following information:

from=Penn Station, to=Central Park

Please confirm that you understand this user request:

yes i need a ride from Penn Station to Central Park

Operator (your) reaction:

Your reply is missing the following information: Central Park

Alright, a ride from Penn Station, let me see.

Respond in a natural and fitting English sentence.

3. Collect natural language paraphrases for the response DAs
    - interface designed to support entrainment
        - context at hand
        - minimal slot description
        - short instructions

Collecting Context-aware Data via CrowdFlower

1. Get natural user utterances in calls to a live dialogue system
   - record calls to live Alex SDS,
     task descriptions use varying synonyms
   - manual transcription + reparsing using Alex SLU
2. Generate possible response DAs for the user utterances
   - using simple rule-based bigram policy
3. Collect natural language paraphrases for the response DAs
   - interface designed to support entrainment
     - context at hand
     - minimal slot description
     - short instructions
   - checks: contents + spelling, automatic + manual
     - ca. 20% overhead (repeated job submission)

Collecting Context-aware Data: Generating response DA

- Handcrafted simple rule-based bigram policy

Collecting Context-aware Data: Generating response DA

- Handcrafted simple rule-based bigram policy
- All possible replies for a single context utterance

what about a connection by bus

iconfirm(vehicle=bus)

inform(from_stop="Dyckman Street", direction="Park Place",
vehicle=bus, line=M103, departure_time=7:05pm)

inform_no_match(vehicle=bus)

request(to_stop)

Collecting Context-aware Data: Generating response DA

- Handcrafted simple rule-based bigram policy
- All possible replies for a single context utterance
    - confirmation
    - answer
    - apology
    - request for additional information

what about a connection by bus

iconfirm(vehicle=bus)

inform(from_stop=”Dyckman Street”, direction=”Park Place”,
vehicle=bus, line=M103, departure_time=7:05pm)

inform_no_match(vehicle=bus)

request(to_stop)

Collecting Context-aware Data: Generating response DA

- Handcrafted simple rule-based bigram policy
- All possible replies for a single context utterance
  - confirmation
  - answer
  - apology
  - request for additional information
- In a real dialogue, the correct reply would depend on longer history, but here we try them all

Entrainment Dataset Summary

Size

total response paraphrases	5,577
unique (delex.) context + response DA	1,859
unique (delex.) context	552
unique (delex.) context with min. 2 occurrences	119
unique response DA	83
unique response DA types	6
unique slots	13

Entrainment

Syntactic	∼59%
Lexical	∼31%
Both	∼19%

- subjective, based on word & phrase reuse, word order, pronouns

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037
n-gram match reranker	68.68	7.577

Entrainment Experiments

- Dataset: public transport information
    - 5.5k paraphrases for 1.8k DA-context combinations
    - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037
n-gram match reranker	68.68	7.577
Prepending context	63.87	6.456

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037
n-gram match reranker	68.68	7.577
Prepending context	63.87	6.456
+ *n*-gram match reranker	**69.26**	**7.772**

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037
n-gram match reranker	68.68	7.577
Prepending context	63.87	6.456
+ *n*-gram match reranker	**69.26**	**7.772**
Context encoder	63.08	6.818
+ *n*-gram match reranker	69.17	7.596

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037
n-gram match reranker	68.68	7.577
Prepending context	63.87	6.456
+ *n*-gram match reranker	**69.26**	**7.772**
Context encoder	63.08	6.818
+ *n*-gram match reranker	69.17	7.596

- Human pairwise preference ranking (crowdsourced)
  - baseline × prepending context + *n*-gram match reranker

Entrainment Experiments

- Dataset: public transport information
  - 5.5k paraphrases for 1.8k DA-context combinations
  - delexicalized

Setup	**BLEU**	**NIST**
Baseline (context not used)	66.41	7.037
n-gram match reranker	68.68	7.577
Prepending context	63.87	6.456
+ *n*-gram match reranker	**69.26**	**7.772**
Context encoder	63.08	6.818
+ *n*-gram match reranker	69.17	7.596

- Human pairwise preference ranking (crowdsourced)
  - baseline × prepending context + *n*-gram match reranker
  - context-aware **preferred in 52.5% cases** (significant)

Entrainment Output Examples

Context	is there a later option
Input DA	iconfirm(alternative=next)
Baseline	Next connection.
n-gram match reranker	You want a later connection.
Prepending context + *n*-gram match reranker	You want a later connection.
Context encoder + *n*-gram match reranker	You want a later option.

Entrainment Output Examples

Context	is there a later option
Input DA	iconfirm(alternative=next)
Baseline	Next connection.
n-gram match reranker	You want a later connection.
Prepending context + *n*-gram match reranker	You want a later connection.
Context encoder + *n*-gram match reranker	You want a later option.

Context	i need to find a bus connection
Input DA	inform_no_match(vehicle=bus)
Baseline	No bus found, sorry.
n-gram match reranker	I did not find a bus route.
Prepending context + *n*-gram match reranker	I’m sorry, I cannot find a bus connection.
Context encoder + *n*-gram match reranker	I’m sorry, I cannot find a bus connection.

Entrainment Output Examples (2)

Context	i rather take the bus
Input DA	inform(vehicle=bus, departure_time=8:01am, direction=Cathedral Parkway, from_stop=Bowling Green, line=M15)
Baseline	At 8:01am by bus line M15 from Bowling Green to Cathedral Parkway.
n-gram match reranker	At 8:01am by bus line M15 from Bowling Green to Cathedral Parkway.
Prepending context + *n*-gram match reranker	You can take the M15 bus from Bowling Green to Cathedral Parkway at 8:01am.
Context encoder + *n*-gram match reranker	At 8:01am by bus line M15 from Bowling Green to Cathedral Parkway.

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
    - no Czech speakers on platforms

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
    - no Czech speakers on platforms

→ Translating an English set (restaurants, Wen et al. 2015)

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
  - no Czech speakers on platforms

→ Translating an English set (restaurants, Wen et al. 2015)

1. delexicalization

inform(name="Fog Harbor Fish House", price_range=cheap, area="Civic Center")
Fog Harbor Fish House is cheap and it is located in Civic Center.

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
  - no Czech speakers on platforms

→ Translating an English set (restaurants, Wen et al. 2015)

1. delexicalization

inform(name="X-name", price_range=X-pricerange, area="X-area")
X-name is X-pricerange and it is located in X-area.

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
  - no Czech speakers on platforms

→ Translating an English set (restaurants, Wen et al. 2015)

1. delexicalization
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, names require inflection)

inform(name="Ferdinanda", price_range=expensive, area="Hradčany")
Ferdinanda is expensive and it is located in Hradčany.

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
  - no Czech speakers on platforms

→ Translating an English set (restaurants, Wen et al. 2015)

1. delexicalization
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, names require inflection)
3. translation by hired translators

inform(name="Ferdinanda", price_range=expensive, area="Hradčany")
Ferdinanda je **levná** *(cheap)* a nachází se na Hradčanech.

Creating a Czech NLG Dataset

- Virtually no non-English NLG datasets available
- Collecting Czech data via crowdsourcing is not an option
  - no Czech speakers on platforms

→ Translating an English set (restaurants, Wen et al. 2015)

1. delexicalization
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, names require inflection)
3. translation by hired translators
4. automatic & manual checks

inform(name="Ferdinanda", price_range=expensive, area="Hradčany")
Ferdinanda je drahá a nachází se na Hradčanech.

Czech NLG: Lemma-tag generation

- 3rd generator mode
    - compromise between full 2-step/joint setups

Czech NLG: Lemma-tag generation

- 3rd generator mode
  - compromise between full 2-step/joint setups

idea: let the seq2seq model decide everything...

Czech NLG: Lemma-tag generation

- 3rd generator mode
  - compromise between full 2-step/joint setups

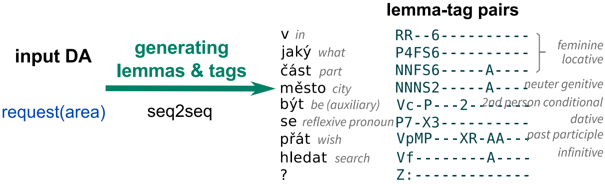
idea: let the seq2seq model decide everything...
but for complex morphological inflection

Czech NLG: Lemma-tag generation

- 3rd generator mode
    - compromise between full 2-step/joint setups

idea: let the seq2seq model decide everything...
but for complex morphological inflection

- generating into list of interleaved morph. tags and lemmas



lemma-tag pairs

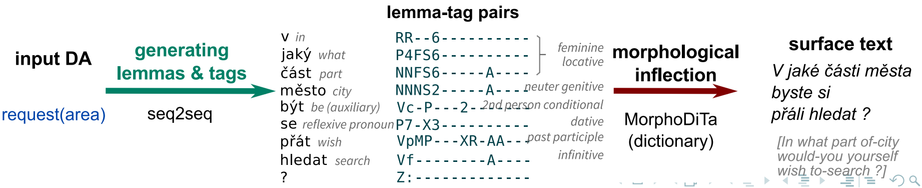
v	*in*	RR--6----------	*feminine locative*
jaký	*what*	P4FS6----------	
část	*part*	NNFS6-----A----	
město	*city*	NNNS2-----A----	*neuter genitive*
být	*be (auxiliary)*	Vc-P---2-------	*2nd person conditional*
se	*reflexive pronoun*	P7-X3----------	*dative*
přát	*wish*	VpMP---XR-AA---	*past participle*
hledat	*search*	Vf--------A----	*infinitive*
?		Z:-------------	

Czech NLG: Lemma-tag generation

- 3rd generator mode
  - compromise between full 2-step/joint setups

idea: let the seq2seq model decide everything...
but for complex morphological inflection

- generating into list of interleaved morph. tags and lemmas
- postprocessing:
  - MorphoDiTa dictionary
  - list of surface forms for names



v	*in*	`RR--6----------`	*feminine locative*
jaký	*what*	`P4FS6----------`	
část	*part*	`NNFS6-----A----`	
město	*city*	`NNNS2-----A----`	*neuter genitive*
být	*be (auxiliary)*	`Vc-P---2-------`	*2nd person conditional*
se	*reflexive pronoun*	`P7-X3----------`	*dative*
přát	*wish*	`VpMP---XR-AA---`	*past participle*
hledat	*search*	`Vf--------A----`	*infinitive*
?		`Z:-------------`	

V jaké části města byste si přáli hledat ?

[In what part of-city would-you yourself wish to-search ?]

Czech NLG: Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***   *(‘<name> is cheap’)*

Czech NLG: Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***   *(‘<name> is cheap’)*
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově*   *(‘in <neighborhood>’)*

Czech NLG: Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý*** *('<name> is cheap')*
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově* *('in <neighborhood>')*

inform(name="X-name", price_range=X-pricerange, area="X-area")
X-name je X-pricerange a nachází se v X-area.
X-name is X-pricerange and it is located in X-area.

Czech NLG: Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý*** *('<name> is cheap')*
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově* *('in <neighborhood>')*

→ Keep values in input DAs (don't delexicalize)

  - still generating delexicalized outputs

inform(name="X-name", price_range=X-pricerange, area="X-area")
X-name je X-pricerange a nachází se v X-area.
X-name is X-pricerange and it is located in X-area.

Czech NLG: Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***   *(‘<name> is cheap’)*
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově*   *(‘in <neighborhood>’)*

→ Keep values in input DAs (don’t delexicalize)

  - still generating delexicalized outputs

inform(name=“**Café Savoy**”, price_range=**cheap**, area=“**Smíchov**”)
X-name je X-pricerange a nachází se **na** X-area.
X-name is X-pricerange and it is located in X-area.

Czech NLG: Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý*** *('<name> is cheap')*
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově* *('in <neighborhood>')*

→ Keep values in input DAs (don't delexicalize)

- still generating delexicalized outputs

! This is proof-of-concept

- exploiting small number of lexical values
- real world: morphological properties / character embeddings

inform(name="Café Savoy", price_range=cheap, area="Smíchov")
X-name je X-pricerange a nachází se na X-area.
X-name is X-pricerange and it is located in X-area.

Full Czech Restaurants BLEU/NIST Results

	Setup			
input DAs	**generator mode**	**lexicalization**	**BLEU**	**NIST**
delexicalized	joint (direct to strings)	random	13.47	3.442
		most frequent	19.31	**4.346**
		n-gram LM	19.40	4.274
		RNN LM	**19.54**	4.273
	lemma-tag	random	17.18	3.985
		most frequent	18.22	4.162
		n-gram LM	17.95	4.132
		RNN LM	18.51	4.162
	two-step with t-trees	random	14.93	3.784
		most frequent	16.16	3.969
		n-gram LM	16.13	3.970
		RNN LM	16.39	3.974
lexically informed	joint (direct to strings)	random	12.56	3.300
		most frequent	17.82	4.164
		n-gram LM	17.85	4.082
		RNN LM	17.93	4.094
	lemma-tag	random	19.96	4.306
		most frequent	20.86	4.427
		n-gram LM	20.54	4.399
		RNN LM	**21.18**	**4.448**
	two-step with t-trees	random	16.13	3.919
		most frequent	17.15	4.073
		n-gram LM	17.24	4.078
		RNN LM	17.62	4.112

- understandable Czech
- some fluency errors
- semantic errors very rare

- lexically informed better
- two-step with trees worse
- RNN lexicalization best

Czech: Human Evaluation

- Selected setups based on BLEU/NIST (7 out of 24)

Czech: Human Evaluation

- Selected setups based on BLEU/NIST (7 out of 24)
- WMT-style multi-way relative comparisons

Czech: Human Evaluation

- Selected setups based on BLEU/NIST (7 out of 24)
- WMT-style multi-way relative comparisons
- overall preference (no criteria)

Czech: Human Evaluation

- Selected setups based on BLEU/NIST (7 out of 24)
- WMT-style multi-way relative comparisons
- overall preference (no criteria)
- TrueSkill$^{\text{TM}}$, bootstrap clustering

	Setup		True	Rank	BLEU
input DAs	**generator mode**	**lexicalization**	**Skill**		
delexicalized	joint (direct to strings)	RNN LM	**0.511**	1	19.54
delexicalized	lemma-tag	RNN LM	0.479	2-4	18.51
lexically informed	lemma-tag	RNN LM	0.464	2-4	**21.18**
lexically informed	lemma-tag	most frequent	0.462	2-4	20.86
lexically informed	joint (direct to strings)	RNN LM	0.413	5	17.93
lexically informed	two-step with t-trees	RNN LM	0.343	6-7	17.62
lexically informed	lemma-tag	*n*-gram LM	0.329	6-7	20.54

Czech Output Examples (1)

Input DA	inform(name=“Švejk Restaurant”, near=Stromovka, price_range=cheap)
Reference	*Restaurace Švejk je poblíž Stromovky a nabízí nízké ceny.* Švejk Restaurant is near Stromovka and it offers low prices.
Best BLEU/NIST	*Restaurace Švejk je levná restaurace, poblíž Stromovky.* Švejk Restaurant is a cheap$_{\textbf{fem}}$ restaurant$_{\textbf{fem}}$, near Stromovka.
Most preferred	*Restaurace Švejk je levná podnik blízko Stromovky* Švejk Restaurant is a cheap$_{\textbf{fem}}$ place$_{\textbf{inan}}$ close to Stromovka
Input DA	inform(name=“U Konšelů”, phone=X-phone)
Reference	*Do U Konšelů můžete zavolat na telefonu 242817033.* to U Konšelů you can call on the phone 242817033.
Best BLEU/NIST	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.* price to the restaurant U Konšelů you can call the number 242817033.
Most preferred	*Telefonní číslo U Konšelů je 242817033.* the phone number of U Konšelů is 242817033.
Input DA	inform_no_match(area=Vinohrady)
Reference	*Lituji, ale v lokalitě Vinohrad se nenachází žádné* I am sorry but in the area of Vinohrady are not located any *restaurace.* restaurants
Best BLEU/NIST	*Na Vinohradech nejsou žádné restaurace vhodné pro děti.* in Vinohrady there are no restaurants appropriate for children.
Most preferred	*V Vinohrad nejsou žádné takové restaurace.* in Vinohrady$_{\textbf{gen}}$ there are no such restaurants.

Czech Output Examples (2)

Input DA	inform(type=restaurant, good_for_meal=dont_care, count=3)
Reference	*Jsou zde 3 restaurace, které jsou vhodné pro jakékoli jídlo* there are 3 restaurants$_{\textbf{nom}}$ that are suitable for any meal
Best BLEU/NIST	*Našla jsem 3 restaurací, které jsou vhodné pro jakékoli jídlo.* I found 3 restaurants$_{\textbf{gen}}$, which are suitable for any meal.
Most preferred	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.* on the list are 3 restaurants$_{\textbf{gen}}$, that offer all kinds of meals.
Input DA	inform(area=Hradčany, type=restaurant, kids_allowed=no, count=2)
Reference	*V lokalitě Hradčan jsem našla 2 restaurace, které nedovolují* in the area of Hradčany I found 2 restaurants, which do not allow *vstup dětem.* entry to children.
Best BLEU/NIST	*V oblasti Hradčan se nabízí 2 restaurace, které nejsou* in the area of Hradčany offer themselves 2 restaurants, which are not *vhodné pro děti.* appropriate for children.
Most preferred	*Na Hradčany se nehodí 2 restaurace, které nejsou vhodné* for Hradčany are not suitable 2 restaurants, which are not appropriate *pro děti.* for children.