Sequence-to-Sequence Natural Language Generation

Ondřej Dušek
work done with Filip Jurčíček

Institute of Formal and Applied Linguistics, Charles University, Prague
Interaction Lab, Heriot-Watt University, Edinburgh

March 28, 2016
ÚFAL Monday Seminar

1. Introduction to the problem
    a) our task + problems we are solving
2. Sequence-to-sequence Generation
    a) basic model architecture
    b) generating directly / via deep syntax trees
    c) experiments on the BAGEL Set
3. Context-aware extensions (user adaptation/entrainment)
    a) collecting a context-aware dataset
    b) making the basic seq2seq setup context-aware
    c) experiments on our dataset
4. Generating Czech
    a) creating a Czech NLG dataset
    b) generator extensions for Czech
    c) experiments on our dataset
5. Conclusions and future work ideas

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.

  - no content selection here

NLG in Spoken Dialogue Systems

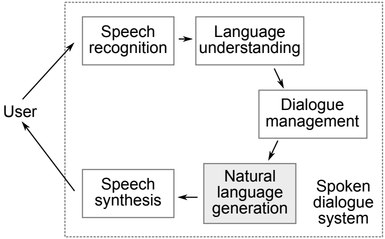
- converting a meaning representation (dialogue acts, DAs) to a sentence

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

↓

X is an Italian restaurant near the river.

  - no content selection here
- input: from dialogue manager
- output: to TTS

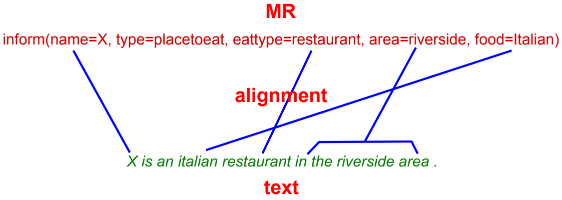


Problem 1: Generating from Unaligned Data

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

Problem 1: Generating from Unaligned Data

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



Problem 1: 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

Problem 1: 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

Problem 1: 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**.*

Problem 1: Gen. from Unaligned Data – **Delexicalization**

- Limitation / way to address data sparsity

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.

Problem 1: Gen. from Unaligned Data – **Delexicalization**

- Limitation / way to address data sparsity
  - many slot values seen once or never in training

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.

Problem 1: Gen. from Unaligned Data – **Delexicalization**

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

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.

Problem 1: Gen. from Unaligned Data – **Delexicalization**

- Limitation / 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

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.

Problem 1: Gen. from Unaligned Data – **Delexicalization**

- Limitation / 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

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.

Problem 1: Gen. from Unaligned Data – **Delexicalization**

- Limitation / 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

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.

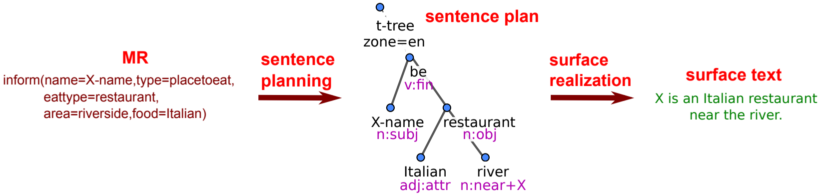
Problem 1: Gen. from Unaligned Data – **Delexicalization**

- Limitation / 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:	rest. name, phone number, postcode

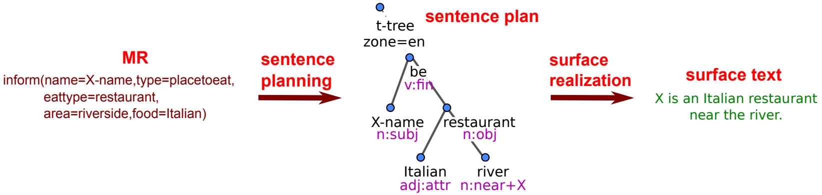
Problem 2: Comparing Different NLG Architectures

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



Problem 2: Comparing Different NLG Architectures

- 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



Problem 2: Comparing Different NLG Architectures

- 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

Problem 2: Comparing Different NLG Architectures

- 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
  - two-step setup simplifies structure generation by abstracting away from surface grammar

Problem 2: Comparing Different NLG Architectures

- 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
  - two-step setup simplifies structure generation by abstracting away from surface grammar
  - joint setup avoids error accumulation over a pipeline

Problem 2: Comparing Different NLG Architectures

- 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
    - two-step setup simplifies structure generation by abstracting away from surface grammar
    - joint setup avoids error accumulation over a pipeline
- we try both in one system + compare

Problem 3: Adapting to the User (Entrainment)

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

Problem 3: Adapting to the User (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, <u>the next ride</u> was not found.

Problem 3: Adapting to the User (Entrainment)

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

Problem 3: Adapting to the User (Entrainment)

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success
- natural source of variation

Problem 3: Adapting to the User (Entrainment)

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

Problem 3: Adapting to the User (Entrainment)

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

Problem 3: Adapting to the User (Entrainment)

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success
- natural source of variation
- typical NLG only takes the input DA into account
  - no way of adapting to user's way of speaking
  - no output variance (must be fabricated, e.g., by sampling)

Problem 3: Adapting to the User (Entrainment)

- speakers are influenced by previous utterances
  - adapting (entraining) to each other
  - reusing lexicon and syntax
- entrainment is natural, subconscious
- entrainment helps conversation success
- natural source of variation
- typical NLG only takes the input DA into account
  - no way of adapting to user's way of speaking
  - no output variance (must be fabricated, e.g., by sampling)
- entrainment in NLG limited to rule-based systems so far

Problem 3: Adapting to the User (Entrainment)

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

Problem 4: Multilingual NLG

- English: little morphology

Problem 4: Multilingual NLG

- English: little morphology
  - vocabulary size relatively small

Problem 4: Multilingual NLG

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

Problem 4: Multilingual NLG

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

Problem 4: 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
- None of this works with rich morphology

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*

Problem 4: 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
- None of this works with rich morphology
  - → Czech is a good language to try

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*

Problem 4: 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
- None of this works with rich morphology
  - → Czech is a good language to try
- Extensions to our generator to address this:
  - 3rd generator mode: generating lemmas & morphological tags

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

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

Problem 4: 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
- None of this works 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)

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

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

Our NLG system

- based on sequence-to-sequence neural network models

Our NLG system

- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
    - learns to produce meaningful outputs from little training data

Our NLG system

- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
  - learns to produce meaningful outputs from little training data
- ✓ multiple operating modes for comparison:
  a) generating sentences token-by-token (joint 1-step NLG)

Our NLG system

- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
  - learns to produce meaningful outputs from little training data
- ✓ multiple operating modes for comparison:
  a) generating sentences token-by-token (joint 1-step NLG)
  b) generating deep syntax trees in bracketed notation
  (sentence planner stage of traditional NLG pipeline)

Our NLG system

- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
  - learns to produce meaningful outputs from little training data
- ✓ multiple operating modes for comparison:
  a) generating sentences token-by-token (joint 1-step NLG)
  b) generating deep syntax trees in bracketed notation (sentence planner stage of traditional NLG pipeline)
- ✓ context-aware: adapts to previous user utterance

Our NLG system

- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
  - learns to produce meaningful outputs from little training data
- ✓ multiple operating modes for comparison:
  a) generating sentences token-by-token (joint 1-step NLG)
  b) generating deep syntax trees in bracketed notation (sentence planner stage of traditional NLG pipeline)
- ✓ context-aware: adapts to previous user utterance
- ✓ works for English and Czech

Our NLG system

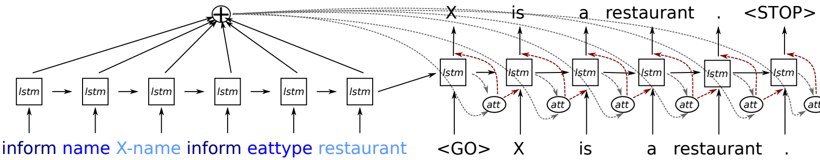
- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
  - learns to produce meaningful outputs from little training data
- ✓ multiple operating modes for comparison:
  a) generating sentences token-by-token (joint 1-step NLG)
  b) generating deep syntax trees in bracketed notation
  (sentence planner stage of traditional NLG pipeline)
- ✓ context-aware: adapts to previous user utterance
- ✓ works for English and Czech
  c) 3rd generator mode: lemma-tag pairs

Our NLG system

- based on sequence-to-sequence neural network models
- ✓ trainable from unaligned pairs of input DAs + sentences
  - learns to produce meaningful outputs from little training data
- ✓ multiple operating modes for comparison:
  a) generating sentences token-by-token (joint 1-step NLG)
  b) generating deep syntax trees in bracketed notation (sentence planner stage of traditional NLG pipeline)
- ✓ context-aware: adapts to previous user utterance
- ✓ works for English and Czech
  c) 3rd generator mode: lemma-tag pairs
  - includes proper name inflection for Czech

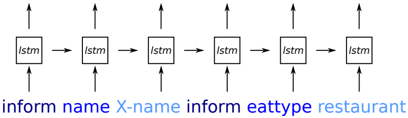
1. Introduction to the problem
   a) our task + problems we are solving
2. Sequence-to-sequence Generation
   a) basic model architecture
   b) generating directly / via deep syntax trees
   c) experiments on the BAGEL Set
3. Context-aware extensions (user adaptation/entrainment)
   a) collecting a context-aware dataset
   b) making the basic seq2seq setup context-aware
   c) experiments on our dataset
4. Generating Czech
   a) creating a Czech NLG dataset
   b) generator extensions for Czech
   c) experiments on our dataset
5. Conclusions and future work ideas

Our Seq2seq Generator architecture



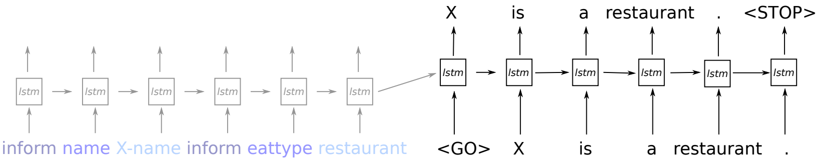
- Sequence-to-sequence models with attention

Our Seq2seq Generator architecture



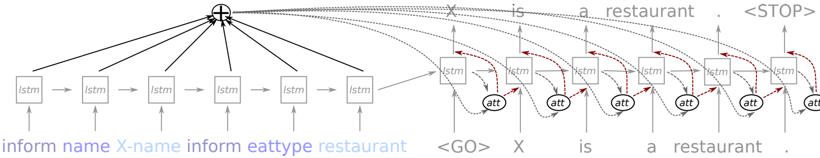
- Sequence-to-sequence models with attention
  - Encoder LSTM RNN: encode DA into hidden states

Our Seq2seq Generator architecture



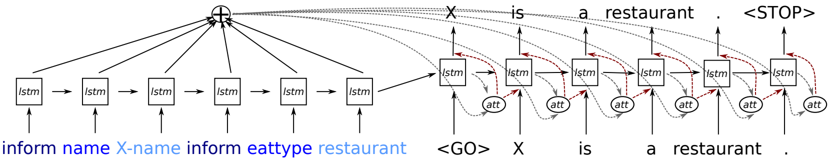
- Sequence-to-sequence models with attention
  - Encoder LSTM RNN: encode DA into hidden states
  - Decoder LSTM RNN: generate output tokens

Our Seq2seq Generator architecture



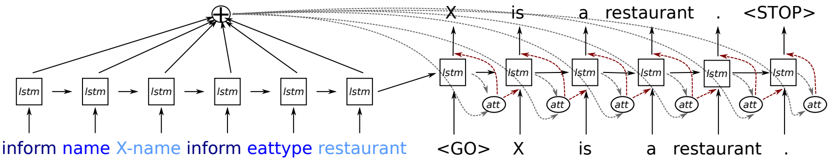
- Sequence-to-sequence models with attention
  - Encoder LSTM RNN: encode DA into hidden states
  - Decoder LSTM RNN: generate output tokens
  - attention model: weighing encoder hidden states

Our Seq2seq Generator architecture



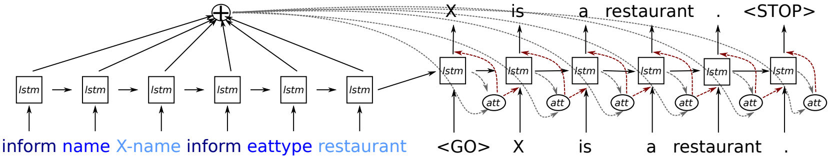
- Sequence-to-sequence models with attention
  - Encoder LSTM RNN: encode DA into hidden states
  - Decoder LSTM RNN: generate output tokens
  - attention model: weighing encoder hidden states
- basic greedy generation

Our Seq2seq Generator architecture



- Sequence-to-sequence models with attention
  - Encoder LSTM RNN: encode DA into hidden states
  - Decoder LSTM RNN: generate output tokens
  - attention model: weighing encoder hidden states
- basic greedy generation
  + beam search, *n*-best list outputs

Our Seq2seq Generator architecture



- Sequence-to-sequence models with attention
  - Encoder LSTM RNN: encode DA into hidden states
  - Decoder LSTM RNN: generate output tokens
  - attention model: weighing encoder hidden states
- basic greedy generation
  + beam search, *n*-best list outputs
  + reranker ($\rightarrow$)

Reranker

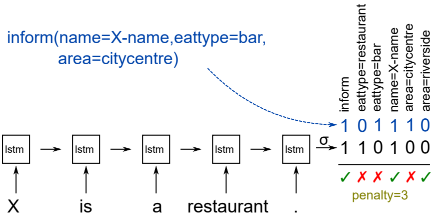
- generator may not cover the input DA perfectly
  - missing / superfluous information

Reranker

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

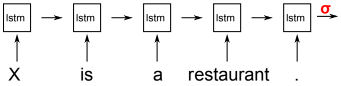
Reranker

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



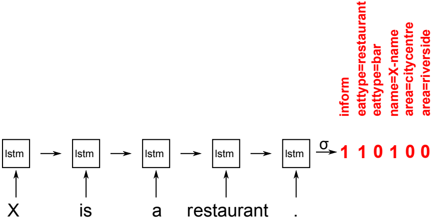
Reranker

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



Reranker

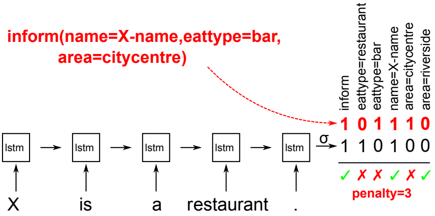
- generator may not cover the input DA perfectly
    - missing / superfluous information
    - we would like to penalize such cases
- check whether output conforms to the input DA + rerank
    - NN with LSTM encoder + sigmoid classification layer



- 1-hot DA representation

Reranker

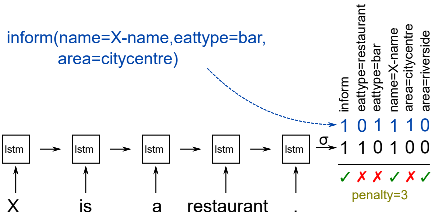
- generator may not cover the input DA perfectly
  - missing / superfluous information
  - we would like to penalize such cases
- check whether output conforms to the input DA + rerank
  - NN with LSTM encoder + sigmoid classification layer



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

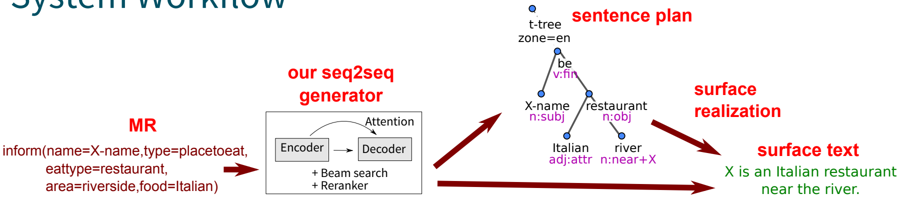
Reranker

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

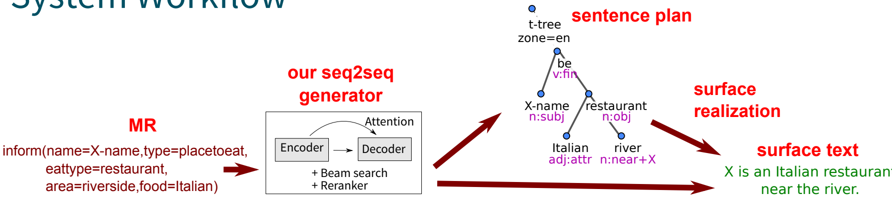


- 1-hot DA representation
- penalty = Hamming distance from input DA (on 1-hot vectors)

System Workflow

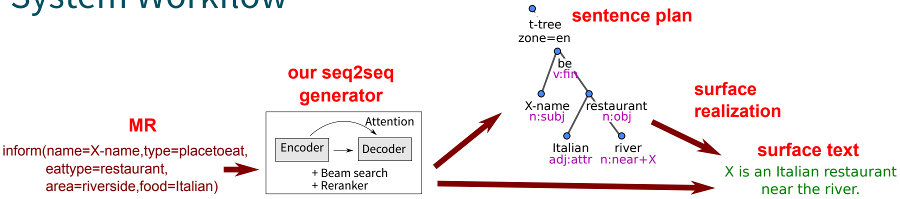


System Workflow



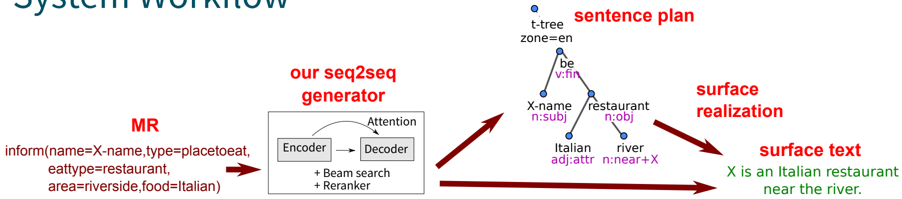
- main generator based on sequence-to-sequence NNs

System Workflow



- main generator based on sequence-to-sequence NNs
- input: tokenized DAs

System Workflow

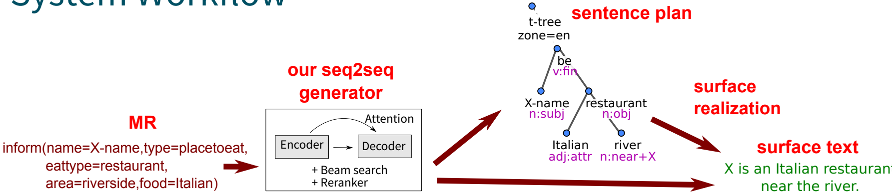


- main generator based on sequence-to-sequence NNs
- input: tokenized DAs
- output:

  2-step mode – deep syntax trees, in bracketed format

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

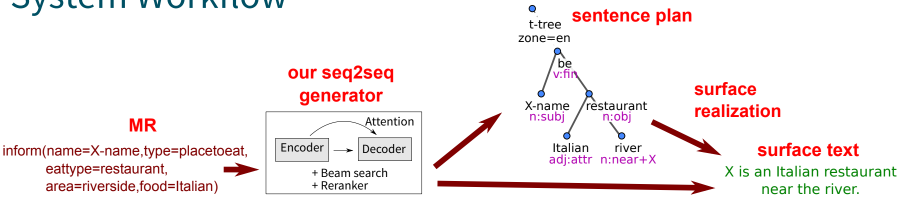
System Workflow



- main generator based on sequence-to-sequence NNs
- input: tokenized DAs
- output:

  2-step mode – deep syntax trees, in bracketed format
  joint mode – sentences

System Workflow



- main generator based on sequence-to-sequence NNs
- input: tokenized DAs
- output:

  2-step mode – deep syntax trees, in bracketed format

  joint mode – sentences
- 2-step mode: deep syntax trees post-processed by a surface realizer

Experiments

- BAGEL dataset:
  202 DAs / 404 sentences, restaurant information

Experiments

- BAGEL dataset:
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods

Experiments

- BAGEL dataset:
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods
  - partially delexicalized (names, phone numbers $\rightarrow$ “X”)

Experiments

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

Experiments

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

Experiments

- BAGEL dataset:
  202 DAs / 404 sentences, restaurant information
  - much less data than previous seq2seq methods
  - partially delexicalized (names, phone numbers $\rightarrow$ “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)

Results

	Setup	BLEU	NIST	ERR
prev	Mairesse et al. (2010) *– alignments*	∼67	-	0
	Dušek & Jurčíček (2015)	59.89	5.231	30

Results

		Setup	BLEU	NIST	ERR
prev		Mairesse et al. (2010) – *alignments*	∼67	-	0
		Dušek & Jurčíček (2015)	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
		(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
		(beam size 100)	**62.76**	**5.669**	**19**

Sample Outputs

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.

1. Introduction to the problem
   a) our task + problems we are solving
2. Sequence-to-sequence Generation
   a) basic model architecture
   b) generating directly / via deep syntax trees
   c) experiments on the BAGEL Set
3. Context-aware extensions (user adaptation/entrainment)
   a) collecting a context-aware dataset
   b) making the basic seq2seq setup context-aware
   c) experiments on our dataset
4. Generating Czech
   a) creating a Czech NLG dataset
   b) generator extensions for Czech
   c) experiments on our dataset
5. Conclusions and future work ideas

Adding Entrainment to Trainable NLG

- Aim: condition generation on preceding context

Adding Entrainment to Trainable NLG

- Aim: condition generation on preceding context
- Problem: data sparsity

Adding Entrainment to Trainable NLG

- Aim: condition generation on preceding context
- Problem: data sparsity
- Solution: Limit context to just preceding user utterance
    - likely to have strongest entrainment impact

Adding Entrainment to Trainable NLG

- Aim: condition generation on preceding context
- Problem: data sparsity
- Solution: Limit context to just preceding user utterance
  - likely to have strongest entrainment impact
- Need for context-aware training data: we collected a new set
  - input DA
  - natural language sentence(s)

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

Adding Entrainment to Trainable NLG

- Aim: condition generation on preceding context
- Problem: data sparsity
- Solution: Limit context to just preceding user utterance
  - likely to have strongest entrainment impact
- Need for context-aware training data: we collected a new set
  - input DA
  - natural language sentence(s)
  - preceding user 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

Adding Entrainment to Trainable NLG

- Aim: condition generation on preceding context
- Problem: data sparsity
- Solution: Limit context to just preceding user utterance
  - likely to have strongest entrainment impact
- Need for context-aware training data: we collected a new set
  - input DA
  - natural language sentence(s)
  - preceding user 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.*

Collecting the set (via CrowdFlower)

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

Collecting the set (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 the set (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

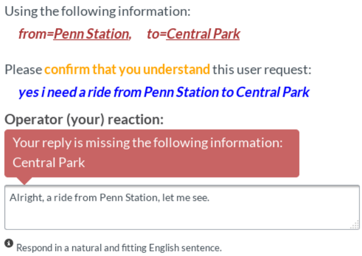
Collecting the set (via CrowdFlower)

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 the set (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 the set (via CrowdFlower)



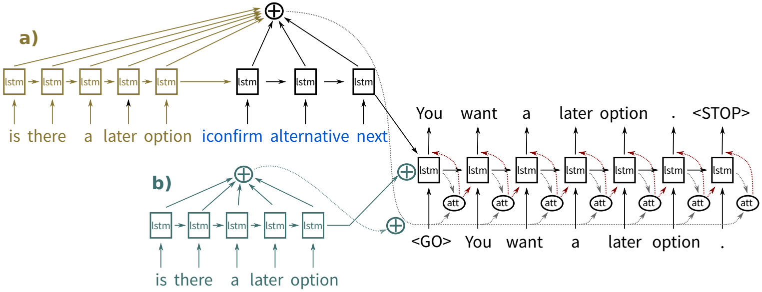
3. Collect natural language paraphrases for the response DAs
   - interface designed to support entrainment
     - context at hand
     - minimal slot description
     - short instructions

Collecting the set (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)

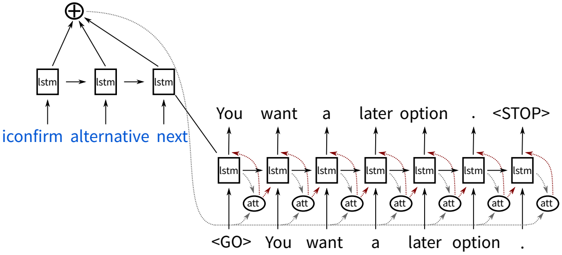
Context in our Seq2seq Generator (1)

- Two direct context-aware extensions:



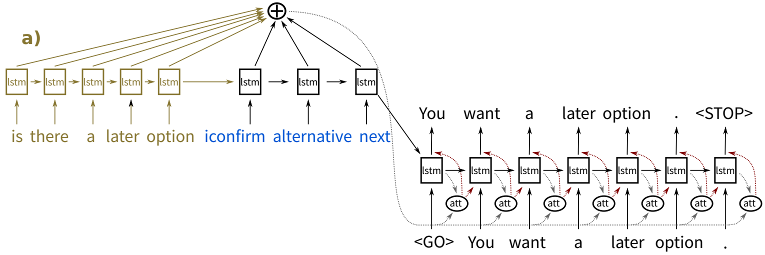
Context in our Seq2seq Generator (1)

- Two direct context-aware extensions:



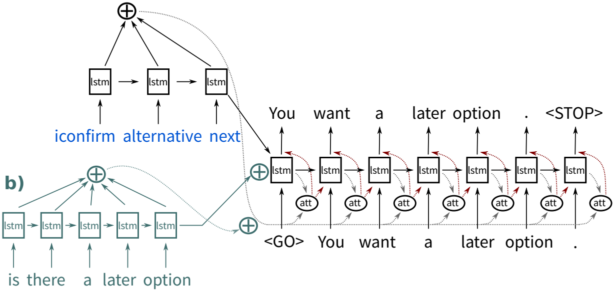
Context in our Seq2seq Generator (1)

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



Context in our Seq2seq Generator (1)

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



Context in our Seq2seq Generator (2)

- One (more) reranker: *n*-gram match

Context in our Seq2seq Generator (2)

- One (more) reranker: *n*-gram match
- promoting outputs that have a word or phrase overlap with the context utterance

Context in our Seq2seq Generator (2)

- One (more) reranker: *n*-gram match
- promoting outputs that have a word or phrase overlap with the context utterance

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 .

Experiments

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

Experiments

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

Automatic evaluation results	**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

Experiments

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

Automatic evaluation results	**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

Experiments

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

Automatic evaluation results	**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)

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.

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.

Output Examples

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.

1. Introduction to the problem
   a) our task + problems we are solving
2. Sequence-to-sequence Generation
   a) basic model architecture
   b) generating directly / via deep syntax trees
   c) experiments on the BAGEL Set
3. Context-aware extensions (user adaptation/entrainment)
   a) collecting a context-aware dataset
   b) making the basic seq2seq setup context-aware
   c) experiments on our dataset
4. Generating Czech
   a) creating a Czech NLG dataset
   b) generator extensions for Czech
   c) experiments on our dataset
5. Conclusions and future work ideas

Creating a Czech Dataset

- Virtually no NLG datasets available, except for English

Creating a Czech Dataset

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

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)

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.

inform(name="Fifth Floor", price_range=expensive, area="Hayes Valley")
Fifth Floor is expensive and it is located in Hayes Valey.

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)

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

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 Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)

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 Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, but need to be inflected)

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

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, but need to be inflected)
3. translation by hired translators

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

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, but need to be inflected)
3. translation by hired translators
4. automatic checks of slot values

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

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, but need to be inflected)
3. translation by hired translators
4. automatic checks of slot values
5. expansion to original size by relexicalizing

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

inform(name="Café Savoy", price_range=cheap, area="Smíchov")
Café Savoy je levn**á** a nachází se na Smíchově.

Creating a Czech Dataset

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

→ Translating an existing English set (restaurant information)

1. deduplicating delexicalized sentences (5,192 → 2,648)
2. localizing restaurant names, landmarks, etc., to Prague
   - (random combinations, but need to be inflected)
3. translation by hired translators
4. automatic checks of slot values
5. expansion to original size by relexicalizing
6. manual relexicalization checks

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

inform(name="Café Savoy", price_range=cheap, area="Smíchov")
Café Savoy je levn**é** a nachází se na Smíchově.

Czech: Lemma-tag generation

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

Czech: Lemma-tag generation

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

idea: let the seq2seq model decide everything...

Czech: 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: 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 Czech tags and lemmas

lemma-tag pairs

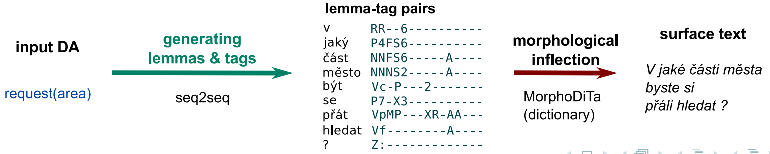
v	RR--6----------
jaký	P4FS6----------
část	NNFS6-----A----
město	NNNS2-----A----
být	Vc-P---2-------
se	P7-X3----------
přát	VpMP---XR-AA---
hledat	Vf--------A----
?	Z:-------------

Czech: 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 Czech tags and lemmas
- postprocessing:
  - MorphoDiTa dictionary
  - list of surface forms for proper names



Inflecting Proper Names

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

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
    - e.g., *obědvat* vs. *oběd*

?confirm(good_for_meal=brunch)

chcete najít vhodnou restauraci na X-good_for_meal ?

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*

?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---

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
- 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---

Inflecting Proper Names

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

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
- Two baselines:
  a) random surface form
  b) most frequent form in training data

?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---

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
- Two baselines:
  a) random surface form
  b) most frequent form in training data
- 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---

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
- Two baselines:
  - a) random surface form
  - b) most frequent form in training data
- 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---

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
- Two baselines:
  a) random surface form
  b) most frequent form in training data
- 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---

Inflecting Proper Names

- Czech proper names & other DA slot values need to be inflected
- Generalized: selecting proper surface form
  - e.g., *obědvat* vs. *oběd*
- Two baselines:
  a) random surface form
  b) most frequent form in training data
- Two LM-based approaches:
  c) *n*-gram LM
  d) RNN LM
  - both give probability distribution over next token

  → select most probable surface form for current slot

?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---

Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***

Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově*

Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově*

inform(name="X-name", price_range=X-pricerange, area="X-area")
X-name je X-pricerange a nachází se v X-area.

Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově*
- 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.

Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***
- Some values require a specific sentence structure
  - ***v*** *Karlíně* vs. ***na*** *Smíchově*
- 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.

Using Lexical Values in DAs

- Different slot values exhibit different morphological behavior
  - *Ananta je levn**á*** vs. *BarBar je levn**ý***
- Some values require a specific sentence structure
  - ***v** Karlíně* vs. ***na** Smíchově*
- Keep values in input DAs (don't delexicalize)
  - still generating delexicalized outputs
- This is proof-of-concept
  - using the fact that number of different items is small
  - 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.

Experiments on Our Datset: BLEU/NIST

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

Experiments on Our Datset: BLEU/NIST

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

Experiments on Our Datset: BLEU/NIST

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

Human Evaluation

Human Evaluation

- Thank you! (/ pending, sorry)

Human Evaluation

- Thank you! (/ pending, sorry)
- Using WMT style multi-way relative comparisons
  - overall preference (no criteria)
  - selected setups only
- TrueSkill™ rating, bootstrap clustering

Human Evaluation

- Thank you! (/ pending, sorry)
- Using WMT style multi-way relative comparisons
  - overall preference (no criteria)
  - selected setups only
- TrueSkill$^{\text{TM}}$ rating, bootstrap clustering

Setup input DAs	Setup generator mode	Setup lexicalization	True Skill	Rank	BLEU
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

Data Inspection

- Different results for automatic vs. human scores
- Comparing "Best BLEU" vs. "Most preferred" on a sample

Data Inspection

- Different results for automatic vs. human scores
- Comparing “Best BLEU” vs. “Most preferred” on a sample
- Counting different error types:

lexicalization: *Restaurace Švejk je levná podnik blízko Stromovky*

Data Inspection

- Different results for automatic vs. human scores
- Comparing “Best BLEU” vs. “Most preferred” on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*

Data Inspection

- Different results for automatic vs. human scores
- Comparing "Best BLEU" vs. "Most preferred" on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*
structure:	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.*

Data Inspection

- Different results for automatic vs. human scores
- Comparing “Best BLEU” vs. “Most preferred” on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*
structure:	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.*
semantic:	*Na Hradčany se nehodí 2 restaurace, které nejsou vhodné pro děti.*

Data Inspection

- Different results for automatic vs. human scores
- Comparing “Best BLEU” vs. “Most preferred” on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*
structure:	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.*
semantic:	*Na Hradčany se nehodí 2 restaurace, které nejsou vhodné pro děti.*
punctuation:	*Děkuji a přeji krásný den*

Data Inspection

- Different results for automatic vs. human scores
- Comparing "Best BLEU" vs. "Most preferred" on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*
structure:	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.*
semantic:	*Na Hradčany se nehodí 2 restaurace, které nejsou vhodné pro děti.*
punctuation:	*Děkuji a přeji krásný den*

- Very similar performance (22 vs. 24 errors)

Data Inspection

- Different results for automatic vs. human scores
- Comparing "Best BLEU" vs. "Most preferred" on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*
structure:	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.*
semantic:	*Na Hradčany se nehodí 2 restaurace, které nejsou vhodné pro děti.*
punctuation:	*Děkuji a přeji krásný den*

- Very similar performance (22 vs. 24 errors)
  - most preferred: often just punctuation
  - ignoring punctuation: 20 vs. 16

Data Inspection

- Different results for automatic vs. human scores
- Comparing “Best BLEU” vs. “Most preferred” on a sample
- Counting different error types:

lexicalization:	*Restaurace Švejk je levná podnik blízko Stromovky*
fluency:	*Cenu do restaurace U Konšelů můžete volat na číslo 242817033.*
structure:	*V nabídce je 3 restaurací, které nabízí všechny druhy jídel.*
semantic:	*Na Hradčany se nehodí 2 restaurace, které nejsou vhodné pro děti.*
punctuation:	*Děkuji a přeji krásný den*

- Very similar performance (22 vs. 24 errors)
  - most preferred: often just punctuation
  - ignoring punctuation: 20 vs. 16
- “Most preferred” setup slightly better

Our System…

✓ works with unaligned data
  - better than our previous work on the BAGEL set

Our System…

- ✓ works with unaligned data
  - better than our previous work on the BAGEL set
- ✓ produces valid outputs even with limited training data

Our System…

- ✓ works with unaligned data
  - better than our previous work on the BAGEL set
- ✓ produces valid outputs even with limited training data
- ✓ allows comparing 2-step & joint NLG
  - generates sentences / trees

Our System…

- ✓ works with unaligned data
  - better than our previous work on the BAGEL set
- ✓ produces valid outputs even with limited training data
- ✓ allows comparing 2-step & joint NLG
  - generates sentences / trees
- ✓ is 1st trainable & capable of entrainment
  - entrainment better than baseline

Our System…

- ✓ works with unaligned data
  - better than our previous work on the BAGEL set
- ✓ produces valid outputs even with limited training data
- ✓ allows comparing 2-step & joint NLG
  - generates sentences / trees
- ✓ is 1st trainable & capable of entrainment
  - entrainment better than baseline
- ✓ works on Czech successfully
  - including proper name inflection

Future Work Ideas

- Remove delexicalization

Our System…

- ✓ works with unaligned data
  - better than our previous work on the BAGEL set
- ✓ produces valid outputs even with limited training data
- ✓ allows comparing 2-step & joint NLG
  - generates sentences / trees
- ✓ is 1st trainable & capable of entrainment
  - entrainment better than baseline
- ✓ works on Czech successfully
  - including proper name inflection

Future Work Ideas

- Remove delexicalization
- Integrate into an end-to-end SDS

Thank you for your attention

Download it!

- 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

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: 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	$\sim$59%
Lexical	$\sim$31%
Both	$\sim$19%

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

Czech Output Examples

Input DA	inform(name="Švejk Restaurant", near=Stromovka, price_range=cheap)
Reference	Restaurace Švejk je poblíž Stromovky a nabízí nízké ceny.
Best BLEU/NIST	Restaurace Švejk je levná restaurace, poblíž Stromovky.
Most preferred	Restaurace Švejk je levná podnik blízko Stromovky
Input DA	inform(name="U Konšelů", phone=X-phone)
Reference	Do U Konšelů můžete zavolat na telefonu 242817033.
Best BLEU/NIST	Cenu do restaurace U Konšelů můžete volat na číslo 242817033.
Most preferred	Telefonní číslo U Konšelů je 242817033.
Input DA	inform(type=restaurant, good_for_meal=dont_care, count=3)
Reference	Jsou zde 3 restaurace, které jsou vhodné pro jakékoli jídlo
Best BLEU/NIST	Našla jsem 3 restaurací, které jsou vhodné pro jakékoli jídlo.
Most preferred	V nabídce je 3 restaurací, které nabízí všechny druhy jídel.
Input DA	inform_no_match(area=Vinohrady)
Reference	Lituji, ale v lokalitě Vinohrad se nenachází žádné restaurace.
Best BLEU/NIST	Na Vinohradech nejsou žádné restaurace vhodné pro děti.
Most preferred	V Vinohrad nejsou žádné takové restaurace.
Input DA	inform(area=Hradčany, type=restaurant, kids_allowed=no, count=2)
Reference	V lokalitě Hradčan jsem našla 2 restaurace, které nedovolují vstup dětem.
Best BLEU/NIST	V oblasti Hradčan se nabízí 2 restaurace, které nejsou vhodné pro děti.
Most preferred	Na Hradčany se nehodí 2 restaurace, které nejsou vhodné pro děti.