Generating Natural Language from OWL and RDF Data
Grammar-Based, Statistical and Neural Approaches

Claire Gardent
CNRS/LORIA and Université de Lorraine, Nancy

CNL, August 2018
Maynooth, Ireland
Joint Work with

(a) Bikash Gyawali  
(b) Shashi Narayan  
(c) Laura Perez-Beltrachini  
(d) Anastasia Shimorina

Funded by the French ANR Project WebNLG
http://webnlg.loria.fr/pages/index.html
Much information is stored in KB and RDF stores.

User Survey: 72% of Internet users find it frustrating to get irrelevant information when web searching.
Source: www.internetsociety.org/survey

Claire Gardent
Generating Natural Language from OWL
Natural Language Generation makes this data accessible

**QUERYING**
Quelo: NLG allows the user to query a Knowledge Base in English
SUMMARISING
Miakt: NLG generates a patient report from an RDF data store.

Fig. 1. The MIAKT Generator
Natural Language Generation makes this data accessible

**VERBALISING**

SWAT: NLG translates the content of an OWL Knowledge Base into English

<table>
<thead>
<tr>
<th>Class label</th>
<th>OWL axioms (Manchester syntax)</th>
<th>Natural Language Definition Extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>22rv1</td>
<td>bearer_of some 'prostate carcinoma' derives_from some 'Homo sapiens' derives_from some prostate</td>
<td>A 22rv1 is a cell line. A 22rv1 is all of the following: something that is bearer of a prostate carcinoma, something that derives from a homo sapiens, and something that derives from a prostate.</td>
</tr>
<tr>
<td>HeLa</td>
<td>bearer_of some 'cervical carcinoma' derives_from some 'Homo sapiens' derives_from some cervix derives_from some 'epithelial cell'</td>
<td>A he la is a cell line. A he la is all of the following: something that is bearer of a cervical carcinoma, something that derives from a homo sapiens, something that derives from an epithelial cell, and something that derives from a cervix.</td>
</tr>
<tr>
<td>Ara-C-resistant murine leukemia</td>
<td>has subclass b117h*  has subclass b140h*</td>
<td>A ara c resistant murine leukemia is a cell line. A b117h, and a b140h are kinds of ara c resistant murine leukemias.</td>
</tr>
<tr>
<td>GM18507</td>
<td>derives_from some 'Homo sapiens' derives_from some lymphoblast has_quality some male</td>
<td>A gm18507 is all of the following: something that has as quality a male, something that derives from a homo sapiens, and something that derives from a lymphoblast.</td>
</tr>
</tbody>
</table>
Given a set of relations selected from the AURA knowledge base, generate a sentence that is grammatical and fluent in English.

The rate of absorption of a central vacuole is directly proportional to the size of the vacuole.
The KBGen Shared Task

Small Training Corpus: 207 training instances (data/text pairs)

3 Participants:

- UDEL: Hand Written Rule Based System (U. Delaware)
- IMS: Statistical System using a probabilistic grammar induced from the training data (U. Stuttgart)
- LOR-KBGEN: Grammar induced from the training data (Lorraine U.)
A Tree Adjoining Grammar (TAG) is automatically induced from the training corpus.

Each grammar rule:
- captures a semantically coherent unit, *Semantic Principle*
- groups syntactic functors with their dependents, *Extended Domain of Locality*

B. Gyawali and C. Gardent
Surface Realisation from Knowledge-Bases.
ACL 2014. Baltimore, USA.
Grammar-Based Generation
Grammar-Based Generation

The train \( \text{train}(A) \) departs \( \text{departure}(B,C) \) at 10am \( \text{tenAM}(D) \).
The train departs at 10am.

Input data: \(\text{train}(t), \text{departure}(e,t), \text{tenAM}(e)\)
The train \( \text{train}(t) \) departs \( \text{departure}(e,t) \) at 10am \( \text{tenAM}(e) \).

Input data: \( \text{train}(t), \text{departure}(e,t), \text{tenAM}(e) \)
Grammar-Based Generation

S
  /\  \
NP   VP
   /\   /\ \
| The train | V[agr:3sg] departs departure(e,t)
  | train(t) |
\VP*^e\PP
   /\   /\ \
  P   NP
   /\   /\ \
  at  10am tenAM(e)
Grammar-Based Generation

```
S
  NP
    The train
    train(t)
  VP
    VP
      V[agr:3sg]
      departs
departure(e,t)
    PP
      at
      10am
tenAM(e)
```
Grammar-Based Generation

The train departs at 10am

The train departs at 10am
Separating Grammar from Lexicon

Since each tree is lexicalised, the resulting grammar can be very large. In practice, we therefore
Since each tree is lexicalised, the resulting grammar can be very large. In practice, we therefore

- abstract over lexical items in the grammar
Since each tree is lexicalised, the resulting grammar can be very large. In practice, we therefore

- abstract over lexical items in the grammar
- use a lexicon to determine which grammar tree is lexicalised/anchored by which lexical items
Separating Grammar from Lexicon

S

NP\downarrow^C \quad \text{VP}^B

V

departs
department(B,C)
Separating Grammar from Lexicon

S
   NP\(^C\)   VP\(^B\)
       \(^V\)
           \(\text{departs} \quad \text{departure}(B,C)\)
Separating Grammar from Lexicon

Grammar-based, Statistical and Neural Approaches
Separating Grammar from Lexicon

Semantics: departure
Tree: nx0V
Syntax: CanonicalSubject
Anchor: departs

departure(B,C)

Semantics: arrival
Tree: nx0V
Syntax: CanonicalSubject
Anchor: arrives

...
Inducing a Grammar from the KBGen Data

For each (data, sentence) pair in the input:

- Parse and Align semantic variables with words
- Project variables up the parse tree
- Extract subtrees (create a grammar)
- Split trees (generalise)
The function of a gated channel is to release particles from the endoplasmic reticulum.
Step 1: Parsing and Variable/Word Alignment

- gated_channel(GC)
- release_of_calcium(RoC)
- particles(P)
- reticulum(R)
Step 2: Variable Projection

- gated_channel(GC)
- release_of_calcium(RoC)
- particles(P)
- reticulum(R)
Step 3: Tree Extraction (Entities)

NP[\textit{idx}=\textit{GC}]

\begin{align*}
\text{DT} & \quad \text{JJ} & \quad \text{N} \\
a & \quad \text{gated} & \quad \text{channel}
\end{align*}

\textit{gated\_channel(GC)}

NP[\textit{idx}=\textit{P}]

\begin{align*}
\text{N} & \\
particles
\end{align*}

\textit{particles(P)}

NP[\textit{idx}=\textit{R}]

\begin{align*}
\text{DT} & \quad \text{N} \\
\text{the} & \quad \text{reticulum}
\end{align*}

\textit{reticulum(R)}
Step 3: Tree Extraction (Events)

```
S
  NP
    NP
      DT
      the
    N
    function
    P
    of
  PP
    NP\[idx=GC\]
  V
    \[idx=RoC\]
  S
    VP
      V\[idx=RoC\]
      NP\[idx=P\]
    TO
      to
    V\[idx=RoC\]
    NP\[idx=P\]
    PP
      P
      NP\[idx=R\]
      from
```

- `Release_Of_Calcium(RoC)`
- `object(RoC,P)`
- `base(RoC,R)`
- `agent(RoC,GC)`
- `has_function(GC,RoC)`

Claire Gardent

Generating Natural Language from OWL
We further extract from each Event tree, subtrees corresponding to Subject-Verb-Object structure and optional modifiers.

```
S_{E_3}
   ________________
   |               |
   |               |
   NP             VP_{E_3}^{E_2}
   /       \       /     \
  NP      PP      VBZ          
  /     \      /         |
DT NN IN  NP_{E_1}^{E_2}  is
  /     \           /     \
the fn of     S_{E_1}^{E_2}  

TO VB E  NP A
  /     \   /     \   
  to     release to     
  /     \   /     \
IN      PP     

release(E) object(E,A)
agent(E,C) has-function(C,E)
```
Step 4: Splitting Trees

The function of C is to release A from B

The function of C is to release A

Release(E) object(E,A)
agent(E,C) has_function(C,E)

base(E,B)
Step 4: Splitting Trees

The function of C is to release A from B

The function of C is to release A
Step 4: Splitting Trees

The function of C is to release A from B
The function of C is to release A
The function of C is to release A to B
Evaluation and Results

- 72 inputs from KBGEN
- Automatic Evaluation: BLEU
- Human-Based Evaluation
  - 12 participants were asked to rate sentences along three dimensions:
    - **fluency**: Is the text easy to read?
    - **grammaticality**: Is the text grammatical?
    - **adequacy**: Does the meaning conveyed by the generated sentence correspond to the meaning conveyed by the reference sentence?
  - Online evaluation (LG-Eval toolkit)
  - Subjects used a sliding scale (1 to 5)
  - Latin Square Experimental Design was used to ensure that each evaluator sees the same number of output from each system and for each test set item.
Results

Automatic Metrics

Human Evaluation

BLEU

UDEL
LOR-KBGen
IMS

Fluency
Grammaticality
Meaning

UDEL
LOR-KBGen
IMS

Generating Natural Language from OWL
Conclusion

Linguistically guided grammar induction:

- permits a fully automated approach (unlike the UDEL system)
- yields output sentences whose quality is close to those produced by a hand written system (unlike the IMS system)
Using NLG to query a KB

Interactive refinement of the user query

- Possible (consistent with KB) extensions of the current user query are computed by an automated reasoner $\Rightarrow$ Set of DL formulae ($F$)
- Each formal extension ($f \in F$) is then verbalised using NLG
- N.B. The user may revise (substitute, delete, add) the current query

L. Perez-Beltrachini and C. Gardent
Incremental Query Generation

C. Gardent and L. Perez-Beltrachini
A Statistical, Grammar-Based Approach to Micro-Planning
Input = KB Query

Professor ⊓ Researcher ⊓ ∃teach.LogicCourse
⊓ ∃worksAt.AlicanteUniversity

I am looking for a professor who is a researcher and teaches a course on logic.
He should work for Alicante University.

**Microplanning Task:** Segment, lexicalise, aggregate and realise
A Statistical Grammar-Based Approach

The grammar

- Enforces grammaticality
- Accounts for language variability (paraphrasing)

The Statistical Module (Hypertagger)

- Enforces microplanning choices (fluency)
- Enhances efficiency (speed)
The Generation Algorithm

- **Lexical Selection**: retrieves TAG trees whose semantic subsumes the input and which are compatible with the hypertagger decisions.
- **Hypertagging**: Selects the n-best sequences of grammar rules (TAG trees) given the input semantics.
- **Surface Realisation**: Combines TAG trees to produce Sentences.
Grammar and Lexicon

The lexicon

- relates KB Symbols, Natural Language Expressions and Syntax (Grammar rules). It is domain specific.
- is acquired automatically

The grammar

- specifies the various syntactic realisations of words. It is generic.
- is a small, manually specified Tree Adjoining Grammar
Automatic Lexicon Induction

The lexicon is automatically derived from KB symbols (Trevisan 2010)

<table>
<thead>
<tr>
<th>Step 1: Tokenize and PoS Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>runsOn → runs/VBD on/IN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2: The result sequence is mapped to one or more Lexical Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>runs/VBD on/IN →</td>
</tr>
<tr>
<td>Semantics: runsOn</td>
</tr>
<tr>
<td>Tree: nx0Vpnx1</td>
</tr>
<tr>
<td>Anchor: should run</td>
</tr>
<tr>
<td>Co-Anchor: P → on</td>
</tr>
</tbody>
</table>
Generic Grammar

A small (100 trees), hand-written generic grammar models subcategorisation and syntactic variation.

Valency/Subcategorisation Variations

NP₀ should generate NP₁  \( nx₀VVnx₁ \)  Canonical
NP₀ should run on NP₁  \( nx₀VVpnx₁ \)  Canonical
NP₀ should be equipped with NP₁  \( nx₀VVVpnx₁ \)  Canonical
NP₀ should be the equipment of NP₁  \( nx₀VVDNpnx₁ \)  Canonical
NP₀ should have access to NP₁  \( nx₀VVNpnx₁ \)  Canonical
NP₀ should be relevant to NP₁  \( nx₀VVApnx₁ \)  Canonical
NP₀ should be an N₁ product  \( nx₀VVDNnx₁ \)  Canonical
NP₀ with NP₁  \( betanx₀Pnx₁ \)  Canonical
Generic Grammar

Syntactic Variations

NP₀ should be equipped with NP₁
and NP₀ should be equipped with NP₁
NP₀ which should be equipped with NP₁
NP₀ (...) and which should be equipped with NP₁
NP₀ (...), which should be equipped with NP₁
NP₀ equipped with NP₁
NP₀ (...) and equipped with NP₁
NP₀ (...), equipped with NP₁
NP₁ with which NP₀ should be equipped
NP₀ (equipped with X) and with NP₁
NP₀ (equipped with X), with NP₁

Canonical
S-Coordination
SubjRel
SubjRelPU
SubjRelPU
PpartOrGerund
SharedSubj
SharedSubj
PObjRel
Ellipsis
Ellipsis

A small (100 trees), hand-written generic grammar models subcategorisation and syntactic variation.
For a given KB symbol, the grammar models multiple syntactic realisations of that symbol

<table>
<thead>
<tr>
<th>CarDealer(X)</th>
<th>locatedIn(X,Y)</th>
<th>City(Y)</th>
<th>sell(Y,Z)</th>
<th>Car(Z)</th>
<th>runOn(Z,W)</th>
<th>Diesel</th>
</tr>
</thead>
<tbody>
<tr>
<td>nx</td>
<td>nx0VVVpnx1</td>
<td>nx</td>
<td>nx0VVVnx1</td>
<td>nx</td>
<td>nx0VVVpnx1</td>
<td>nx</td>
</tr>
<tr>
<td>PRO0VVVpnx1</td>
<td>sCONJnx0VVVpnx1</td>
<td>sCONJnx0VVVnx1</td>
<td>sCONJPRO0VVVpnx1</td>
<td>W0nx0VVVpnx1</td>
<td>ANDWHnx0VVVpnx1</td>
<td>COMMAWHnx0VVVpnx1</td>
</tr>
<tr>
<td></td>
<td>W0nx0VVVpnx1</td>
<td>ANDWHnx0VVVnx1</td>
<td>sCONJPRO0VVVnx1</td>
<td>W0nx0VVVnx1</td>
<td>ANDWHnx0VVVnx1</td>
<td>COMMAWHnx0VVVnx1</td>
</tr>
<tr>
<td></td>
<td>betanx0VPpnx1</td>
<td>betanx0VPpnx1</td>
<td>betanx0ANDVPpnx1</td>
<td>betanx0COMMAVPpnx1</td>
<td>W1pnx1nx0VV</td>
<td>betavx0ANDVVVpnx1</td>
</tr>
<tr>
<td></td>
<td>betanx0ANDVPpnx1</td>
<td>betanx0ANDVPpnx1</td>
<td>betanx0COMMAVPpnx1</td>
<td>W1pnx1nx0VV</td>
<td>betavx0ANDVVVnx1</td>
<td>betavx0COMMAVVVpnx1</td>
</tr>
<tr>
<td></td>
<td>betanx0COMMAVPpnx1</td>
<td>betanx0COMMAVPpnx1</td>
<td>betanx0COMMAVPpnx1</td>
<td>W1pnx1nx0VV</td>
<td>betavx0ANDVVVnx1</td>
<td>betavx0COMMAVVVnx1</td>
</tr>
</tbody>
</table>

I am looking for a car dealer located in a city who should sell cars. The car should run on diesel.
The **hypertagger** prunes the initial search space and favours Tree/Syntactic Classes sequences which yield fluent sentences.

CarDealer ⊓ ∃locatedIn.City ⊓ ∃sell.Car ⊓ ∃runOn.Diesel

Tbetanx0VPpnx1 TANDWHn0x0VVnx1 Tnx0VVpnx1 Tnx
I am looking for a car dealer located in a city and who should sell a car. The car should run on diesel.

Tnx0VPpnx1 Tnx0VVVnx1 Tnx0VVVpnx1
I am looking for a car dealer. The car dealer should be located in a city. The car dealer should sell a car. The car should run on diesel.
Making Choices (Hypertagging)

CRF Hypertagging Model

We learn a linear-chain CRF model to predict the mapping between observed input features and hidden syntactic labels $y = \{y_1, \ldots, y_L\}$.

$$P(y \mid x) = \frac{1}{Z(x)} \prod_{l=1}^{L} \exp \sum_{k=1}^{K} \theta_k \Phi_k(y_{l-1}, x)$$ (1)

The hypertagger finds the optimal hypertag sequence $y^*$ for a given input semantics $x$:

$$y^* = \arg\max_y P(y \mid x)$$
Data

Training Data for the CRF

- 206 training instances = (KB query, tree sequence) pairs
- From 11 ontologies (Domain Independent)
- Input Length (min:2, max:19, avg: 7.44)
- CRF trained and tested using 10 fold cross validation

Features

- KB Symbol: Shape and content (words) of relation names (unigram and bigrams)
- Lexical features: word overlap between KB symbols, presence/absence of prepositions, etc.
- Entity Chaining Features: distribution of discourse entities in the input query
- Structural features: length of the input, number of predications over the same entity ...
Experimental Setup

Grammar and Lexicon

- Grammar: 69 trees, 10 syntactic classes
- Lexicon: 13 KB, 10K entries, 1296 concepts and elations, average lexical ambiguity: 7.73.

Evaluation Metrics

- Hypertagging Accuracy
- Coverage and Speed
- Output quality (Human Evaluation)
- Qualitative Analysis (Microplanning)

Comparison Models

- Template-Based Model
- Symbolic Grammar-Based Model
Results: Hypertagging Accuracy

- Token Accuracy
  - Trees
  - Syntactic Classes

- Sequence Accuracy
  - Trees
  - Syntactic Classes

Claire Gardent
Generating Natural Language from OWL
Results: Coverage

![Coverage Chart]

Claire Gardent
Generating Natural Language from OWL
Results: Speed

- **Time (in ms)**
  - Trees
  - Syntactic Classes

**Trees**
- One: 0, 500
- Five: 1,000
- Ten: 1,500

**Syntactic Classes**
- One: 0, 500
- Five: 1,000
- Ten: 2,000
Results: Output quality

**Human Evaluation**

- 48 input queries
- from 13 knowledge bases (2 not used in training corpus)
- 24 raters
- Online evaluation
- Sliding ruler
- Scale 0-50
- Latin Square design
Results: Output quality

- Clarity
- Fluency

Human Score

Template vs Hybrid

Symbolic vs Hybrid

Generating Natural Language from OWL
Results: Output quality (BLEU Scores)

- **ALL**
  - BLEU Score: 0.2, 0.4, 0.6, 0.8

- **Generated**
  - BLEU Score: 0.2, 0.4, 0.6, 0.8

Legend:
- Blue: Templates
- Green: Symbolic
- Red: Hybrid
**Example Output: Sentence Segmentation**

3 relations, 4 concepts: 1 sentence

*I am looking for a used car whose color should be white, which should be located in a France and whose model should be a toyota 4 runner.*

4 relations, 5 concepts: 2 sentences

*I am looking for a new car whose exterior color should be beige and whose body style should be a utility vehicle. The new car should run on a natural gas and should be located in a country.*

3 relations, 5 concepts: 2 sentences

*I am looking for a new car whose body style should be a utility vehicle, an off road. The new car should run on a natural gas and should be located in a country.*
I am looking for a car dealer located in a country and who should sell a car whose make should be a toyota. The car should run on a fuel and should be equipped with a manual gear transmission system. (Participial)

I am looking for a car dealer who sells a car whose model is a toyota. It should be located in a country. (Sentence with Pronominal Subject)

I am looking for a new car, an off road whose body style should be a utility vehicle. The new car should run on a natural gas and should be located in a country. (Coordinated VP)

I am looking for a car produced by a car make. The car make should be the make of a toyota. The car make should be located in a city and should produce a land rover freelander. (Canonical Declarative Sentence)
Example Output: Aggregation

**VP Coordination**

\[ \text{NewCar (\ldots)} \sqcap \exists \text{runOn.NaturalGas} \sqcap \exists \text{locatedInCountry.Country} \]

*I am looking for a new car (\ldots). This new car (should run on natural gas and should be located in a country)\_VP.*

**Relative Clause Coordination**

\[ \text{CommunicationDevice} \sqcap \exists \text{assistsWith.Understanding} \sqcap \exists \text{assistsWith.HearingDisability} \]

*I am looking for a communication device (which should assist with a understanding and which should assist with a hearing disability)\_RelCl.*
**NP Coordination**

CarDealer □∃sell.CrashCar □∃sell.NewCar

*I am looking for a car dealer who should sell (a crash car and a new car)*$_{NP}$.

**N-Ary NP Coordination**

Car □∃equippedWith.ManualGearTransmission
□∃equippedWith.AirBagSystem

*I am looking for a car equipped with (a manual gear transmission system, an alarm system, a navigation system and an air bag system)*$_{NP}$.
Summary

Ambiguous Grammar = High Expressivity, Large Search Space

Hypertagging = Making Choices
[NLG]
Provide a benchmark on which to train, evaluate and compare microplanners for data-to-text generation.

[Semantic Web]
Train, evaluate and compare verbalisers for RDF Data
John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot.

- Generating Referring Expressions: Describing entities
- Lexicalisation: Choosing lexical items
- Surface Realisation: Choosing syntactic structures
- Aggregation: Avoiding repetition
- Sentence segmentation: Segmenting the content into sentence size chunks
Creating the WebNLG Dataset

- RDF KB (DBPedia) → Content Selection → Data
- Text produced by crowdworkers

<table>
<thead>
<tr>
<th>WebNLG</th>
<th># data-text pairs</th>
<th>40,049</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># distinct inputs</td>
<td>15,095</td>
</tr>
<tr>
<td></td>
<td># DBPedia Categories</td>
<td>15</td>
</tr>
</tbody>
</table>

Laura Perez-Beltrachini, Rania Mohammed Sayed and Claire Gardent
Building RDF Content for Data-to-Text Generation
*COLING*, 2016.

Claire Gardent, Anastasia Shimorina, Shashi Narayan and Laura Perez-Beltrachini
Creating Training Corpora for NLG Micro-Planning
Training and Testing Data

- Train/Dev/Test split: 80/10/10
- 10 seen categories: Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam, City and WrittenWork
- 5 unseen categories: Athlete, Artist, MeanOfTransportation, CelestialBody, Politician

<table>
<thead>
<tr>
<th></th>
<th>Train+Dev</th>
<th>Test Seen</th>
<th>Test Unseen</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entries</td>
<td>7,812</td>
<td>971</td>
<td>891</td>
<td>9,674</td>
</tr>
<tr>
<td>Data/text pairs</td>
<td>20,370</td>
<td>2,495</td>
<td>2,433</td>
<td>25,298</td>
</tr>
</tbody>
</table>
The Participants

61 downloads, 6 participants, 8 systems

3 Pipeline Systems
Tilb-Pipeline, UIT-VNU and UPF-FORGe

1 SMT-Based System
Tilb-SMT

5 Neural-Based Systems
ADAPT, Melbourne, PKUWriter, Tilb-NMT and Baseline
## Pipeline Systems

<table>
<thead>
<tr>
<th></th>
<th>Order</th>
<th>Aggr.</th>
<th>Templ.</th>
<th>REG</th>
<th>Gr.</th>
<th>re-ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TILB-Pipeline</strong></td>
<td>+</td>
<td>-</td>
<td>Induced</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td><strong>UIT-VNU</strong></td>
<td>-</td>
<td>-</td>
<td>Induced</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>UPF-FORGe</strong></td>
<td>+</td>
<td>+</td>
<td>Manual</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>
# Seq2seq Systems

<table>
<thead>
<tr>
<th></th>
<th>Pre-processing</th>
<th>Word Repr</th>
<th>Add. Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tilb-NMT</strong></td>
<td>Delex</td>
<td></td>
<td>REG Module</td>
</tr>
<tr>
<td><strong>PKUWriter</strong></td>
<td>Delex and</td>
<td>Glove vectors</td>
<td>Rerank</td>
</tr>
<tr>
<td><strong>Melbourne</strong></td>
<td>Sem Typing</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ADAPT</strong></td>
<td>Tokenize RDF</td>
<td>Subwords</td>
<td></td>
</tr>
</tbody>
</table>
Global Results

<table>
<thead>
<tr>
<th>BLEU</th>
<th></th>
<th>METEOR</th>
<th></th>
<th>TER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Melbourne</td>
<td>45.13</td>
<td>UPF-FORGe</td>
<td>0.39</td>
<td>Melbourne</td>
<td>0.47</td>
</tr>
<tr>
<td>Tilb-SMT</td>
<td>44.28</td>
<td>Tilb-SMT</td>
<td>0.38</td>
<td>Tilb-SMT</td>
<td>0.53</td>
</tr>
<tr>
<td>PKUWriter</td>
<td>39.88</td>
<td>Melbourne</td>
<td>0.37</td>
<td>PKUWriter</td>
<td>0.55</td>
</tr>
<tr>
<td>UPF-FORGe</td>
<td>38.65</td>
<td>Tilb-NMT</td>
<td>0.34</td>
<td>UPF-FORGe</td>
<td>0.55</td>
</tr>
<tr>
<td>Tilb-Pipeline</td>
<td>35.29</td>
<td>AdapT</td>
<td>0.31</td>
<td>Tilb-Pipeline</td>
<td>0.56</td>
</tr>
<tr>
<td>Tilb-NMT</td>
<td>34.60</td>
<td>PKUWriter</td>
<td>0.31</td>
<td>Tilb-NMT</td>
<td>0.60</td>
</tr>
<tr>
<td>Baseline</td>
<td>33.24</td>
<td>Tilb-Pipeline</td>
<td>0.30</td>
<td>Baseline</td>
<td>0.61</td>
</tr>
<tr>
<td>AdapT</td>
<td>31.06</td>
<td>Baseline</td>
<td>0.23</td>
<td>Uit-Vnu</td>
<td>0.82</td>
</tr>
<tr>
<td>Uit-Vnu</td>
<td>7.07</td>
<td>Uit-Vnu</td>
<td>0.09</td>
<td>AdapT</td>
<td>0.84</td>
</tr>
</tbody>
</table>

- 6 systems above the baseline (4 well above it)
- Neural NLG
  - Glove vectors and semantic typing of entities help (**Melbourne**)
  - Relexicalisation works better than subwords (**AdapT**)

Generating Natural Language from OWL
Neural and SMT systems are better are “reproducing” seen data

Rule based systems (UPF-FORGe, Tilb-Pipeline) seems to produce text that is more different from references than learned systems (higher METEOR and TER)
Results on Unseen Categories

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th>METEOR</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPF-FORGe</td>
<td>35.70</td>
<td>0.37</td>
<td>0.55</td>
</tr>
<tr>
<td>Melbourne</td>
<td>33.27</td>
<td>0.33</td>
<td>0.55</td>
</tr>
<tr>
<td>Tilb-SMT</td>
<td>29.88</td>
<td>0.31</td>
<td>0.61</td>
</tr>
<tr>
<td>PKUWriter</td>
<td>25.36</td>
<td>0.24</td>
<td>0.65</td>
</tr>
<tr>
<td>Tilb-NMT</td>
<td>25.12</td>
<td>0.21</td>
<td>0.67</td>
</tr>
<tr>
<td>Tilb-Pipeline</td>
<td>20.65</td>
<td>0.19</td>
<td>0.72</td>
</tr>
<tr>
<td>ADAPT</td>
<td>10.53</td>
<td>0.07</td>
<td>0.80</td>
</tr>
<tr>
<td>Baseline</td>
<td>06.13</td>
<td>0.03</td>
<td>0.87</td>
</tr>
<tr>
<td>UIT-VNU</td>
<td>0.11</td>
<td></td>
<td>1.4</td>
</tr>
</tbody>
</table>

- **UPF-FORGe** performs well on unseen data and much better than most neural systems.
And also

**NLG**
- Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang and Wei Wang
  GTR-LSTM: A Triple Encoder for Sentence Generation from RDF Data.
  *ACL*, 2018.
- Emiel Krahmer, Thiago Castro Ferreira, Sander Wubben, Ákos Kádár and Diego Moussallem
  NeuralREG: An end-to-end approach to referring expression generation.
  *ACL*, 2018.
- Emilie Colin and Claire Gardent.
  Generating Syntactic Paraphrases.
  *EMNLP*, 2018.

**Sentence Simplification**
- Shashi Narayan, Claire Gardent, Shay Cohen and Anastasia Shimorina
  Split and Rephase
- Roee Aharoni and Yoav Goldberg
  Split and Rephrase: Better Evaluation and a Stronger Baseline
  *ACL*, 2018.

**Relation Extraction**
- Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu and Jun Zhao
  Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism
  *ACL*, 2018.
What next?

- Better NLG models
- Other text types and communication goals
- Multilingual Generation
THANKS!