In this lecture

- Introduction to Natural Language Generation (NLG)
  - the use of corpora & statistical models in NLG

- Summarisation
  - Single-document
  - Multi-document

- Evaluation using corpora: BLEU/NIST/ROUGE and related metrics
Part 1

Natural Language Generation
What is NLG?

• NLG systems aim to produce understandable texts (in English or other languages) typically from non-linguistic input.

Examples:

• Automatic generation of weather reports.
  • Input: data in the form of numbers (Numerical Weather Prediction models)
  • Output: short text representing a weather forecast
  • Many systems developed in this domain.

• STOP:
  • generates smoking cessation letters based on a user-input questionnaire
  • [http://www.csd.abdn.ac.uk/research/stop/](http://www.csd.abdn.ac.uk/research/stop/)
Weather report example

S 8-13 increasing 18-23 by morning, then easing 8-13 by midnight.

S 8-13 increasing 13-18 by early morning, then backing NNE 18-23 by morning, and veering S 13-18 by midday, then easing 8-13 by midnight.

SUMTIME: http://cgi.csd.abdn.ac.uk/~ssripada/cgi_bin/startSMT.cgi
Other examples: story generation

- STORYBOOK (Callaway & Lester 2002):
  - input = story plan: sequential list of operators specifying underlying structure of a narrative

  (actor-property exist-being woodman001)
  (refinement and-along-with woodman001 wife001)
  (refinement belonging-to wife001 woodman001)
  (specification exist-being process-step-type once-upon-a-time)

- output:
  - Once upon a time there was a woodman and his wife.
 Dialogue fragment:

- **System1**: Welcome…. What airport would you like to fly out of?
- **User2**: I need to go to Dallas.
- **System3**: Flying to Dallas. What departure airport was that?
- **User4**: from Newark on September the 1st.

What should the system say next?

**Plan for next utterance**

(after analysis of User4)

- implicit-confirm(orig-city:NEWARK)
- implicit-confirm(dest-city:DALLAS)
- implicit-confirm(month:9)
- implicit-confirm(day-number:1)
- request(depart-time)

**Output next utterance:**

- What time would you like to travel on September the 1st to Dallas from Newark?

Walker et al. (2001). SPoT: A trainable sentence planner. *Proc. NAACL*
Types of input to an NLG system

- Raw data (e.g. Weather report systems):
  - Typical of **data-to-text** systems
  - These systems need to pre-analyse the data

- Knowledge base:
  - Symbolic information (e.g. database of available flights)

- Content plan:
  - representation of what to communicate (usually in some canonical representation)
  - e.g.: complete story plan (STORYBOOK)

- Other sources:
  - Discourse/dialogue history
    - Keep track of what’s been said to inform planning
NLG tasks & architecture
The architecture of NLG systems

- A pipeline architecture
  - represents a “consensus” of what NLG systems actually do
- very modular
- not all implemented systems conform 100% to this architecture
Concrete example

- BabyTalk systems (Portet et al 2009)
  - summarise data about a patient in a Neonatal Intensive Care Unit
  - main purpose: generate a summary that can be used by a doctor/nurse to make a clinical decision

A micro example

There were 3 successive bradycardias down to 69.

Input data: unstructured raw numeric signal from patient’s heart rate monitor (ECG)
A micro example: pre-NLG steps

(1) Signal Analysis (pre-NLG)
- Identify interesting patterns in the data.
- Remove noise.

(2) Data interpretation (pre-NLG)
- Estimate the importance of events
- Perform linking & abstraction

SEQUENCE (BRADYCARDIA)
- BRADYCARDIA (16:58:48) Imp: 31.64
- BRADYCARDIA (17:01:15) Imp: 79.80
- BRADYCARDIA (17:03:57) Imp: 80.21
- BRADYCARDIA (17:04:30) Imp: 39.97
- BRADYCARDIA (17:05:01) Imp: 34.60
- BRADYCARDIA (17:08:03) Imp: 68.24
Document planning/Content Selection

• **Main tasks**
  - Content selection
  - Information ordering

• **Typical output is a document plan**
  - tree whose leaves are messages
  - nonterminals indicate rhetorical relations between messages (Mann & Thompson 1988)
    - e.g. justify, part-of, cause, sequence…
A micro example: Document planning

(1) Signal Analysis (pre-NLG)
- Identify interesting patterns in the data.
- Remove noise.

(2) Data interpretation (pre-NLG)
- Estimate the importance of events
- Perform linking & abstraction

(3) Document planning
- Select content based on importance
- Structure document using rhetorical relations
- Communicative goals (here: assert something)
A micro example: Microplanning

- **Lexicalisation**
  - Many ways to express the same thing
  - Many ways to express a relationship
  - e.g. `SEQUENCE(x,y,z)`
    - x happened, then y, then z
    - x happened, followed by y and z
    - x,y,z happened
    - there was a sequence of x,y,z
  - Many systems make use of a lexical database.
A micro example: Microplanning

- **Aggregation:**
  - given 2 or more messages, identify ways in which they could be merged into one, more concise message
  - e.g. be(HR, stable) + be(HR, normal)
    - (No aggregation) *HR is currently stable. HR is within the normal range.*
    - (conjunction) *HR is currently stable and HR is within the normal range.*
    - (adjunction) *HR is currently stable within the normal range.*
A micro example: Microplanning

- Referring expressions:
  - Given an entity, identify the best way to refer to it
  - e.g. BRADYCARDIA
    - bradycardia
    - it
    - the previous one
  - Depends on discourse context! (Pronouns only make sense if entity has been referred to before)
A micro example

<table>
<thead>
<tr>
<th>Event</th>
<th>TYPE</th>
<th>existential</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRED</td>
<td>be</td>
</tr>
<tr>
<td></td>
<td>TENSE</td>
<td>past</td>
</tr>
<tr>
<td></td>
<td>ARGS</td>
<td>[THEME bradycardia, VALUE 69]</td>
</tr>
</tbody>
</table>

(4) Microplanning
Map events to semantic representation
- **lexicalise**: bradycardia vs sudden drop in HR
- **aggregate** multiple messages (3 bradycardias = one sequence)
- decide on how to refer (bradycardia vs it)
A micro example: Realisation

- **Subtasks:**
  - map the output of microplanning to a syntactic structure
  - needs to identify the best form, given the input representation
    - typically many alternatives
    - which is the best one?
  - apply inflectional morphology (plural, past tense etc)
  - linearise as text string
A micro example

(4) Microplanning
Map events to semantic representation
• lexicalise: bradycardia vs sudden drop in HR
• aggregate multiple messages (3 bradycardias = one sequence)
• decide on how to refer (bradycardia vs it)
• choose sentence form (there were…)

(5) Realisation
• map semantic representations to syntactic structures
• apply word formation rules

```
Event
TYPE existential
PRED be
TENSE past
ARGS [THEME bradycardia, VALUE 69]

(4) Microplanning
Map events to semantic representation
• lexicalise: bradycardia vs sudden drop in HR
• aggregate multiple messages (3 bradycardias = one sequence)
• decide on how to refer (bradycardia vs it)
• choose sentence form (there were…)

(5) Realisation
• map semantic representations to syntactic structures
• apply word formation rules

```
Rules vs statistics

- Many NLG systems are rule-based
- Growing trend to use statistical methods.

Main aims:
- increase linguistic coverage (e.g. of a realiser) “cheaply”
- develop techniques for fast building of a complete system
Using statistical methods

Language models and realisation
Advantages of using statistics

- Construction of NLG systems is extremely laborious!
  - e.g. BabyTalk system took ca. 4 years with 3-4 developers

- Many statistical approaches focus on specific modules
  - best-studied: statistical realisation
  - realisers that take input in some canonical form and rely on language models to generate output
  - advantage:
    - easily ported to new domains/applications
    - coverage can be increased (more data/training examples)
Overgeneration and ranking

- The approaches we will consider rely on “overgenerate-and-rank” approach:

- **Given**: input specification (“semantics” or canonical form)

  1. Use a simple rule-based generator to produce many alternative realisations.
  2. Rank them using a language model.
  3. Output the best (= most probable) realisation.
Advantages of overgeneration + ranking

- There are usually many ways to say the same thing.
- e.g. ORDER(eat(you, chicken))
  - Eat chicken!
  - It is required that you eat chicken!
  - It is required that you eat poulet!
  - Poulet should be eaten by you.
  - You should eat chicken/chickens.
  - Chicken/Chickens should be eaten by you.
Where does the data come from?

• Some statistical NLG systems were built based on parallel data/text corpora.
  • allows direct learning of correspondences between content and output
  • rarely available

• Some work relies on Penn Treebank:
  • Extract input: process the treebank to extract “canonical specifications” from parsed sentences
  • train a language model
  • re-generate using a realiser and evaluate against original treebank
Extracting input from treebank

- Penn treebank input:

```
(S (PP (IN Without))
  (NP (NNP GM)))
(, ,)
(NP-SBJ
  (NP (JJ overall) (NNS sales))
  (PP (IN for)
    (NP (DT the) (JJ other)
      (NNP U.S.) (NNS automakers))))
(VP (VBD were)
  (ADJP-PRD (RB roughly) (JJ flat)
    (PP (IN with)
      (NP (CD 1989) (NNS results ""))))))
```

Extracting input from treebank

- Converted into required input representation:

```
((cat clause)
  (circum ((accompaniment ((cat pp) (position front) (accomp-polarity -)
    (np ((cat proper) (lex "GM")))))))))
  (process ((type ascriptive) (tense past)))
  (participants ((carrier ((cat common) (lex "sale") (number plural)
    (descriptor ((cat adj) (lex "overall"))))))
    (qualifier ((cat pp) (prep ((lex "for"))))
      (np ((cat common) (lex "automaker") (definite yes)
        (number plural) (status different)
        (classifier ((cat proper) (lex "U.S.")))))))))
  (attribute ((cat ap) (lex "flat") (modifier ((cat adv) (lex "roughly"))))
    (qualifier ((cat pp) (prep ((lex "with"))))
      (np ((cat common) (lex "result") (number plural)
        (classifier ((cat date) (year 1989)))))))))
```

A case study

The NITROGEN/HALogen statistical realiser
Nitrogen and HALogen

- Pioneering realisation systems with wide coverage (i.e. handle many phenomena of English grammar)
- Based on overgeneration/ranking
- HALogen (Langkilde-Geary 2002) is a successor to Nitrogen (Langkilde 1998)
  - main differences:
    - representation data structure for possible realisation alternatives
    - HALogen handles more grammatical features
Structure of HALogen

Symbolic Generator
- Rules to map input representation to syntactic structures
  - Lexicon
  - Morphology

best sentence

Statistical ranker
- n-gram model (from Penn Treebank)

multiple outputs represented in a “forest”
HALogen Input

Grammatical specification
(e1 / eat
  :subject (d1 / dog)
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today))

Semantic specification
(e1 / eat
  :agent (d1 / dog)
  :patient (b1 / bone
    :premod(m1 / meaty))
  :temp-loc(t1 / today))

- Labeled feature-value representation specifying properties and relations of domain objects (e1, d1, etc)
- Recursively structured
- Order-independent
- Can be either grammatical or semantic (or mixture of both)
  - recasting mechanism maps from one to another
HALogen base generator

- Consists of about 255 hand-written rules
- Rules map an input representation into a packed set of possible output expressions.
  - Each part of the input is recursively processed by the rules, until only a string is left.
- Types of rules:
  1. recasting
  2. ordering
  3. filling
  4. morphing
Recasting

- Map semantic input representation to one that is closer to surface syntax.

**Semantic specification**
(e1 / eat
  :patient (b1 / bone
    :premod(m1 / meaty))
  :temp-loc(t1 / today)
  :agent (d1 / dog))

**IF** relation = :agent
  AND sentence is not passive
  **THEN** map relation to :subject

**Grammatical specification**
(e1 / eat
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today)
  :subject (d1 / dog))
Ordering

- Assign a linear order to the values in the input.

**Grammatical specification**
(e1 / eat
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today)
  :subject (d1 / dog))

**Put subject first unless sentence is passive.**
**Put adjuncts sentence-finally.**

**Grammatical specification + order**
(e1 / eat
  :subject (d1 / dog)
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today))
Filling

- If input is under-specified for some features, add all the possible values for them.
- NB: this allows for different degrees of specification, from minimally to maximally specified input.
- Can create multiple “copies” of same input

Grammatical specification + order
(e1 / eat
  :subject (d1 / dog)
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today))
Morphing

- Given the properties of parts of the input, add the correct inflectional features.

Grammatical specification + order
(e1 / eat
  :tense(past)
  :subject (d1 / dog)
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today))

Grammatical specification + order
(e1 / ate
  :subject (d1 / dog)
  :object (b1 / bone
    :premod(m1 / meaty))
  :adjunct(t1 / today))
The output of the base generator

- Problem:
  - a single input may have literally hundreds of possible realisations after base generation
  - these need to be represented in an efficient way to facilitate search for the best output

- Options:
  - word lattice
  - forest of trees
Option 1: lattice structure (Langkilde 2000)

“You may have to eat chicken”: 576 possibilities!
Properties of lattices

• In a lattice, a complete left-right path represents a possible sentence.

• Lots of duplication!
  • e.g. the same word “chicken” occurs multiple times
  • ranker will be scoring the same substring more than once

• In a lattice path, every word is dependent on all other words.
  • can’t model local dependencies
Option 2: Forests (Langkilde ‘00,’02)

S
OR

S.328

PRP.3

you

VP.248

VP.327

OR

NP.318

NP.3.318

to be eaten by

the chicken

VP.248

...
Properties of forests

- Efficient representation:
  - each individual constituent represented only once, with pointers
  - ranker will only compute a partial score for a subtree once
  - several alternatives represented by disjunctive (“OR”) nodes

- Equivalent to a non-recursive context-free grammar
  - S.469 $\rightarrow$ S.328
  - S.469 $\rightarrow$ S.358
  - …
Statistical ranking

• Uses n-gram language models to choose the best realisation $r$:

$$r_{best} = \arg \max_{r \in \text{forest}} \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})$$

$$= \arg \max_{r \in \text{forest}} \prod_{i=1}^{n} P(w_i \mid w_{i-1}) \ [\text{Markov assumption}]$$
Performance of HALogen

Minimally specified input frame (bigram model):

- It would sell its fleet age of Boeing Co. 707s because of maintenance costs increase the company announced earlier.

Minimally specified input frame (trigram model):

- The company earlier announced it would sell its fleet age of Boeing Co. 707s because of the increase maintenance costs.

Almost fully specified input frame:

- Earlier the company announced it would sell its aging fleet of Boeing Co. 707s because of increased maintenance costs.
Observations

- The usual issues with n-gram models apply:
  - bigger $n \rightarrow$ better output, but more data sparseness

- Domain dependent
  - relatively easy to train, assuming corpus in the right format
Evaluation

How should an NLG system/module be evaluated?
Evaluation in NLG

- Types of evaluation:
  - **Intrinsic**: evaluate output in its own right (linguistic quality etc)
  - **Extrinsic**: evaluate output in the context of a task with target users

- Intrinsic evaluation of realisation output often relies on metrics like BLEU and NIST.
BLEU: Modified n-gram precision

- Let $t$ be a translation/generated text
- Let \{r1,\ldots,rn\} be a set of reference translations/texts
- Let $n$ be the maximum ngram value (usually 4)

\[
\text{do for 1 to } n:\n\text{For each ngram in } t:\n\quad \text{max\_ref\_count} := \text{max times it occurs in some } r
\quad \text{clipped\_count} := \text{min(count,max\_ref\_count)}
\quad \text{score} := \text{total clipped counts/total unclipped counts}
\]

- Scores for different ngrams are combined using a geometric mean.
- A brevity penalty is added to the score to avoid favouring very short ngrams.
BLEU example (unigram)

\[ t = \text{the the the the the the the} \]
\[ r_1 = \text{the dog ate the meat pie} \]
\[ r_2 = \text{the dog ate a meat pie} \]

- only one unigram ("the") in \( t \)
- \( \text{max\_ref\_count} = 2 \)
- \( \text{clipped\_count} = \min(\text{count}, \text{max\_ref\_count}) = \min(2,6) = 2 \)
- \( \text{score} = \frac{\text{clipped\_count}}{\text{count}} = \frac{2}{6} \)
NIST: modified version of BLEU

- A version of BLEU developed by the US National Institute of Standards and Technology.

- Instead of just counting matching ngrams, weights counts by their informativeness
  - for any matching ngram between $t$ and reference corpus, the rarer the ngram in the reference corpus the better
Alternative metrics

• Some version of edit (Levenshtein) distance is often used.
  • score reflecting the no. of insertions (I), deletions (D) and substitutions (S) required to transform a string into another string.

• NIST simple string accuracy (SSA): essentially average edit distance
  • SSA = 1-(I+D+S)/(length of sentence)
BLEU/NIST in NLG

• HALogen’s output compared to reference Treebank outputs using BLEU/SSA.

• Fully specified input:
  • output produced for ca. 83% of inputs
  • SSA = 94.5
  • BLEU = 0.92

• Minimally specified input:
  • output produced for ca. 79.3%
  • SSA = 55.3
  • BLEU = 0.51
How adequate are these measures?

- An important question for NLG:
  - Is matching a gold standard corpus all that matters?
  - (As with MT, a complete mismatch is possible, but the output could still be perfectly OK).

- Some recent work suggests that corpus-based metrics give very different results from task-based experiments.
  - Therefore, difficult to identify a relationship between a measure like BLEU and results on system’s “adequacy in a task”.