Lecture 10: Generation and speech synthesis

Pierre Lison, Language Technology Group (LTG) Department of Informatics

Fall 2012, October 12 2012

Outline

• General architecture
• Natural language generation
• Speech synthesis
• Summary
Outline

• General architecture
• Natural language generation
• Speech synthesis
• Summary

A simple schema

Extra-linguistic environment

Language understanding

Recognition hypotheses $\tilde{u}_u$

Speech recognition

Interpreted utterance $\tilde{a}_u$

Dialog management

Intended response $a_m$

Generation

Utterance to synthesise $u_m$

Speech synthesis

User

input speech signal (user utterance)

output speech signal (machine utterance)
A simple schema

List of basic components (4)

- **Natural language generation (NLG)** is the reverse task of NLU: given a high level representation of the response, *find the rights words to express it*

- How to express (or realise) the given intention might depend on various contextual factors

\[ a_m = \text{Confirm} \quad \rightarrow \quad u_m = \begin{array}{c}
2.0 \quad \text{Yes, I agree!} \\
1.3 \quad \text{Yes, I already love this class!} \\
0.8 \quad \text{Sure!}
\end{array} \]
List of basic components (5)

- Finally, speech synthesis (TTS, for «text-to-speech») is the task of generating a speech signal corresponding to the selected system reply.
- Can be modulated in various ways (voice, intonation, accent, etc.)

\[ u_m = \text{Yes, I agree!} \]

Outline

- General architecture
- **Natural language generation**
  - Shallow generation
  - Deep generation
  - Statistical generation
  - Generation of referring expressions
- Speech synthesis
- Summary
The goal of NLG is to convert a high-level communicative goal into a concrete utterance.

As for natural language understanding (NLU), a wide range of methods exists for NLG, with varying degrees of complexity.

- Some of them are «shallow» approaches based on canned utterances.
- Others adopt a «deep» approach based on generic grammatical resources and reasoning patterns.
- And of course, we can also train statistical systems to generate optimal utterances based on data.

Shallow approaches to NLG.

- The system designer manually maps the communicative goals $a_m$ to specific handcrafted utterances $u_m$.
- The utterances might contain slots to be filled.

<table>
<thead>
<tr>
<th>Goal $a_m$</th>
<th>Utterance $u_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AskRepeat</td>
<td>«Sorry, could you please repeat?»</td>
</tr>
<tr>
<td>Assert(cost(ticket, price))</td>
<td>«This ticket will cost you {price} USD»</td>
</tr>
</tbody>
</table>
| Ask(departure)        | «Please state where you are flying from»           
                        | «Where are you departing from?»                    |
Shallow NLG

- Shallow approaches are by far the most popular in commercial systems
  - Limited effort: there are rarely more than a few hundreds prompts for a given system
  - Gives the designer full control over the system behaviour (important for quality assurance)
  - One can introduce some variation by randomly selecting the utterance from a set of possible candidates

Deep NLG

- Shallow approaches rely on the detailed specification of every possible utterance
- A good part of this process is domain-independent and could be automatised

Deep NLG pipeline

Communicative Goal $a_m$ → Sentence planner → Surface realiser → Prosody assigner → Utterance $u_m$
Deep NLG

• Pipeline of modules:

  • **Sentence planning**: selection of *abstract linguistic items* (lexemes, semantic structure) necessary to achieve the communicative goal.

  • **Surface realisation**: construction of a *surface utterance* based on the abstract items and language-specific constraints (word order, morphology, function words, etc.)

  • **Prosody assignment**: determination of the utterance’s *prosodic structure* based on information structure (e.g. what is in focus, what is given vs. what is new)

Sentence planning

• How to perform sentence planning?

  • Recall Grice’s **cooperative principle**, and in particular the Maxim of Quantity: *say exactly as much as is necessary for your contribution*

  • The goal is therefore to find the best way to convey the system’s intention, in the fewest possible words

  • ... but while remaining clear and unambiguous!

  • The communicative goal must sometimes be split in several separate utterances
Surface realisation

• Given a high-level semantics of the utterance provided by the utterance planner, one can then realise it in a concrete utterance

• This is the inverse operation as classical parsing!

• Some grammatical formalisms are «bidirectional» or reversible, i.e. they can be used for both parsing and generation

• HPSG or CCG grammars are reversible (at least can be made reversible, given some grammar engineering)

Deep NLG

• Sentence planning and surface realisation are intertwined operations

• Some systems perform both operations together

• Example: the SPUD and CRISP systems, based on TAG grammars and classical planning algorithms


Prosodic assignment

- Information structure:
  - *theme*: part of an utterance which is talked about (given)
  - *rheme*: what is said about the theme (new)

- Linguistic realisation of this structure in word order, syntax and intonation

Q: I know the AMERICAN amplifier produces MUDDY treble,
Q1: but WHAT does the BRITISH amplifier produce?
A1: (The BRITISH amplifier produces)\(_{th}\) (CLEAN treble)\(_{rh}\)
L+H* LH% H* LL$
Q2: but WHAT produces CLEAN treble?
A2: (The BRITISH amplifier)\(_{rh}\) (produces CLEAN treble)\(_{th}\)
H* LL% L+H* LH$


Statistical generation

- Deep, logic-based approaches to generation can be «brittle»:
  - Requires fine-grained *grammatical resources*
  - Need to *rank* large numbers of alternative utterances produced for a given semantic representation
  - ... and according to which *quality measures*?
  - *User adaptation* is difficult
Statistical generation

• Statistical generation can help us produce more fluent, user-tailored utterances

• Two strategies:
  • Supervised learning: learning generation from annotated examples
  • Reinforcement learning: learning via trial-and-error and feedback

• Possibility to jointly optimise DM and NLG?


Generation of referring expressions

• Generating referring expressions (GRE) is an interesting subproblem of NLG

• Objective: given a reference to an object/entity in the context, find the best referring expression for it!

Let’s say we want to talk about this object

The object?

The triangular object?

The orange triangular object that is to the right of the pink pyramid and to the left of the white cylinder?
Generation of referring expressions

• GRE typically searches for the minimal distinguishing expression for the target

• A distinguishing expression matches the target, but none of the distractors (other salient objects in the context)

Dale and Reiter’s Incremental Algorithm:

1. order the properties $P$ by preference

2. Iterate through ordered list of properties $P$

3. add attribute to description being constructed if it rules out any remaining distractors

4. terminate when a distinguishing description has been constructed (or no more properties)

Incremental algorithm: example

• Assume three properties: Shape, Colour and Size, with Shape > Colour > Size

• We want to talk about object 4

<table>
<thead>
<tr>
<th>Step</th>
<th>Current expression</th>
<th>Remaining distractors</th>
</tr>
</thead>
<tbody>
<tr>
<td>We analyse the Shape property. Object 4 has Shape=triangular</td>
<td>The object</td>
<td>{1,2,3,5,6,7}</td>
</tr>
<tr>
<td>Adding the property Shape=triangular removes distractors {1,2,3,6,7}</td>
<td>The triangular object</td>
<td>{5}</td>
</tr>
<tr>
<td>We analyse the Colour property. Object 4 has Colour=orange</td>
<td>The triangular object</td>
<td>{}</td>
</tr>
<tr>
<td>Adding the property Colour=orange remove the distractor 5</td>
<td>The orange triangular object</td>
<td>∅</td>
</tr>
<tr>
<td>Found distinguishing expression!</td>
<td>The orange triangular object</td>
<td>∅</td>
</tr>
</tbody>
</table>

Outline

• General architecture

• Natural language generation

Speech synthesis:

• Text analysis

• Waveform synthesis

• Summary
Speech synthesis

• The last component of our architecture is the speech synthesiser (or «text-to-speech», TTS)

• The TTS module converts a concrete utterance into a speech waveform

• This mapping is performed in two steps:
  1. Conversion of input utterance into a phonemic representation (text analysis)
  2. Conversion of phonemic representation into the waveform (waveform synthesis)
Text analysis in TTS

• How do we produce the phonemic representation?

1. *Text normalisation* (abbreviations, numbers, etc.)

2. *Phonetic analysis*, based on a pronunciation dictionary and a grapheme-to-phoneme (g2p) converter

3. *Prosodic analysis* to determine e.g. prosodic phrases, pitch accents, and overall tune

Prosodic analysis

• Utterances can be structured in *intonational phrases*

  • Correlated, but not identical to syntactic phrases!

  • These phrases can be extracted based on features such as punctuation, presence of function words etc.

• Words can be more or less prosodically *prominent*

  • E.g. emphatic accents, pitch accents, unaccented, reduced

• Finally, utterances are also characterised by their global *tune* (rise and fall of $F_0$ over time)
Phonemic representation

- At the end of the text analysis (normalisation + phonemic and prosodic analysis), we end up with an internal phonemic representation of our utterance.

```
do       you       really       want       to       see       all       of       it
   110  110  50  50  75  64  57  82  72  41  47  54  130  76  90  44  62  46  220
```

phonemes (ARPA format)

prosodic boundaries

Values for the $F_0$ contour

Waveform synthesis

- Once we have a phonemic representation, we need to convert it into a waveform.

- Two families of methods:
  - Concatenative synthesis: glue together pre-recorded units of speech (taken from a speech corpus)
  - Formant & articulatory synthesis: generate sounds using acoustic models of the vocal tract
## Waveform synthesis

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| **Concatenative synthesis** | • More natural-sounding & intelligible speech  
• Easier modelling, limited signal processing | • Requires a speech corpus  
• Limited flexibility |
| **Formant and articulatory synthesis** | • Explicit model of speech production  
• Many parameters can be tweaked | • Robotic sounds  
• Complex modelling and signal processing |

### Concatenative synthesis

- **Concatenative synthesis:**
  - We record and store various units of speech in a database
  - When synthesising a sound, we search the appropriate segments in this database
  - We then «glue» them together to produce a fluent sound

*Target:* wIntr=del («winter day»)
Concatenative synthesis

- Concatenative methods differ by the kind of «units of speech» they are using
  - **Diphone synthesis**: phone-like units going from the middle of one phone to the middle of the next one
  - **Unit selection**: units of different sizes, can be much larger than a diphone
  - Most commercial TTS systems deployed today are based on unit selection

Diphone synthesis

```
target:  w I n t r= d c I 
      _-w w-I I-n n-t t-r= r--d d-c I c I-
```

Diphone = sound segments from the middle of one phone to the middle of the next phone

[diagram borrowed from M. Schröder]
Diphone synthesis

- For diphone synthesis, the acoustic database consists of recorded diphones
  - Usually embedded in carrier phrases
  - Must be carefully segmented, labelled, pitch-marked, etc.
- After concatenation, the sound must be adjusted to meet the desired prosody
  - Such signal processing might distort the speech sound!
  - Limited account of pronunciation variation (only coarticulation due to neighbouring phone)

Unit selection synthesis

- In unit selection synthesis, the «units of speech» come from a segmented corpus of natural speech

```
target: w I n t r = d c I
```

“Which of these?”
“Let’s discuss the question of interchanges another day.”

acoustic unit database
units = (di-)phone segments recorded in
natural sentences (natural intonation)
Unit selection synthesis

• How do we search for the best units matching our phonemic specifications?
  • Search for a unit that matches as closely as possible our requirements ($F_0$, stress level, etc.) for the unit
  • ... and that concatenates smoothly with its neighbours
• Given a specification $s_t$, we search for the unit $u_t$ that minimises two costs:
  • Target cost $T(u_t, s_t)$: how well the specification matches $u_t$
  • Join cost $J(u_t, u_{t+1})$: how well $u_t$ joins with its neighbour $u_{t+1}$

Assume that we are given an internal phonemic representation $S=\{s_1, s_2, \ldots s_n\}$

We want to find the best sequence of speech units for $S$

In other words, we search for the unit sequence $Û=\{u_1, u_2, \ldots u_n\}$ such that:

$$
ÎU = \arg\min_U \sum_{t=1}^{n} T(s_t, u_t) + \sum_{t=1}^{n-1} J(u_t, u_{t+1})
$$

Target cost between specification $s_t$ and unit $u_t$

Join cost between unit $u_t$ and unit $u_{t+1}$
Unit selection synthesis

• Unit selection can produce high-quality sounds
  • Depending on the corpus size and quality, of course

• But it’s rather inflexible: difficult to modulate the prosody of the speech sound
  • How can we e.g. change the sound’s emotional content?
  • Alternative: annotate the speech corpus with fine-grained informations, and use these in the selection
  • But requires a much larger corpus!

Outline

• General architecture
• Natural language generation
• Speech synthesis
• Summary
Summary

• We started by describing different methods for natural language generation (NLG):

  • Shallow methods rely on canned utterances, possibly augmented with some slots to fill in
  
  • Deep NLG relies on grammatical resources and logical reasoning to plan & realise the utterance
  
  • Finally, statistical methods automatically learn the mapping between communicative goals and their corresponding utterances from data

• We also focused on the problem of generating referring expressions (GRE):

  • Given a reference to an object/entity, try to find the best linguistic expression for it
  
  • To achieve this, we need to find an expression which is both distinguishing (matches the target object, but no other object) and minimal
Summary

• We finally described the speech synthesis task:
  
  • First step: convert the utterance into an internal phonemic representation, together with a prosodic structure
  
  • Second step: convert this representation into a waveform

Incremental NLG + TTS?

• Some recent work on incremental NLG and TTS
  
  • Allows the system to be much more reactive (to correct its own production, and to react to user feedback)
  
  • Can change or rephrase the utterance «on the fly»
  
  • Other advantage: can start playing the sound even before the full synthesis is complete
For our last session, we’ll:

- describe how to evaluate spoken dialogue systems
- and wrap up everything we have seen

If you have any questions or need help (for the 2nd assignment, or on the course in general), we can also talk about it!