NPFL123 Dialogue Systems

10. Text-to-Speech Synthesis

https://ufal.cz/npfl123

Ondřej Dušek, Vojtěch Hudeček, Tomáš Nekvinda
& Jan Cuřín, Petr Fousek

25. 4. 2022
Text-to-speech synthesis

- Last step in DS pipeline
  - from NLG (system utterance text)
  - to the user (audio waveform)
- Needed for all but the simplest DSs
- Sequence-to-sequence conversion
  - from discrete symbols (letters)
  - to continuous time series (audio waves)
    - regression problem
    - mimicking human articulation in some way
- Typically a 2-step pipeline:
  - **text analysis** (frontend) – converting written to phonetic representation
  - **waveform synthesis** (backend) – phonemes to audio
Human articulatory process

• text (concept) → movement of muscles → air movement (sound)

• source excitation signal = air flow from lungs
  • vocal cords resonation
    • base frequency (F0)
    • upper harmonic frequencies
  • turbulent noise

• frequency characteristics moderated by **vocal tract**
  • shape of vocal tract changes (tongue, soft palate, lip, jaw positions)
  • some frequencies resonate
  • some suppressed

(from Heiga Zen's slides)
Sounds of Speech

• **phone/sound** – any distinct speech sound
• **phoneme** – sound that distinguishes meaning
  • changing it for another would change meaning (e.g. *dog* → *fog*)
• **vowel** – sound produced with open vocal tract
  • typically **voiced** (=vocal chords vibrate)
  • quality of vowels depends mainly on vocal tract shape
• **consonant** – sound produced with (partially) closed vocal tract
  • voiced/voiceless (often come in pairs, e.g. *[p] – [b]*)
  • quality also depends on type + position of closing
    • stops/plosives = total closing + “explosive” release ([p], [d], [k])
    • nasals = stops with open nasal cavity ([n], [m])
    • fricatives = partial closing (induces friction – hiss: [f], [s], [z] …)
    • approximants = movement towards partial closing & back, half-vowels ([w], [j] …)
Sounds of Speech

- Word examples according to Received Pronunciation ("Queen’s English"), may vary across dialects
- More vowels: diphthongs (changing jaw/tongue position, e.g. [ei] wait, [əʊ] show)

http://www.ipachart.com/  
(clickable with sounds!)
Spectrum

- speech = compound wave
  - different frequencies (spectrum)
  - shows in a spectrogram
    - frequency – time – loudness
- base vocal cord frequency $F_0$
  - present in voiced/vocals
  - absent in voiceless
- formants = loud upper harmonics
  - of base vocal cord frequency
  - $F_1$, $F_2$ – 1$^{st}$, 2$^{nd}$ formant
    - distinctive for vowels
- noise – broad spectrum
  - consonants (typical for fricatives)

formant frequency in basic vowels

plosive = stop (silence) + explosion (short noise)

fricative = noise

diphthong – moving formants

voiced

voiceless

https://en.wikipedia.org/wiki/Spectrogram
http://www.speech.kth.se/courses/GSLT_SS/ove.html
From sounds to utterances

• phones group into:
  • **syllables** – minimal pronounceable units
  • **stress units** (~ words) – group of syllables with 1 stressed
  • **prosodic/intonation units** (~ phrases)
    • independent prosody (single prosodic/pitch contour)
    • tend to be separated by pauses
  • utterances (~ sentences, but can be longer)

• neighbouring phones influence each other a lot!
• **stress** – changes in timing/F0 pitch/intensity (loudness)
• **prosody/melody** – F0 pitch
  • sentence meaning: question/statement
  • tonal languages: syllable melody distinguishes meaning
TTS Prehistory

• 1st mechanical speech production system
  • Wolfgang von Kempelen’s speaking machine (1790’s)
  • model of vocal tract, manually operated
  • (partially) capable of monotonous speech

• 1st electric system – Voder
  • Bell labs 1930, operated by keyboard (very hard!)
  • pitch control

• 1st computer TTS systems – since 1960’s

• Production systems – since 1980’s (→)

(Lemmetty, 1999)
https://en.wikipedia.org/wiki/Voder
https://youtu.be/TsdOej_nC
TTS pipeline

- frontend & backend, frontend composed of more sub-steps
  - frontend typically language dependent, but independent of backend

(from Heiga Zen’s slides)
Segmentation & normalization

• remove anything not to be synthesized
  • e.g. HTML markup, escape sequences, irregular characters

• segment sentences

• segment words (Chinese, Japanese, Korean scripts)

• spell out:
  • abbreviations (context sensitive!)
  • dates, times
  • numbers (ordinal vs. cardinal, postal codes, phone numbers…)
  • symbols (currency, math…)

• all typically rule-based

10 NPFL123 L10 2022
Grapheme-to-Phoneme Conversion

• main approaches: pronouncing dictionaries + rules
  • rules good for languages with regular orthography (Czech, German, Dutch)
  • dictionaries good for irregular/historical orthography (English, French)
  • typically it’s a combination anyway
    • rules = fallback for out-of-vocabulary items
    • dictionary overrides for rules (e.g. foreign words)
  • can be a pain in a domain with a lot of foreign names
    • you might need to build your own dictionary (even with a 3rd-party TTS)
• phonemes typically coded using ASCII (SAMPA, ARPABET…)
• pronunciation is sometimes context dependent
  • part-of-speech tagging
  • contextual rules

\[
\begin{align*}
  \text{record (NN)} &= ['\mathcal{E}ko:d] \\
  \text{read (VB)} &= ['\mathcal{E}d] \\
  \text{record (VB)} &= ['\mathcal{I}ko:d] \\
  \text{read (VBD)} &= ['\mathcal{E}d] \\
  \text{the oak} &= [\mathcal{D}i\mathcal{E}k\mathcal{O}k] \\
  \text{the one} &= [\mathcal{D}o\mathcal{E}\mathcal{E}n]
\end{align*}
\]
Intonation/stress generation

• rules/statistical
  • predicting intensity, F0 pitch, speed, pauses
    • stress units, prosody units
  • language dependent
    • traditionally: classification – bins/F0 change rules

• based on:
  • punctuation (e.g. “?”)
  • chunking (splitting into intonation units)
  • words (stressed syllables)
  • part-of-speech tags (some parts-of-speech more likely to be stressed)
  • syntactic parsing
SSML (Speech Synthesis Markup Language)

• manually controlling pronunciation/prosody for a TTS
  • must be supported by a particular TTS
    • e.g. Alexa supports this (a lot of other vendors, too)

• XML-based markup:
  • <break>
  • <emphasis level="strong"/>
  • <lang>
  • <phoneme alphabet="ipa" ph="ˈbɑ.təl"> (date, fraction, address, interjection…)
  • <prosody rate="slow">, <prosody pitch="+15.4%">, <prosody volume="x-loud"/>
  • <say-as interpret-as="digits"> (date, fraction, address, interjection…)
  • <sub alias="substitute">subst</sub> (abbreviations)
  • <voice>
  • <w role="amazon:VBD">read</w> (force part-of-speech)

Waveform Synthesis

- many different methods possible
- **formant-based** (~1960-1980’s)
  - rule-based production of formants & other components of the wave
- **concatenative** (~1960’s-now)
  - copy & paste on human recordings
- **parametric** – model-based (2000’s-now)
  - similar to formant-based, but learned from recordings
  - HMMs – dominant approach in the 2000’s
  - NNs – can replace HMMs, more flexible
- **NN-based end-to-end methods**
  - now state-of-the-art
Formant-based Synthesis

• early systems
• based on careful handcrafted analysis of recordings
  • “manual” system training
  • very long evolution – DECTalk took ~20 years to production
  • barely intelligible at first
• rules for composing the output sound waves
  • based on formants resonators + additional components
  • rules for sound combinations (e.g. “b before back rounded vowels”)
  • rules for suprasegmentals – pitch, loudness etc.
• results not very natural, but very intelligible in the end
• very low hardware footprint

http://www.festvox.org/history/klatt.html (examples 17 & 35)
Concatenative Synthesis

• Cut & paste on recordings
  • can’t use words or syllables – there are too many (100k’s / 10k)
  • can’t use phonemes (only ~50!) – too much variation
    • coarticulation – each sound is heavily influenced by its neighbourhood
• using diphones = 2\textsuperscript{nd} half of one phoneme & 1\textsuperscript{st} half of another
  • about 1,500 diphones in English – manageable
  • this eliminates the heaviest coarticulation problems (but not all)
  • still artefacts at diphone boundaries
• smoothing/overlay & F0 adjustments
  • over-smoothing makes the sound robotic
  • pitch adjustments limited – don’t sound natural
• needs lots of recordings of a single person
• diphone representations: formants, LPC, waveform

http://www.festvox.org/history/klatt.html (examples 18 & 22)
https://www.ims.uni-stuttgart.de/institut/mitarbeiter/moehler/synthspeech/ (Festival English diphone example, MBROLA British English example)

Dixon & Maxey (1968)
formant diphones

Olive (1977)
LPC diphones

Festival (1997)
diphone synthesis
http://www.cstr.ed.ac.uk/projects/festival/

MBROLA (1996)
http://tcts.fpms.ac.be/synthesis/
Unit-selection Concatenative Synthesis

• using more instances of each diphone
  • minimize the smoothing & adjustments needed

• selecting units that best match the target position
  • match target pitch, loudness etc. (specification $s_t$) – target cost $T(u_t, s_t)$
  • match neighbouring units – join cost $J(u_t, u_{t+1})$
  • looking for best sequence $\tilde{U} = \{u_1, ..., u_n\}$, so that:

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

• solution: Viterbi search
  • leads to joins of stuff that was recorded together

• a lot of production systems use this
  • still state-of-the-art for some languages
  • but it’s not very flexible, requires a lot of single-person data to sound good

http://www.cs.cmu.edu/~awb/festival_demos/general.html
http://mary.dfki.de/
https://text-to-speech-demo.ng.bluemix.net/
https://deepmind.com/blog/wavenet-generative-model-raw-audio/
Model-based Parametric Synthesis

- trying to be more flexible, less resource-hungry than unit selection
- similar approach to formant-based – modelling
  - but this time learned statistically from a corpus
- inverse of model-based ASR (next lecture)
- ideal: model $p(x|w, X, W)$
  - auxiliary representations – features
  - approximate by step-by-step maximization:
    - extract features from corpus (acoustic, linguistic)
    - learn model based on features
    - predict features given text (linguistic, then acoustic)
    - synthesize given features

[Diagram showing the process of model-based parametric synthesis]

training waveforms $X$

training transcriptions $W$

text analysis

vocoder analysis

acoustic features

linguistic features

model training

acoustic model

predicted linguistic features

feature prediction

predicted acoustic features

vocoder synthesis

waveform $x$

synthesis

text $w$

text analysis

predicted linguistic features

feature prediction

predicted acoustic features

vocoder synthesis

waveform $x$
Features for model-based synthesis

- Acoustics: piecewise stationary source-filter model
  - spectrum (filter/resonance frequencies): typically MFCCs, $\Delta$, $\Delta\Delta$
  - excitation (sound source): voiced/unvoiced, log F0, $\Delta$, $\Delta\Delta$

- Linguistics:
  - phonemes
  - stress
  - pitch

(from Heiga Zen's slides)

(Tokuda et al., 2013)

(from Pierre Lison's slides)
HMM-based Synthesis

• Using HMMs as the speech model
• Context-dependent phoneme-level HMMs
  • concatenated into a big utterance-level HMM
  • transition & emission probabilities
    – multivariate Gaussian distributions
  • loops – handling different phoneme lengths
• Too many possible contexts → use decision-tree-based clustering
  • ~10M possible context combinations
  • regression trees (outputs = real-valued Gaussian parameters)
• Generating from this would result in step-wise sequence
  • sample from each Gaussian, wait a few ms, sample…
  • → this is where $\Delta, \Delta\Delta$ are used
HMM-based Synthesis

• Pros vs. concatenative:
  • small data footprint
  • robust to data sparsity
  • flexible – can change voice characteristics easily

• Con:
  • lowered segmental naturalness

FLite/HTS
(various settings)
http://flite-hts-engine.sp.nitech.ac.jp/index.php

MARY TTS
HSMM-based
http://mary.dfki.de/

(Tokuda et al., 2013)
NN-based synthesis

- Replacing clunky HMMs and decision trees with NNs
  - Basic – feed forward networks
    - predict conditional expectation of acoustic features given linguistic features at current frame
    - trained based on mean squared error
  - Improvement – RNNs
    - same, but conditioned on current & previous frames
    - predicts smoother outputs (given temporal dependencies)
- NNs allow better features (e.g. raw spectrum)
  - more data-efficient than HMMs
- This is current production quality TTS

(source: Heiga Zen’s slides)

- Google LSTM parametric
- IBM Watson DNN
  - [https://text-to-speech-demo.ng.bluemix.net/](https://text-to-speech-demo.ng.bluemix.net/)
WaveNet

- Removing acoustic features – direct waveform generation
  - no need for spectrum
- Based on convolutional NNs
  - 16k steps/sec → need very long dependencies
  - dilated convolution – skipping steps
  - exponential receptive field w.r.t. # of layers
  - conditioned on linguistic features
  - predicting quantized waves using softmax
- Not tied to ±stationary frames
  - can generate highly non-linear waves
- Very natural, Google’s top offering now

(van den Oord et al., 2016)
https://arxiv.org/abs/1609.03499
https://deepmind.com/blog/wavenet-generative-model-raw-audio/

(from Heiga Zen’s slides)
• Different approach: removing linguistic features
  • trained directly from pairs of waveforms & transcriptions
  • generates linear scale spectrograms (at frame level)
  • Griffin-Lim conversion: spectrogram \rightarrow \text{waveform}
    • estimate the missing wave phase information
• Based on seq2seq models with attention
  • encoder – CBHG (1D convolution + highway net + GRU)
  • decoder – seq2seq predicts mel-scale spectrograms, $r$ steps at a time
    • neighbouring frames in speech are correlated
  • postprocessing – to linear scale
    • access to whole decoded sequence
• Very natural outputs
Extensions: Faster, Multilingual

• Faster: Convolutions instead of RNNs
  • predicting mel spectrograms (requires an additional vocoder, Griffin-Lim is too weak for that)
  • encode phonemes
  • predict duration ($k$ frames)
  • copy encodings $k$ times & decode

• Multilingual: Meta-learning
  • predict network parameters for each language with a smaller network
  • added speaker ID – multi-speaker
  • can learn consistent voice with multiple languages

[https://github.com/janvainer/speedyspeech](https://github.com/janvainer/speedyspeech)

[https://github.com/Tomiinek/Multilingual_Text_to_Speech](https://github.com/Tomiinek/Multilingual_Text_to_Speech)
Summary

• Speech production
  • “source-filter”: air + vocal cords vibration + resonation in vocal tract
  • sounds/phones, phonemes
  • consonants & vocals
  • spectrum, formants
  • pitch, stress

• Text-to-speech system architectures
  • rule/formant-based
  • concatenative – diphone, unit selection
  • model-based parametric: HMM, NNs
  • end-to-end neural: WaveNet, Tacotron
Contact us:

https://ufaldsg.slack.com/
{odusek,hudecek}@ufal.mff.cuni.cz
Skype/Meet/Zoom (by agreement)

Get these slides here:

http://ufal.cz/npfl123

References/Inspiration/Further:

• Pierre Lison’s slides (Oslo University): https://www.uio.no/studier/emner/matnat/ifi/INF5820/h14/timeplan/index.html
• Heiga Zen’s lecture (ASRU 2015): https://ai.google/research/pubs/pub44630