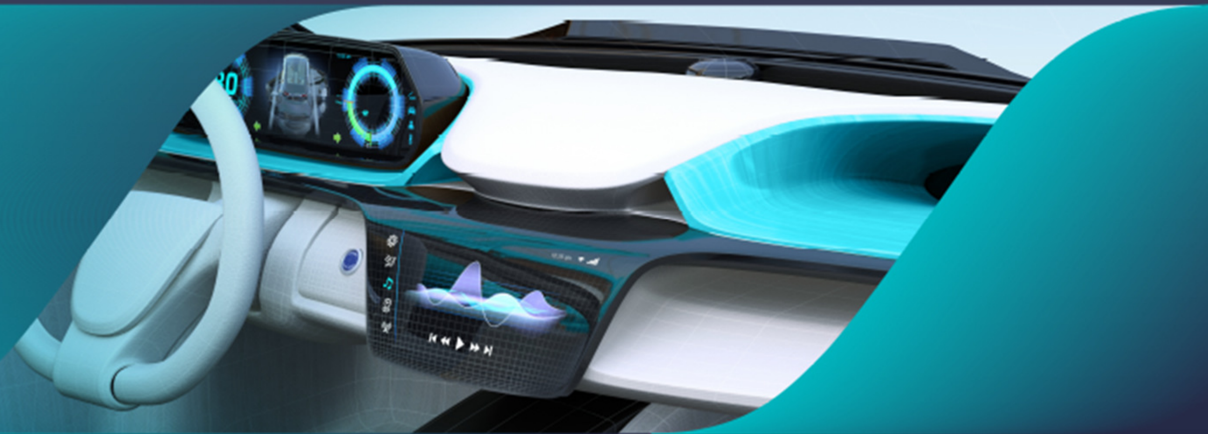
Introduction to Speech Recognition & Linguistics



Séminaire du Cental
Louvain-la-Neuve, 19 December 2019
Peter Dirix (Cerence, KU Leuven)
Peter.Dirix@cerence.com

Overview

- Introduction to speech recognition
- What does a Linguistics team do in ASR?

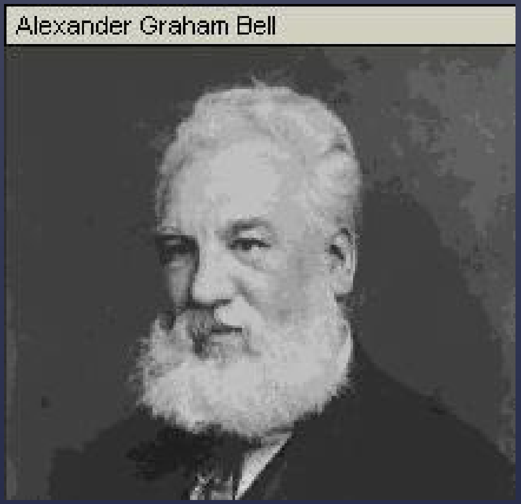
General introduction to ASR

Overview

- Very short history of speech recognition
- Stochastic approach: acoustic & language models, lexicon
- Pre- and postprocessing
- Testing
- Adaptation
- Issues, deep learning & future

Speech recognition - history

- 1876: Alexander Graham Bell invents the telephone – he was actually looking for a system for his deaf wife that could render speech into text
- 1939: First speech synthesizer (Bell Laboratories)
- 1952: First speech recognition system (Bell Laboratories)
- 1971: DARPA starts financing of speech recognition research
- 1976: IBM – *Hidden Markov Models*
- 1980s: First applications (IBM, Dragon Systems, Kurzweil)
- 1990: First dictation tool (Dragon – 5K words)
- 1997: IBM ViaVoice (first PC dictation tool)
- 1997-2006: Consolidation around Lernout & Hauspie/Scansoft/Nuance
- 2010s: Siri, SVoice (phone), television, cars, …



Types of speech recognition

- ASR = automatic speech recognition
- Discontinous vs. Continous speech recognition
  - Command & control (‘Lights on’, number choices in automated call centers)
  - Discontinuous speech recognition (pauses between words - obsolete)
  - Continuous speech recognition (natural speech)
- Vocabulary size
  - Present day: large-vocabulary continuous speech recognition (LVCSR)
- Speaker dependency
  - Speaker-dependent systems: require training for acoustic adaptation (used for dictation)
  - Speaker-independent systems: most non-dictation systems

Speech recognizer

- Traditionally: Stochastic (statistical) approach
- Combination of elements:
  - Acoustic model (AM) – modeling the digitized signal to phoneme mapping
  - Pronunciation lexicon – modeling the phoneme-to-word mapping
  - Language model (LM) – modeling the word context
  - Search engine/decoder: software linking the components

Pronunciation lexicon (1)

- The lexicon contains the tokens available for training and recognition
  - Phonetic transcriptions: pronunciations (“prons”)
  - Morphological and syntactic info: part of speech, inflection class
  - Semantic info: first-name, tv-show
- Only words in the lexicon can be recognized!!!
  - But words might be added automatically and ‘pron-guessed’

Pronunciation lexicon (2)

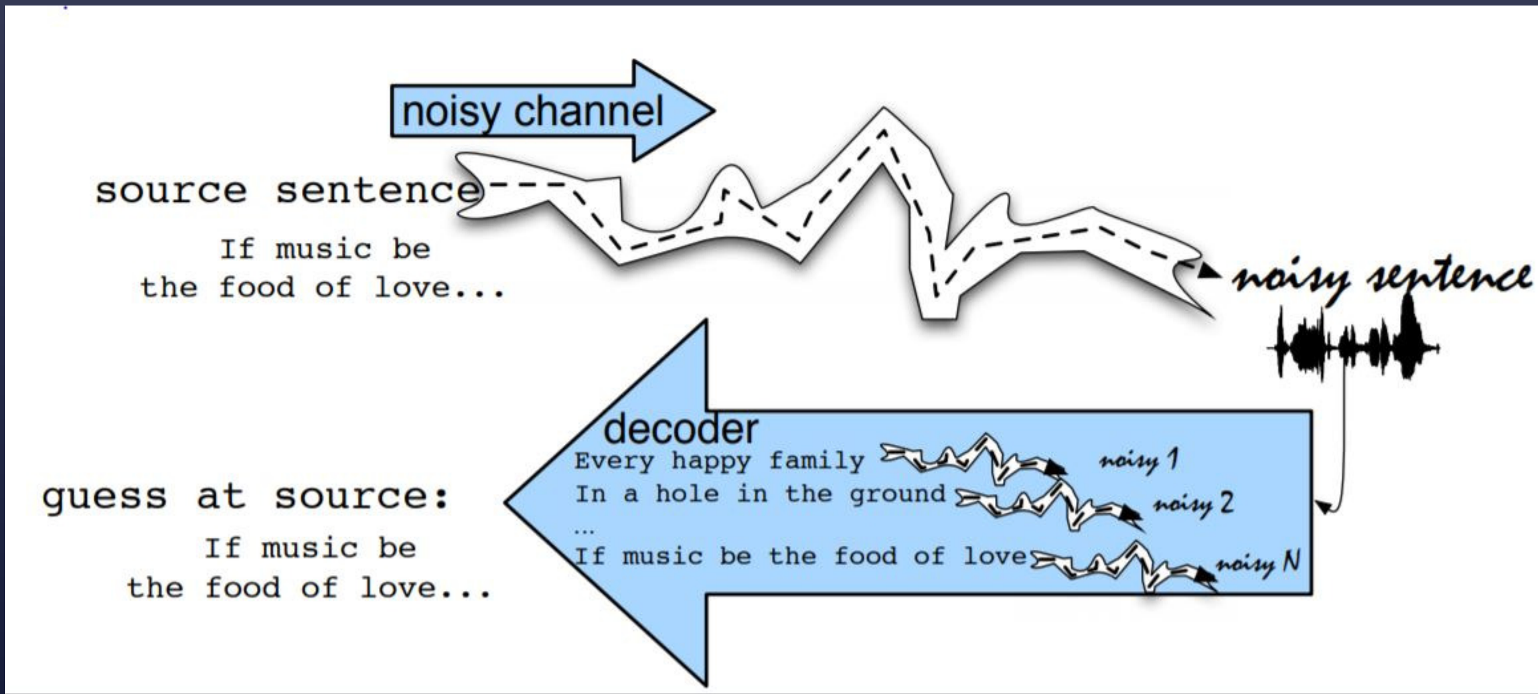
- Tokens are not equivalent to (paper) dictionary entries – token philosophy
  - All word forms, not just lemmas
  - Single words: bottle version Clinton
  - Multiwords: New_York ad_hoc
  - Acronyms and alphanumerics: NATO R2-D2
  - Letters: A c
  - Symbols: . , $
  - No numbers in digits

Phoneme set

- Words are split in phonemes (phones)
  - 25-150 phonemes per language
  - Native phonemes (+ frequent foreign phonemes) – threshold needed for training
  - Tonal languages: vowel + tone = a single phoneme
  - Pause fillers added as phonemes
  - Phonemes cover for allophonic variation

Stochastic approach – Noisy channel model

- Search through space of all possible sentences
- Pick the one that is most probable given the waveform



Stochastic approach – Noisy channel model

- What is the most likely sentence out of all sentences in a language L given some acoustic input O (observation)?
- Acoustic input O = sequence of individual observations
  - $O = o_1, o_2, \ldots, o_t$
- Utterance W = sequence of words
  - $W = w_1, w_2, \ldots, w_n \quad \in \; L$

Stochastic approach – Noisy channel model

- Pick highest probability

$$\hat{W} = \underset{W \in L}{\arg\max}\, P(W \mid O)$$

- Apply Bayes' theorem

$$\hat{W} = \underset{W \in L}{\arg\max}\, \frac{P(O \mid W)P(W)}{P(O)}$$

- Denominator is the same for all candidate-transcriptions:

$$\hat{W} = \underset{W \in L}{\arg\max}\, P(O \mid W)P(W)$$

Stochastic approach – Noisy channel model

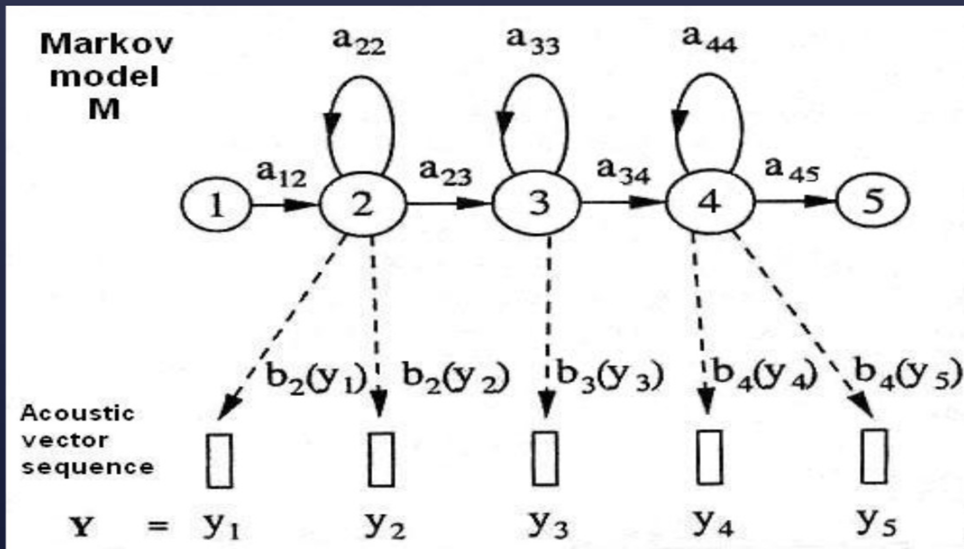
- $$\hat{W} = \underset{W \in L}{\arg\max}\, P(O \mid W)P(W)$$
  - **P (O | W)** represents the link between AM and lexicon (likelihood)
  - **P (W)** represents the link between lexicon and LM (prior)

Acoustic model (1)

- Polyphones
  - Grouped in n-phones, e.g. diphones (2-phones) and triphones (3-phones)
  - Triphones are modeled by states – using *Hidden Markov Model* (HMM):
    - Vowels, consonants: 2-3 states
    - Diphthongs, triphtongs: 3 states
    - Pause fillers: 5-6 states

Acoustic model (2): HMMs

- State transitions are probabilistic (a_{ij})
- Each state produces an acoustic vector y_t with probability $b_j(y_t)$
- Input state does not uniquely define next state
- $P(Y) = a_{12}\, b_2(y_1)\, a_{22}\, b_2(y_2)\, a_{23}\, b_3(y_3)\ldots$



Acoustic model training

- Select a phoneme set for a particular language
- Create a pronunciation lexicon using this phoneme set
- Define number of states for each phoneme
- Audio data collection (100s of hours)
  - Training + test data
  - Balanced on gender, age, accent/region, …
  - Using same style and vocabulary as target product
  - Create *truth transcriptions*
  - For cars, you might need in-car data or noisified data
- Train HMM using audio, transcriptions and pronunciation lexicon
  - Calculate for each triphone:
    - Transition probabilities between the states a_{ij}
    - Production probabilities $b_j(y_t)$ of the acoustic vector

Acoustic model (3)

Input signal

"dictaphone"

Spectral analysis

Set of acoustic vectors:

$O = (o_1, o_2, o_3, .., o_k)$

<u>Lexicon</u>

dictaphone $(o_1,o_2,o_3,.., o_k)$, $(o_{11},o_{17},o_{23},.., o_m)$,...

dictation $(o_1,o_2,o_3,.., o_k)$, $(o_1,o_{17},o_{23},.., o_t)$, ...

...

telephone $(o_5,o_2,o_{13},.., o_k)$, $(o_1,o_{71},o_3,.., o_n)$,...

P(O|W)
AM score

P(O|"dictaphone") = S_1
P(O|"dictation") = S_2
P(O|"telephone") = S_3

P(O|W) – acoustic model (probability of a given vector sequence O, given a word W)

Language models (1)

- In $\hat{W} = \arg\max_{W \in L} P(O \mid W) P(W)$, **P(W)** is the language model
- LM represents context, i.e. syntax and semantics
  - Only models local dependencies
  - Models word sequences, called n-grams , which each have a probability
  - Topic-dependent
- AM generates hypotheses, LM scores the hypotheses
  - $P(W = w_1, w_2, \ldots, w_n) = P(w_n \mid w_1, w_2, \ldots, w_{n-1})$

Language models (2)

- ASR uses 2- to 4-grams
  - Unigrams: $P(W) = P(w_n)$
  - Bigrams (2-grams): $P(W) = P(w_n \mid w_{n-1})$
  - Trigrams (3-grams): $P(W) = P(w_n \mid w_{n-2}, w_{n-1})$
  - 4-grams: $P(W) = P(w_n \mid w_{n-3}, w_{n-2}, w_{n-1})$
- Start with 4-grams, back off to lower *n*
- E.g. “I need to buy milk for my (cereal|serial|stereo)”
  - “milk for my cereal” is a likely 4-word sequence
  - “milk for my serial” and “milk for my stereo” are not

Language models (3)

- Class models
  - Create word classes based on PoS, morphology, semantics
  - Create n-gram models for word classes
  - Combine with word n-gram model
  - Classes can be collapsed automatically or rule-based
  - Classes can also be derived automatically by clustering
- N-gram model could be replaced by a context-free grammar
  - Only terminals or with ‘slots’
  - Only utterances in grammar can be recognized

Language models (4): model size

- Unigrams
  - Depends on the language
  - General dictation models: usually about 150K words, medical topics 50K
  - Mobile models: insert all needed lexical entries (can be millions)
- Higher-order n-grams
  - Usually 10s of millions
  - Still data sparsity issues: back-off to lower-order n-grams/class model

Language model training (1)

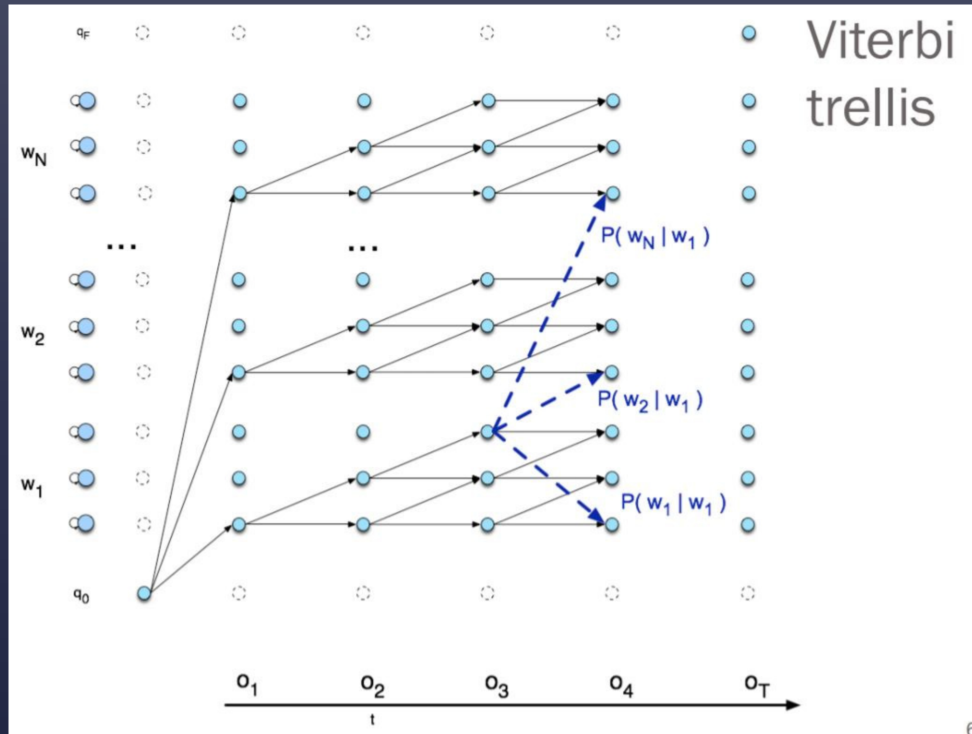
- LMs are trained from large text corpora
  - Usually billions of words: “There’s no better data than more data” (but also CICO – crap in, crap out)
  - The raw text is fed into the tokenizer for normalization first
  - Tokenized text is used to create n-grams
- Only words appearing in the data will end up in the product’s lexicon
  - We do use a background lexicon in dictation products
  - Can force in words too
- Unknown words (out-of-vocabulary words or OOVs)
  - Vetted, and if necessary added to the lexicon and pronned
  - Normalization + common misspellings: respell rules – apply to corpus
    - E.g.: contatc -> contact; disk -> disc (UK English)

Language model training (2)

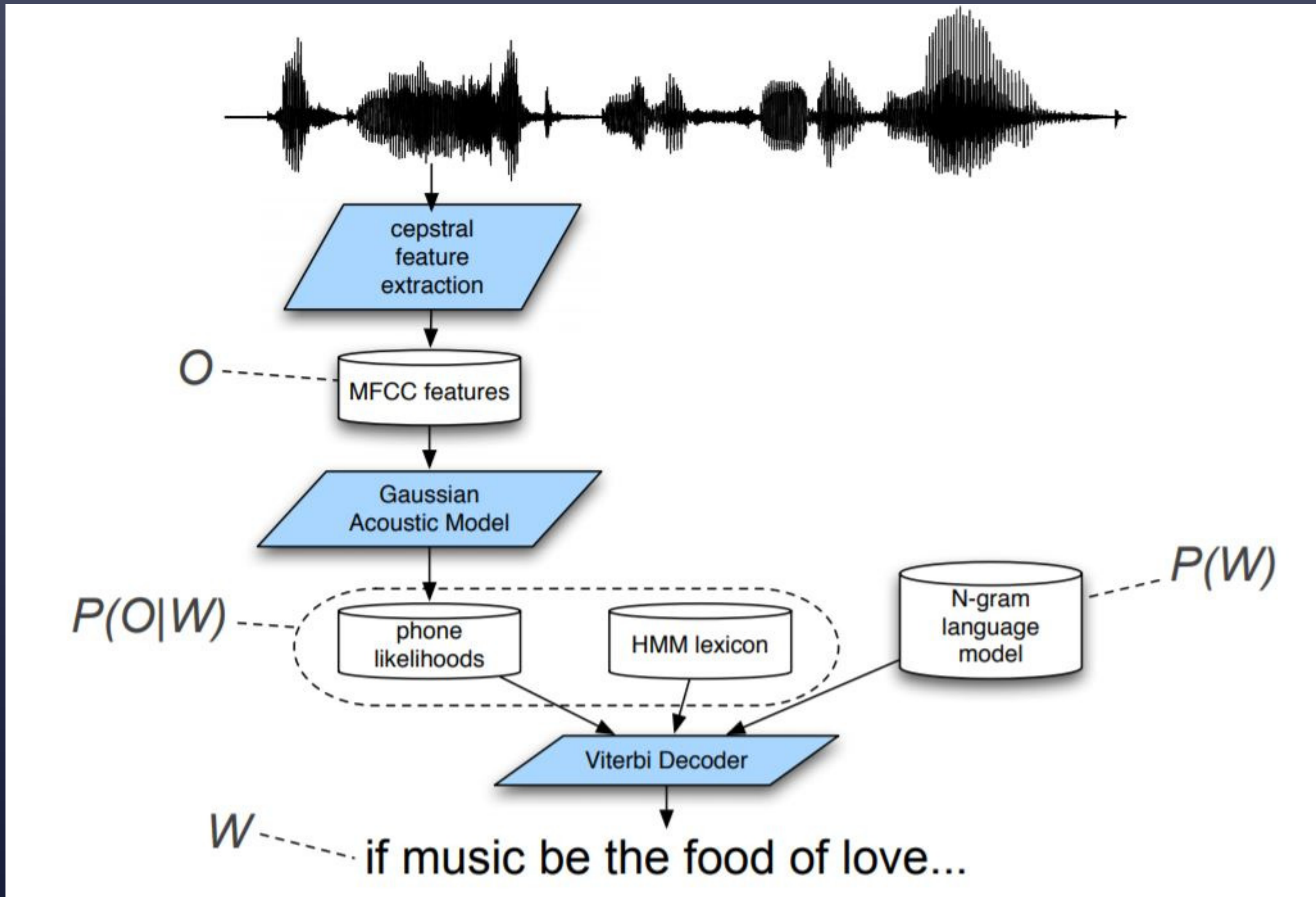
- Different products use different data sources
  - Medical dictation: medical reports, patient records, user data
  - General dictation: purchased general data (like newspaper articles), scraped data (blogs, news, …), user data
  - Automotive: generated data (carrier phrases with slots like street addresses, music artists and titles), user data
- LMs have different weight-set configurations
  - Navigation: weight street addresses more
  - Play a song: weight music data more

Search engine/decoder

- Search engine applies *Viterbi algorithm* to find the best path through the states
- *Viterbi algorithm* = dynamic programming combining AM, LM, and lexicon to get transcribed utterance
- Computing the joint probability of the observation sequence and together with the best state sequence



Architecture



Pron(unciation) guessing

- Also called G2P (grapheme to phoneme)
- Guess pronunciation for words in training data but not in lexicon
- Might request one or several pronunciations
- Several approaches:
  - Rule-based
  - N-gram based
  - Neural networks

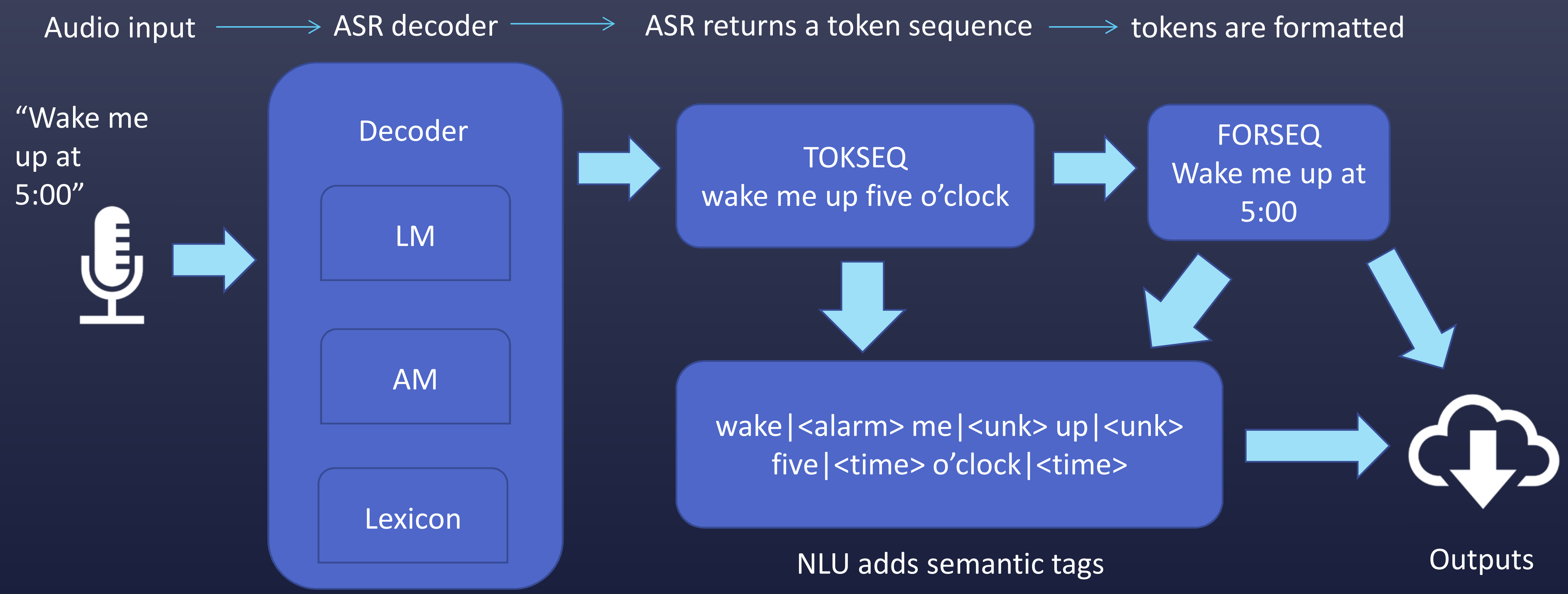
Preprocessing

- Data need to be clean, flat text
- Normalize on spelling also fixing most common spelling mistakes
- Deal with numbers in digits, dates, times, etc.
- Might consist of decompounding, clitic detachment etc., depending on the language

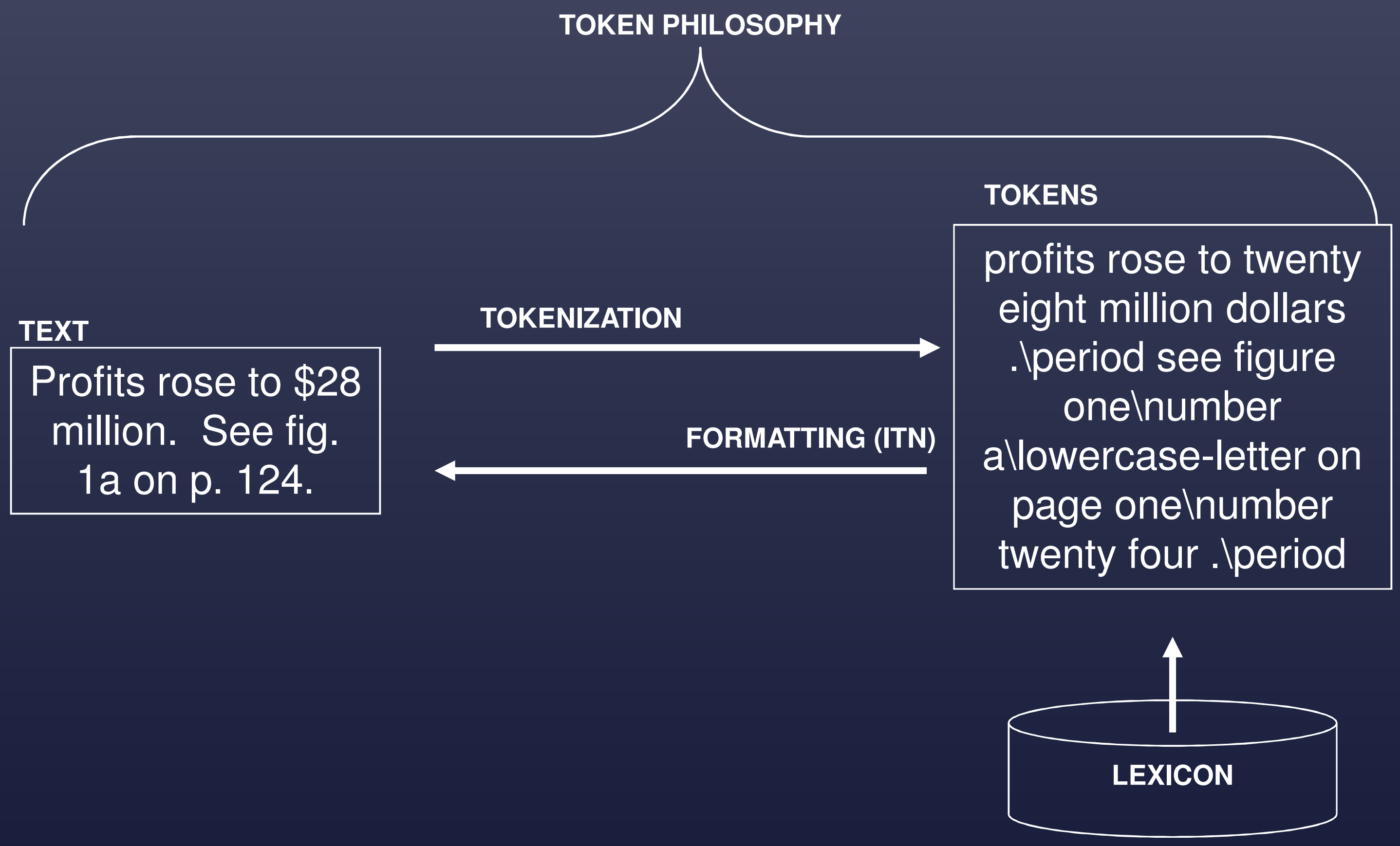
Postprocessing

- Output needs to be consistent
- Output needs to be in human-readable format, so numbers in digits, dates, times etc.
- Depending on the language, compounding or clitic reattachment might be needed
- Profanity filtering?

Basic recognition pipeline



Tokenization versus formatting



ASR testing

- Set aside part of audio data collection as test set
- Apply ASR process and compare with *truth transcriptions*
  - Usually at token level, sometimes as formatted text level
  - Adjudication maps exist to cover for equivalences
- Quality is given as:
  - WER (word error rate)
  - SER (sentence error rate)
  - CER (character error rate) – Chinese and Japanese
- Impact from improvements on WER
  - AM algorithm improvements > LM algorithm improvements > lexical improvements

Speaker and LM adaptation

- Training AM by reading prompted texts – dictation (enrollment)
- Adding speaker-defined lexicon (+ pronunciations) – dictation, mobile products
- Adding speaker-defined text corpora – dictation, mobile products

Data: Adaptation of generic models

- Audio adapted to a particular situation (e.g. in-car data) – might be simulated
- In-domain data to adapt a general system
- Lexicon should cover domain, including all necessary named entities
- Might include foreign-language words, in particular named entities (POIs, song/movie titles, ...) – also needing transcriptions in the target-language phoneme set
- Nuance used e.g. *value-adding resellers* for particular domains

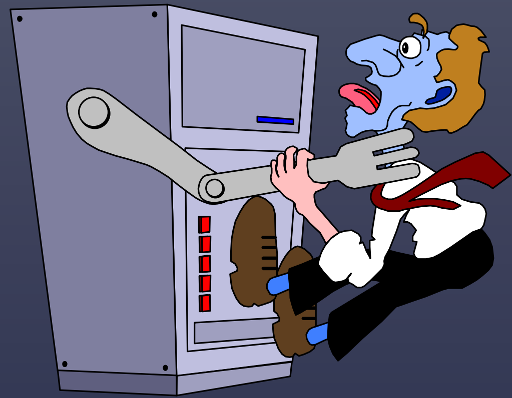
Issues with speech recognition (1)

- Variation between speakers
  - Physiological factors: form and length of the vocal tract (depending on gender, age, …)
  - Sociolinguistic factors: accent, dialect, level of education, age, …
- Variation within one speaker
  - Physiological factors: having a cold, emotion, …
- Variation within the environment
  - Distance to microphone
  - Background noise
- Phonological factors
  - Coarticulation, prosody, phrasal stress, position in the sentence

Issues with speech recognition (2)

- Training data might not be available
  - Would need to be collected and transcribed
  - Even if they exist, there might be legal/privacy issues (e.g. medical records)
  - Data ageing and relevancy
- Limited context
  - E.g. Google search – no context to help disambiguation
- Multilinguality
  - Speakers might have to pronounce foreign words/utterances, like music titles, business names, addresses
  - Wide variety in proficiency in other languages
  - Also depends on context (carrier phrase or not?)
  - Might want to dictate messages in different languages (combination of ASR systems)

Issues with speech recognition (3)



- User expectations and behavior
  - Is the result displayed or does it lead to an action?
  - Dictated punctuation vs. auto-punctuation
  - Short words vs. longer words; ambiguity (to-two-too)
  - ‘Good’ and ‘bad’ speakers: stammering, hesitating, recapitulating, speed, loudness variations, …
- Humans also have issues recognizing speech
  - Think of spelling alphabets (Alpha, Bravo, Charlie)
  - Humans often rely on other signals, e.g. visual input or the context, to "hear" the correct sound: the McGurk effect.

Deep learning

- Neural nets have been introduced in commercial ASR since 2014
- On all levels:
  - LM: using about 20K unigrams and combined with n-gram models
  - AM: replace production probabilities by recurrent neural networks (RNNs)
  - Search engine: encode sequence of acoustic vectors using long short-term memories (LSTMs) and decode by another set of RNNs
- Requires far more training data, training time and GPU usage than stochastic models
- Yields considerably better WERs, but errors are more difficult to explain (and to be hacked away)
- Research: end-to-end models

Other remarks

- Products are moving from device (PC, phone, TV, computer) to the cloud
- Non-dictation speech recognition is usually integrated with NLU (natural language understanding) in order to perform the desired action
  E.g.: Navigate to UCL Saint-Luc → will look up GPS coordinates and plot a route
- Medical and other professional dictation software might be integrated with other applications, as imaging tools, content management systems, invoicing systems that manage a whole workflow
- Most of the training resources are also useful for speech synthesis (TTS)

Future

- Integrated dialog systems with speech recognition, natural language understanding, dialog manager, natural language generation and speech synthesis
- Speech-to-speech translation, either direct or through speech recognition, machine translation and speech synthesis
- Domotics: more applications

What does a linguistics team do in ASR?

Pron lexicon and morphology

- Analyse a new language and define set-up
  - E.g. French support for liaison, Italian/Spanish attached clitics, Germanic compounds
- Defining phoneme sets, pronning and transcription guidelines
- Create initial lexicon
- Actual pronning is usually done by linguistic consultants
- Move to morphologically generated lexica to increase coverage and consistency
  - Create morphological paradigms (use external resources if necessary) – for both spelling and pronunciation
  - Tag individual tokens with relevant paradigm(s) – done by consultants
  - Remove non-lemmas

Foreign words and pron guessing

- How do you pronounce a foreign word or phrase?
- Might be ‘nativized/naïve’, ‘foreign/informed’ or more frequently somewhere in between
- Naïve and informed prons are ‘easy’ to pron guess
- In-between prons need to be created manually, or at least up to a level that we can train a pron guesser
- Evaluate and compare set-ups

Tokenization and formatting grammars

- Create/maintain context-free grammars and rules for tokenization and formatting
  - Numerals, dates, times
  - Measurements and prices
  - Street addresses, phone numbers
  - Abbreviations
- Add new domains
  - Medical: vertebrae, blood pressure, ...
  - Navigation: road numbers
  - TV: channels, episodes etc.

Recognition and training data generation grammars

- In the past, in particular for on-device products, the ASR LM was actually a grammar, possibly with non-terminals
  - (I want to) listen to <album-name|song-name|artist-name>
    play <album-name|song-name>
  - Navigate to <street-address|poi>
    Please take me to <street-address|poi>
- Now we train LMs on generated data from similar grammars
- Slot fillers come from large databases
- NLU models are trained on annotated version of the same data

Analyse bugs/field data and nightly update

- We only rebuild LMs every few months/years (depending on priority)
- However, we want to push out updates
- Nightly update:
  - Data patches: add more training data
  - Lexicon patches: add more tokens with pron
  - Rewrites: respell output of recognizer, e.g.
    - Google heurte -> Google_Earth
    - des destination -> des destinations
    - a\verb Cannes -> à\preposition Cannes
  - Can also patch tokenizer and formatter grammars

Merci!