ESSLLI

Incremental Speech and Language Processing for Interactive Systems

Timo Baumann, Arne Köhn,
Universität Hamburg, Informatics Department
Natural Language Systems Division
{baumann,koehn}@informatik.uni-hamburg.de

Contents of the Course

- Monday:
  - introduction, major features of incremental processing
- today:
  - incremental processing for sequence problems
- Wednesday:
  - incremental processing for structured problems
- Thursday:
  - generating output based on structured and partial input
- Friday:
  - wrap-up and outlook, also based on your questions and interests

Contents for today

- speech recognition as an example of sequence problems
  - time-synchronous Viterbi decoding
- evaluation of incremental processing:
  - stability and timing
- part-of-speech tagging as another example
  - late error detection and handling
    and their consequences on the application
- even simpler: incremental grapheme-to-phoneme conversion as an example of *restart-incrementality*

Short Recap

- incremental processing:

  given minimal input start to produce partial output

- non-monotonicity:

  allow to correct previous mistakes

  - this is necessary in order to generate timely output
  - but what if someone else also acted based on these mistakes?

Incrementality: Limitations

[ ]

Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?

    [ ]

Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?

[f O 6   ]

$\boxed{\text{four}}$

Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?

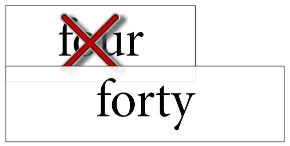
[f O 6 t i:]

$\boxed{\text{four}}$

Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?

[f O 6 t i:]



Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?

[f O 6 t i: n]

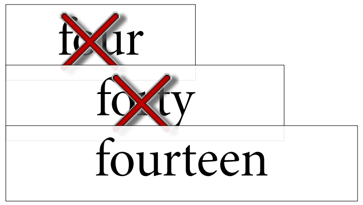
~~four~~

forty

Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?

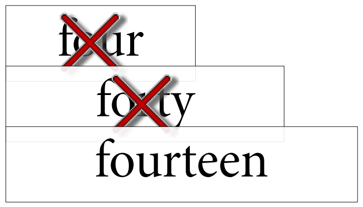
[f O 6 t i: n]



Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?   then [EI dZ 6 z]?

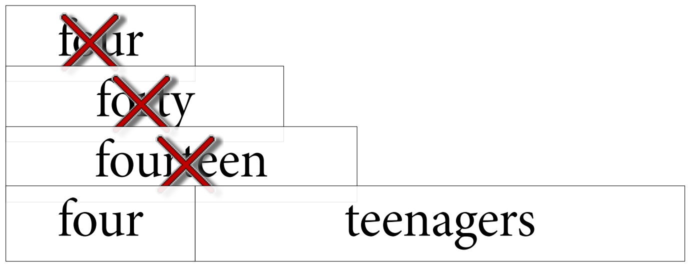
[f O 6 t i: n EI dZ 6 z]



Incrementality: Limitations

- hypotheses are based on *what has been seen so far*
  - later input may result in changes
- example: speech recognition
  - input: [f O 6] → this sounds like "four"!
  - addition of [t i:] → together, this sounds like "forty"!
  - what happens if [n] is next?   then [EI dZ 6 z]?

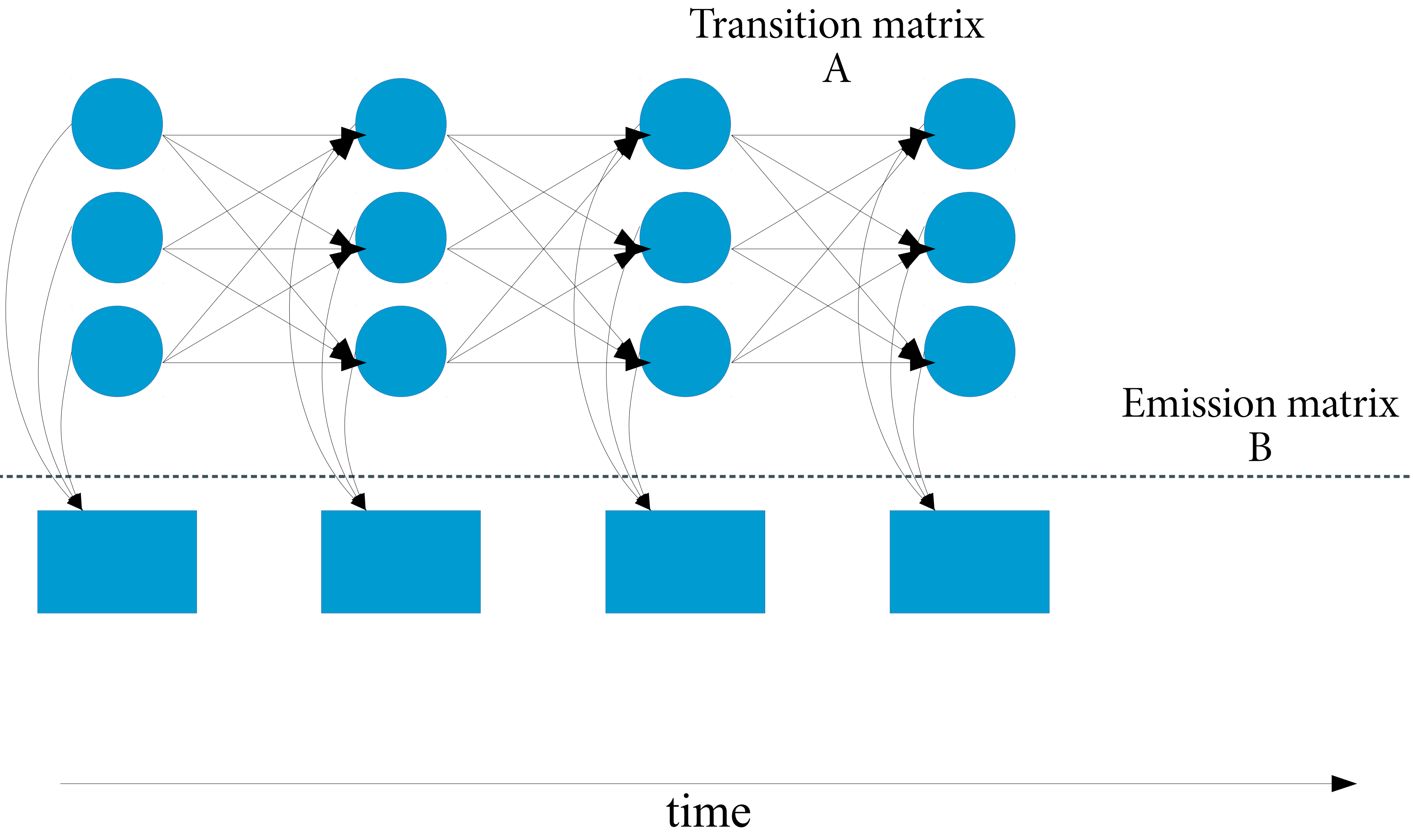
[f O 6 t i: n EI dZ 6 z]



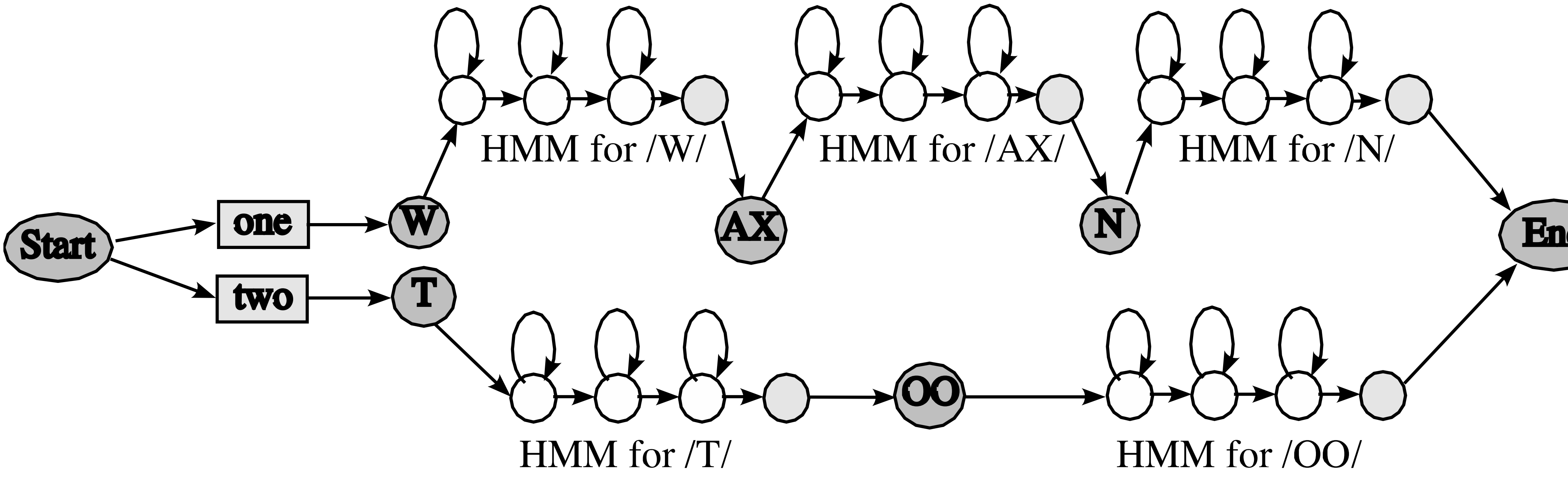
A primer in speech recognition

- chop up speech signal into consecutive frames (e.g. 10 ms)
- devise low-dimensional representation for frames
- score frame sequence against state sequence of a HMM
  - assign what frames belong to which state, given:
    - emission probabilities: how likely does a frame belong to a state (this is the *acoustic model*)
    - transition probabilities: how likely are transitions between states (this are the *language model* and the *pronunciation model*)
- keep a list of *N* best-scoring tokens at any moment in time
- scoring is (most often) performed *time-synchronously* (historically because this reduces memory requirements)

HMM decoding



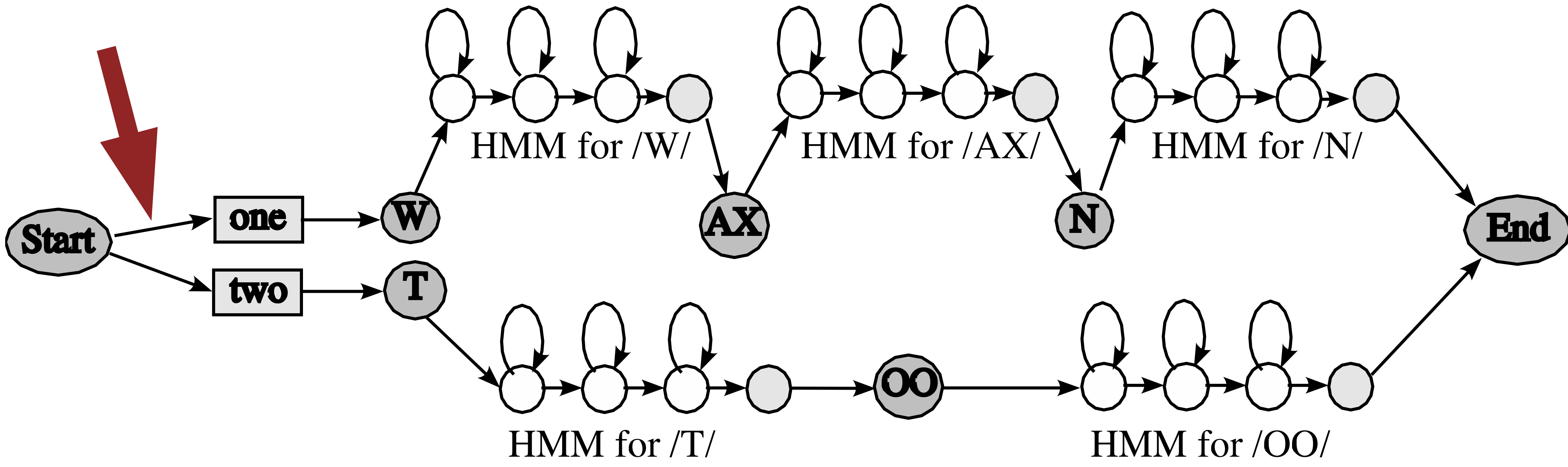
The Search Graph



built from language model (here: S→“one”|“two”),
lexicon (one→/W AX N/, two→/T OO/), and phone models

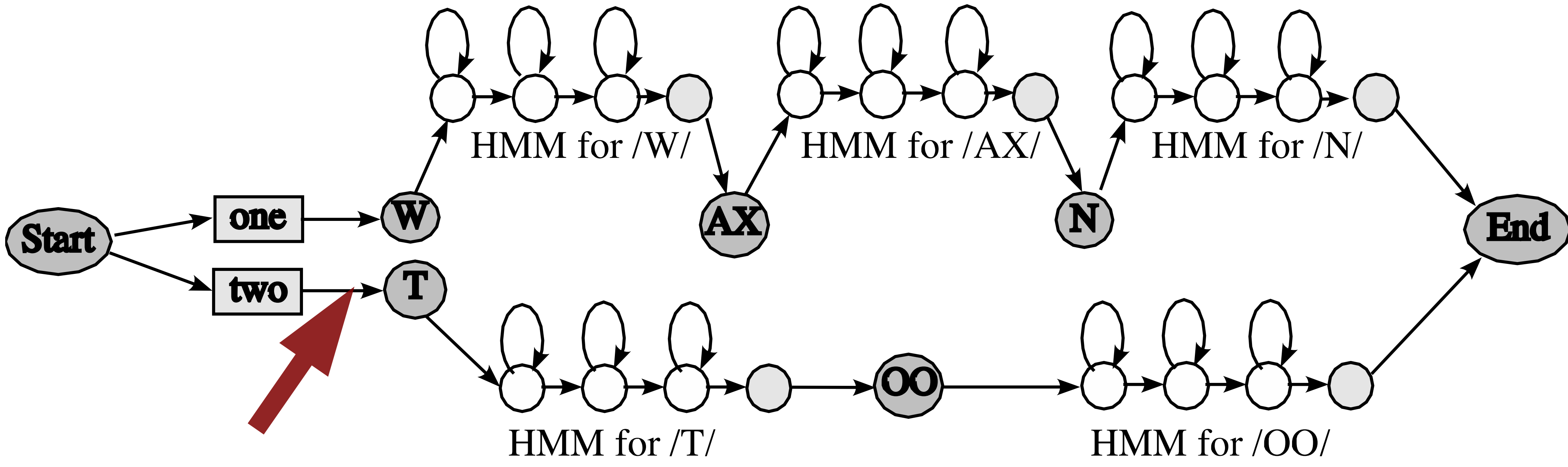
aus: Walker et al., Sphinx-4: A Flexible Open Source Framework for SR, 2004.

The Search Graph



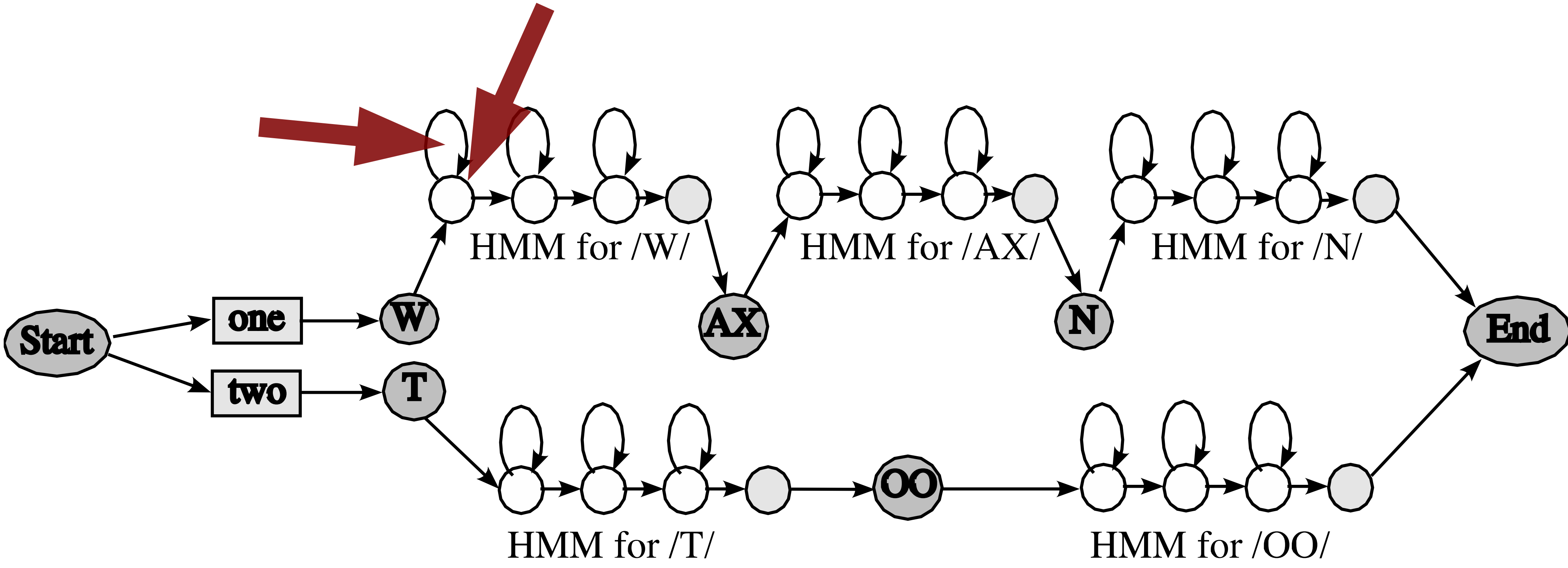
- transition probabilities from language model

The Search Graph



- expansion to sounds from the lexicon

The Search Graph

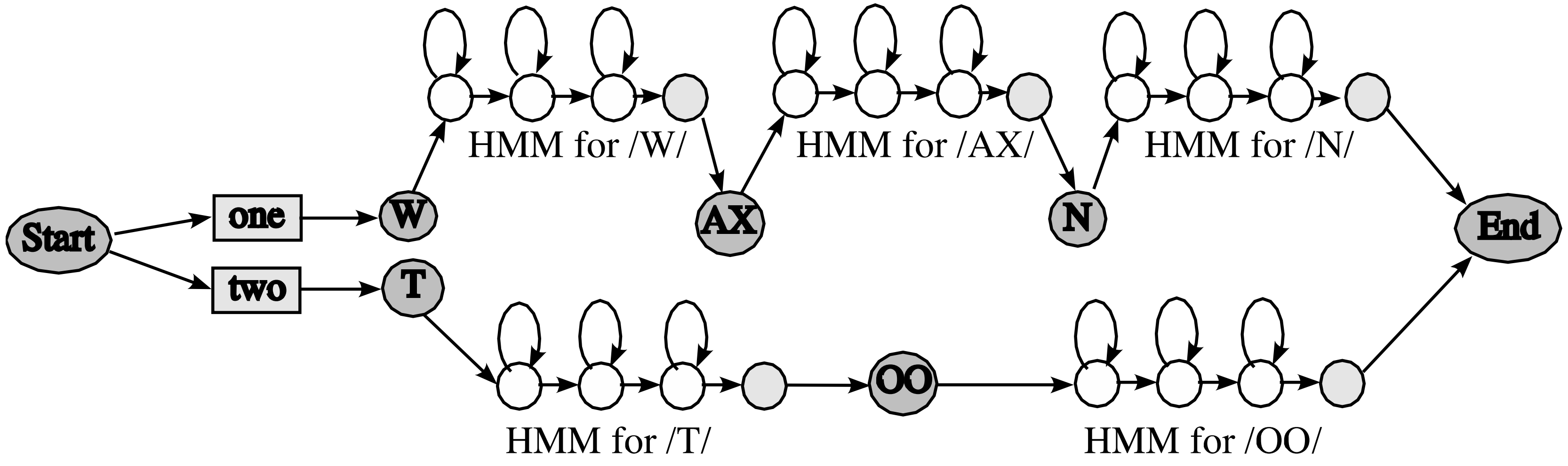


- acoustic model: transition probabilities (A) and emission/observation probabilities (B)

all we need to do is find the most likely
path through the graph

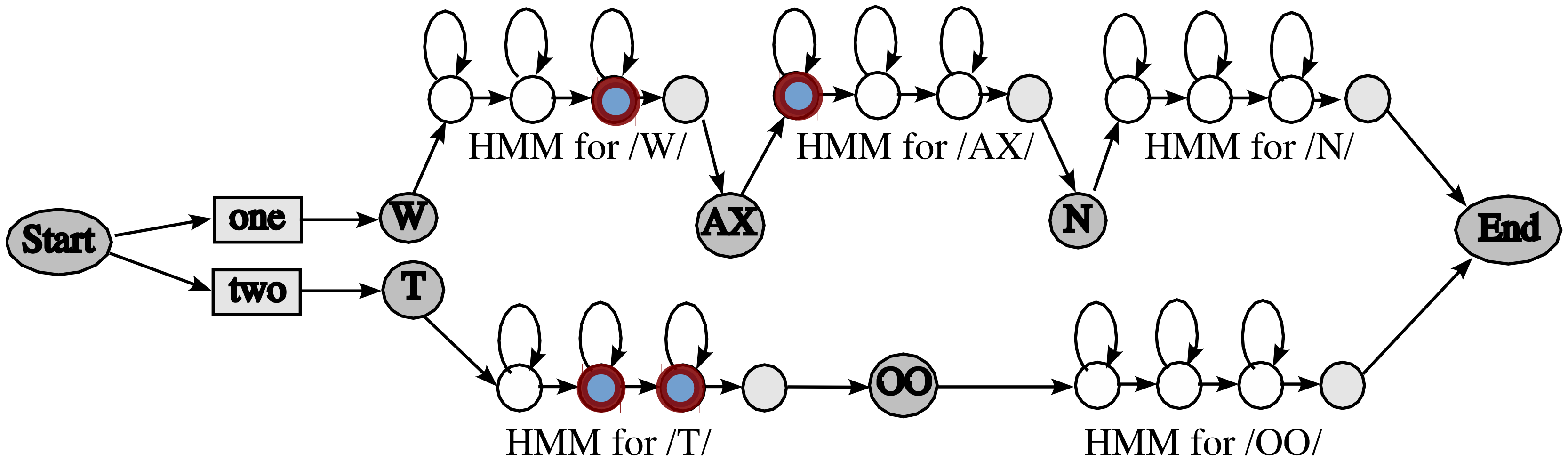
Decoding: Searching the Graph

- we're looking for the path in the graph that
  - distributes the observations to (emitting) phone states
  - while keeping costs at a minimum
    (identical to the highest probability)



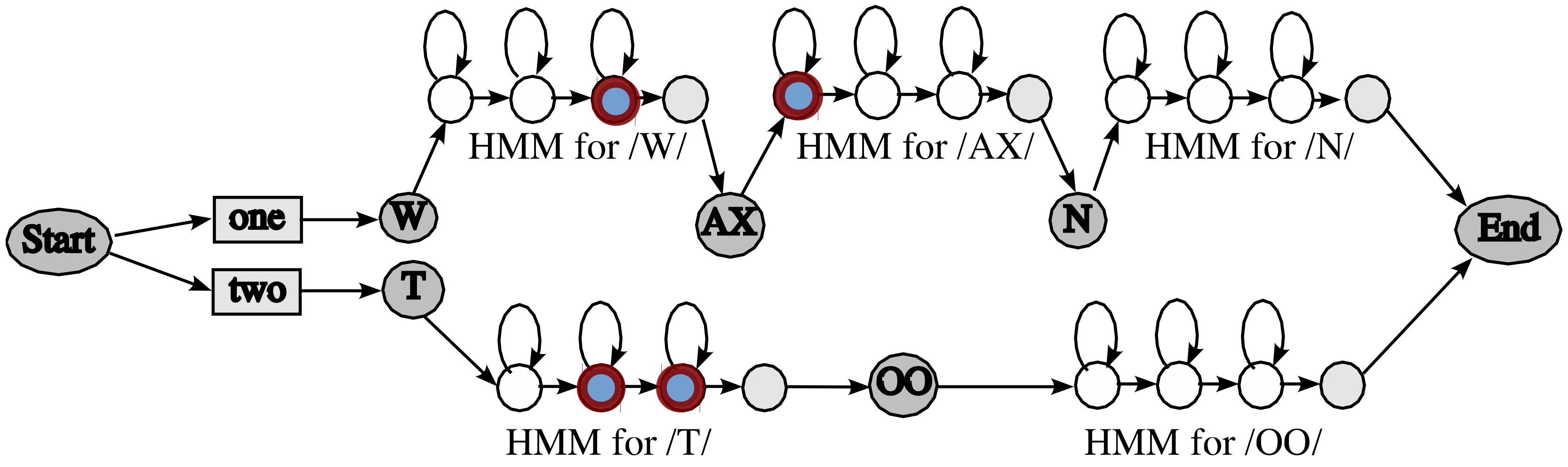
Token-Pass Algorithm: Basic Idea

- time-synchronous search of the observations
  - at every point in time, keep a number of hypotheses, that are represented each by a token
  - generate new tokens from old tokens in every step
  - the winner: best token that reaches the final state in the end

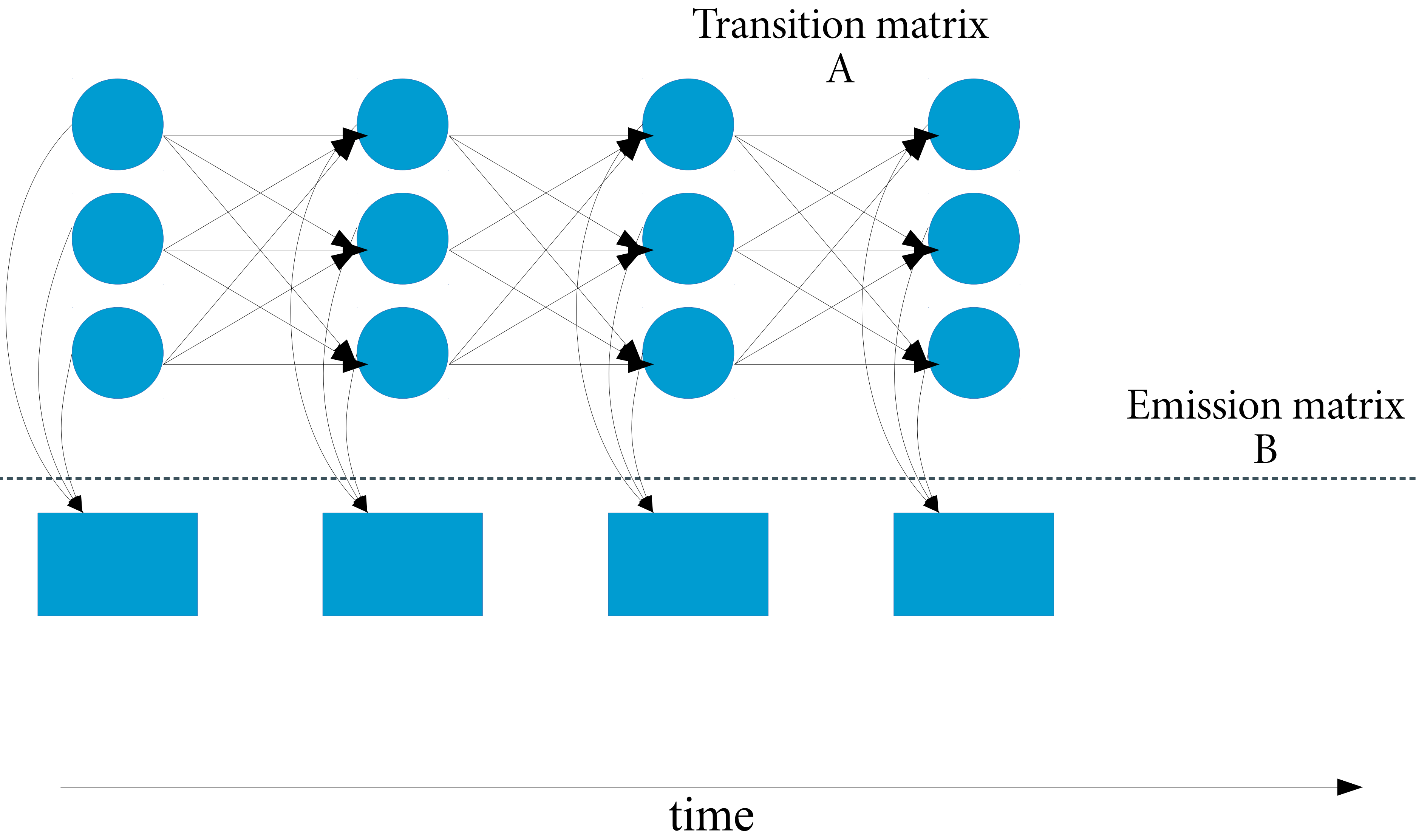


Token-Pass Algorithm: Basic Idea

- every *token*
  - stores the current state in the graph
  - the sum of costs incurred so far
    - possibly differentiated for LM and AM costs
  - details to preceding token (necessary to recover path)



HMM decoding usually performed time-synchronously!



How to “incrementalize” speech recognition

- it's already incremental:
  - at any moment:
    take the best-scoring hypothesis from the token list
  - find the state sequence belonging to this token
  - that's what we want
  - what was best in the last state need not be best in the next state

➔ main challenge is how to reduce the number of changes

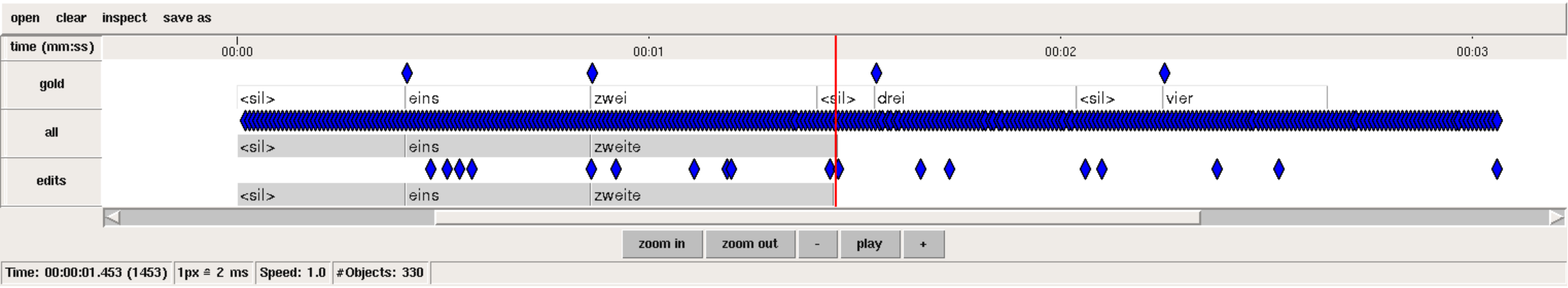
➔ while passing on “good” output as early as possible

➔ i.e, ideally differentiate between “good” and “bad” changes

Video: development of the n-best tokens
(isr-lattice.avi)

The volatility of incremental hypotheses

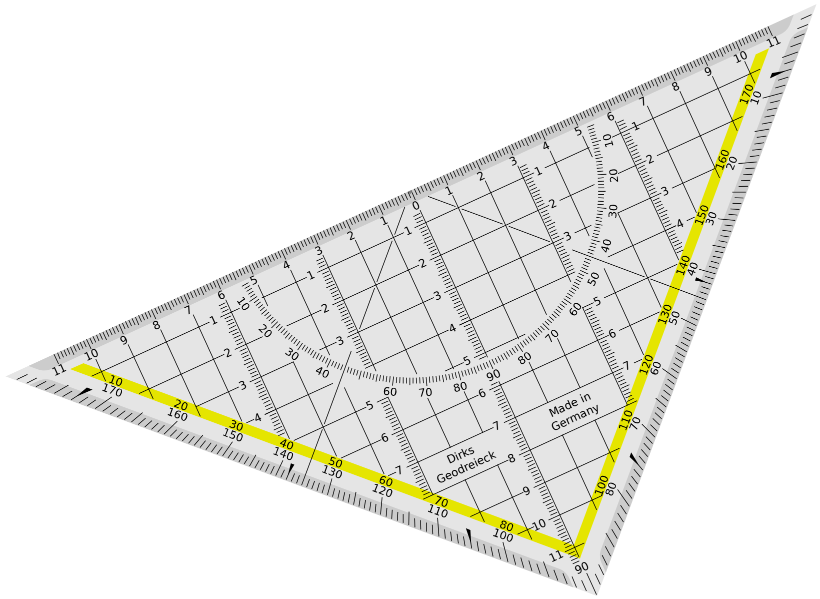
- incremental hypotheses are often only preliminary
  - changes over time – some changes introduce errors



- show video: tedvid.ogv

(Baumann et al., NAACL 2009)

Evaluating incremental speech and language processing



(Baumann et al., Dialogue & Discourse 2011)

Evaluating NLP Systems

Evaluating NLP Systems

- *in-vivo* evaluation in a system (also called extrinsic eval.)

  build a **full system** using our components and measure how well it performs (e.g. user satisfaction, task completion, …)

- *in-vitro* evaluation of components (also intrinsic eval.)

  determine sensible, **generic performance** metrics for individual components (e.g. WER, BLEU, MOS, …)

  – perform performance analyses on (pre-defined?) corpora

- comparison:

  – in-vivo: detailed results, which, however are situation-specifc

  – in-vitro: no guarantee about performance within a full system

Evaluating NLP Systems

- *in-vivo* evaluation in a system (also called extrinsic eval.)

  build a **full system** using our components and measure how well it performs (e.g. user satisfaction, task completion, …)

- *in-vitro* evaluation of components (also intrinsic eval.)

  determine sensible, **generic performance** metrics for individual components (e.g. WER, BLEU, MOS, …)

  – perform performance analyses on (pre-defined?) corpora

- comparison:

  – in-vivo: detailed results, which, however are situation-specifc

  – in-vitro: no guarantee about performance within a full system

Evaluating NLP Systems

- *in-vivo* evaluation in a system (also called extrinsic eval.)

  build a **full system** using our components and measure how well it performs (e.g. user satisfaction, task completion, …)

- *in-vitro* evaluation of components (also intrinsic eval.)

  determine sensible, **generic performance** metrics for individual components (e.g. WER, BLEU, MOS, …)

  – perform performance analyses on (pre-defined?) corpora

- comparison:

  – in-vivo: detailed results, which, however are situation-specifc

  – in-vitro: no guarantee about performance within a full system

Non-incremental in-vitro evaluation

„how to recognize speech“ ← expected result

„how to wreck a nice beach“ ← actual result

- meaningfully compare the two:

Non-incremental in-vitro evaluation

„how to recognize speech“ ← expected result

„**how to wreck a nice beach**“ ← actual result

- meaningfully compare the two:
  - **2** correct, **2** substitutions, **2** insertions → WER = 66 %

Non-incremental in-vitro evaluation

Non-incremental in-vitro evaluation

in general:

- one expected result (gold standard)
- one actual result
- one comparison of the actual to the expected result

- (the above is per item in our corpus, of course we have many problem instances in the corpus)
  - calculate error distributions over the corpus

Incremental in-vitro evaluation

a **sequence** of intermediate results

for every problem instance

a sequence of intermediate results

- how
  how to
  how to wreck
  how to wreck a
  how to recognize
  how to recognize B
  how to wreck a nice beach

a sequence of intermediate results

- how
  how to
  how to wreck
  how to wreck a
  how to recognize
  how to recognize B
  how to wreck a nice beach

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

a sequence of intermediate results

• how	← good
how to	← good
how to wreck	← not so good
how to wreck a	← not so good
how to recognize	← good?
how to recognize B	← partially good
how to wreck a nice beach	← hmpf.

- incremental results develop over time
  → many intermediate results need to be judged

  → **what should they be evaluated against?**

What to compare against: the gold standard

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → if it does, then non-incremental processing is doing something wrong (and should be fixed)

- essentially:

  if the results will turn out to be bad in the end,
  we at least want them to be bad **as soon as possible**,
  and to arrive there **as smoothly as possible**.

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → if it does, then non-incremental processing is doing something wrong (and should be fixed)

- essentially:

  if the results will turn out to be bad in the end,
  we at least want them to be bad **as soon as possible**,
  and to arrive there **as smoothly as possible**.

What to compare against: the gold standard

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → use the processor's **final output** as gold standard for **intermediate results**

  → possibly: limit evaluation to instances where non-incremental processing leads to no/little errors

- all else is covered by non-incremental metrics

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → use the processor's **final output** *how to wreck a nice beach*
  as gold standard for **intermediate results**

  → possibly: limit evaluation to instances where
  non-incremental processing leads to no/little errors

- all else is covered by non-incremental metrics

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → use the processor's **final output** as gold standard for **intermediate results**

  how to wreck a nice beach

  how to wreck

  → possibly: limit evaluation to instances where non-incremental processing leads to no/little errors

  how to recognize

- all else is covered by non-incremental metrics

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → use the processor's **final output** as gold standard for **intermediate results**

  how to wreck a nice beach

  how to wreck

  → possibly: limit evaluation to instances where non-incremental processing leads to no/little errors

  how to recognize

- all else is covered by non-incremental metrics

What to compare against: the gold standard

- incremental processing cannot systematically outperform non-incremental processing

  → use the processor's **final output** as gold standard for **intermediate results**

  how to wreck a nice beach

  how to wreck

  → possibly: limit evaluation to instances where non-incremental processing leads to no/little errors

  how to recognize

- all else is covered by non-incremental metrics

What to compare against

- how
  how to
  how to wreck
  how to wreck a
  how to recognize
  how to recognize B
  how to wreck a nice beach

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

What to compare against

• how	← good
how to	← good
how to wreck	← …
how to wreck a	← …
how to recognize	← this is inconsistent!
how to recognize B	← even worse?
how to wreck a nice beach	← this is wrong, but that's covered by WER

WER=66%

→ we're primarily interested in the *evolution over time*, less in the final result (which is covered by non-incremental metrics)

Evolution of incremental hypotheses

• recognizer setting A:	• recognizer setting B:
ha	
how	
hot	how
how to	how
how torr	how to
how to wreck	how to
how to wreck	how to wreck
how to wreck a	how to wreck
how to wreck on	how to wreck a
how to recognize	how to wreck a
how to wreck on ice	how to wreck a
how to wreck a nice	how to wreck a
how to recognize bee	how to wreck a nice
how to wreck on ice P	how to wreck a nice
how to wreck a nice beach	how to wreck a nice beach

Evolution of incremental hypotheses

- recognizer setting A:
  ha
  how
  hot
  how to
  how torr
  how to wreck
  how to wreck
  how to wreck a
  how to wreck on
  how to recognize
  how to wreck on ice
  how to wreck a nice
  how to recognize bee
  how to wreck on ice P
  how to wreck a nice beach

setting A more often „changes it's mind“

- recognizer setting B:
  how
  how
  how to
  how to
  how to wreck
  how to wreck
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a nice
  how to wreck a nice
  how to wreck a nice beach

Evolution of incremental hypotheses

- recognizer setting A:
  ha
  how
  hot
  how to
  how torr
  how to wreck
  how to wreck
  how to wreck a
  how to wreck on
  how to recognize
  how to wreck on ice
  how to wreck a nice
  how to recognize bee
  how to wreck on ice P
  how to wreck a nice beach

setting A more often „changes it's mind“

- recognizer setting B:
  how
  how
  how to
  how to
  how to wreck
  how to wreck
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a nice
  how to wreck a nice
  how to wreck a nice beach

less change is better

Evolution of incremental hypotheses

- recognizer setting A:
  ha
  how
  hot
  how to
  how torr
  how to wreck
  how to wreck
  how to wreck a
  how to wreck on
  how to recognize
  how to wreck on ice
  how to wreck a nice
  how to recognize bee
  how to wreck on ice P
  how to wreck a nice beach

setting A more often „changes it's mind“

- recognizer setting B:
  how
  how
  how to
  how to
  how to wreck
  how to wreck
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a nice
  how to wreck a nice
  how to wreck a nice beach

setting A is „faster“

Evolution of incremental hypotheses

• recognizer setting A:	• recognizer setting B:
ha	
how	
hot	how
how to	how
how torr	how to
how to wreck	how to
how to wreck	how to wreck
how to wreck a	how to wreck
how to wreck on	how to wreck a
how to recognize	how to wreck a
how to wreck on ice	how to wreck a
how to wreck a nice	how to wreck a
how to recognize bee	how to wreck a nice
how to wreck on ice P	how to wreck a nice
how to wreck a nice beach	how to wreck a nice beach

setting A more often „changes it's mind"

faster is better

setting A is „faster"

Evolution of incremental hypotheses

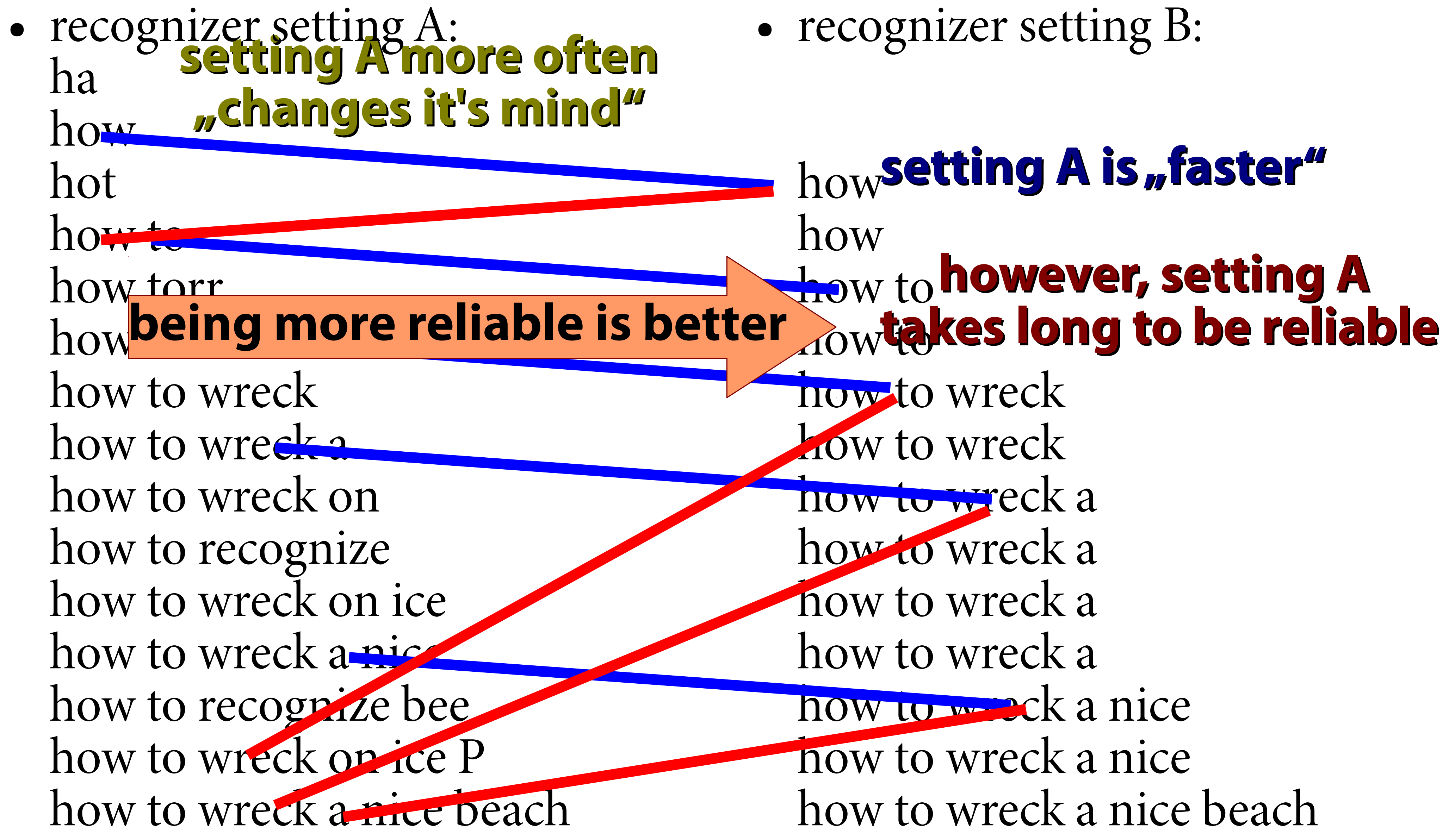
• recognizer setting A:	• recognizer setting B:
ha	
how	
hot	how
how to	how
how torr	how to
how to wreck	how to
how to wreck	how to wreck
how to wreck a	how to wreck
how to wreck on	how to wreck a
how to recognize	how to wreck a
how to wreck on ice	how to wreck a
how to wreck a nice	how to wreck a
how to recognize bee	how to wreck a nice
how to wreck on ice P	how to wreck a nice
how to wreck a nice beach	how to wreck a nice beach

setting A more often „changes it's mind“

setting A is „faster“

however, setting A takes long to be reliable

Evolution of incremental hypotheses



- recognizer setting A:
  ha
  how
  hot
  how to
  how torr
  how
  how to wreck
  how to wreck a
  how to wreck on
  how to recognize
  how to wreck on ice
  how to wreck a nice
  how to recognize bee
  how to wreck on ice P
  how to wreck a nice beach

- recognizer setting B:
  how
  how
  how to
  how to
  how to wreck
  how to wreck
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a nice
  how to wreck a nice
  how to wreck a nice beach

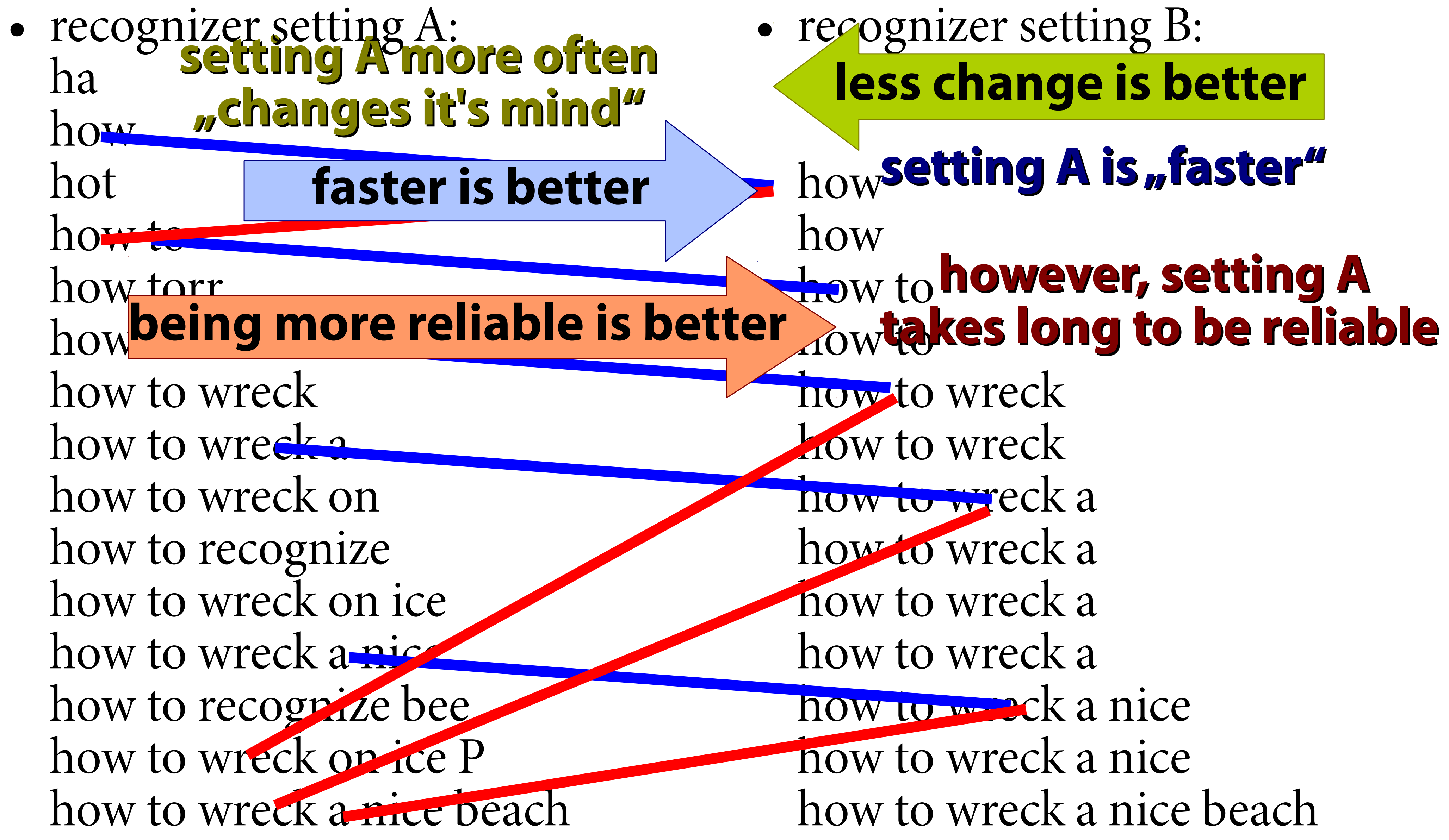
setting A more often „changes it's mind“

setting A is „faster“

however, setting A takes long to be reliable

being more reliable is better

Evolution of incremental hypotheses


setting A more often „changes it's mind“
faster is better
being more reliable is better
less change is better
setting A is „faster“
however, setting A takes long to be reliable

- recognizer setting A:
  ha
  how
  hot
  how to
  how torr
  how to
  how to wreck
  how to wreck a
  how to wreck on
  how to recognize
  how to wreck on ice
  how to wreck a nice
  how to recognize bee
  how to wreck on ice P
  how to wreck a nice beach

- recognizer setting B:
  how
  how
  how to
  how to
  how to wreck
  how to wreck
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a
  how to wreck a nice
  how to wreck a nice
  how to wreck a nice beach

the **fundamental trade-off** of incremental processing

Release early, release often ⇒ release with flaws

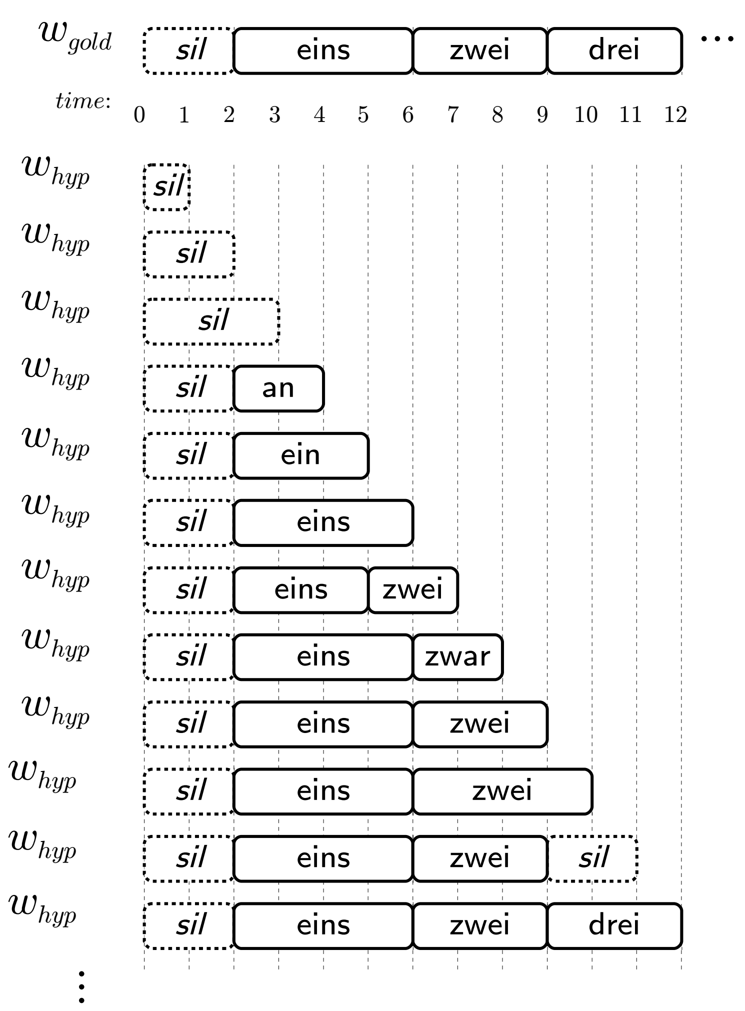
- the earlier results are generated,
  the more likely they will turn out to be wrong

→ **timeliness/stability trade-off**

What's special in Incremental Evaluation?

- incremental processing results in a **sequence of results**
- what should we compare against? (gold standard)
  - final output is good enough
  - limit to cases where final result is sensible
- we're interested in the **evolution** of this sequence
  - timing, and stability of content

A Reduced Example



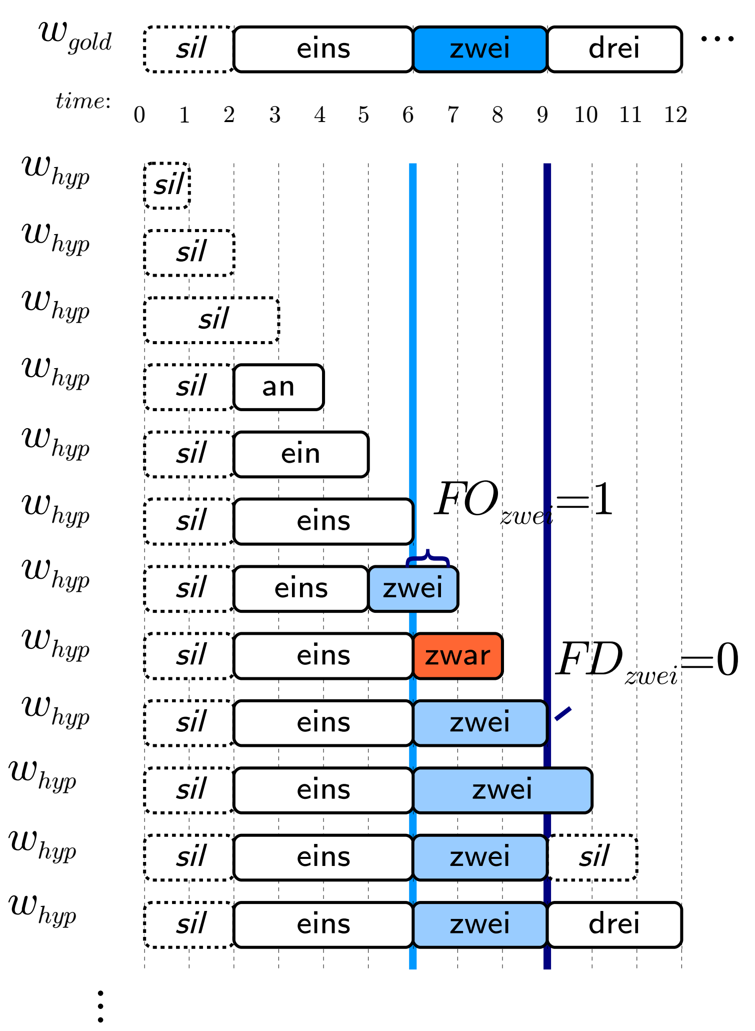
w_{gold} is final hypothesis

two dimensions:

- time we reason **at**: ↓
- time we reason **about**: →

w_{hyp_t} is the word sequence hypothesized at time t

Measuring Timing



when do we find out about a word?

first occurrence: FO

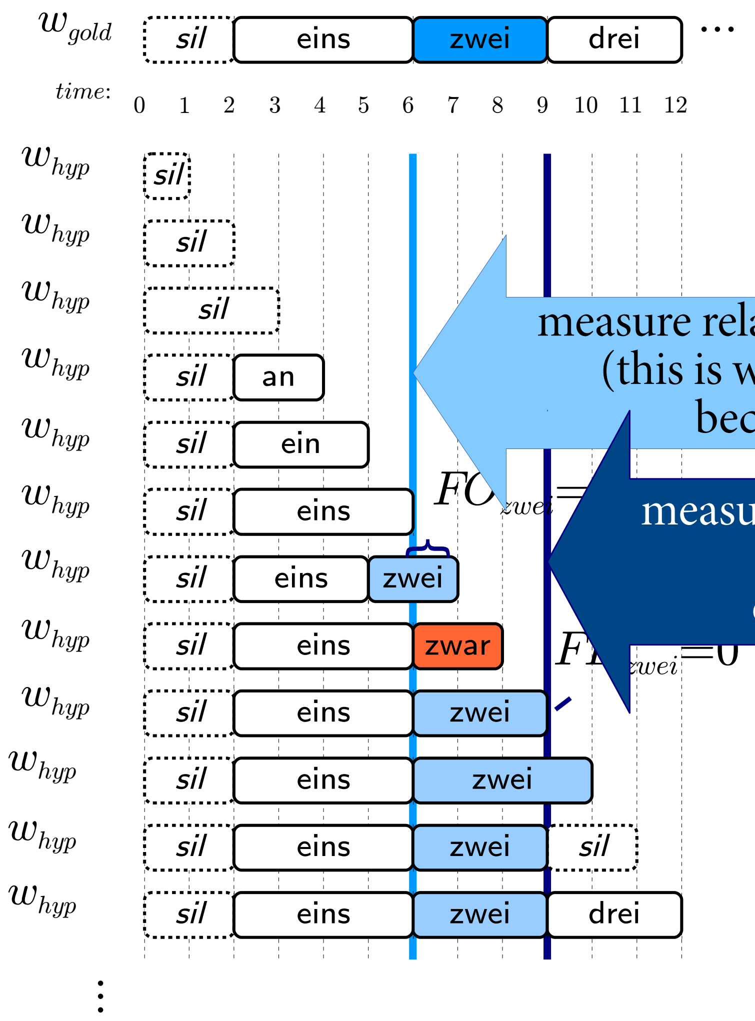
when do we become certain about a word?

final decision: FD

we measure per word

→ averages are important

Measuring Timing



when do we find out about a word?

first o[illegible]e: FO

[illegible] certain

decision: FD

[illegible] word

→ aver[illegible]rtant

measure relative to beginning of word (this is when first evidence became available)

measure relative to end of the word (this is when all evidence is available)

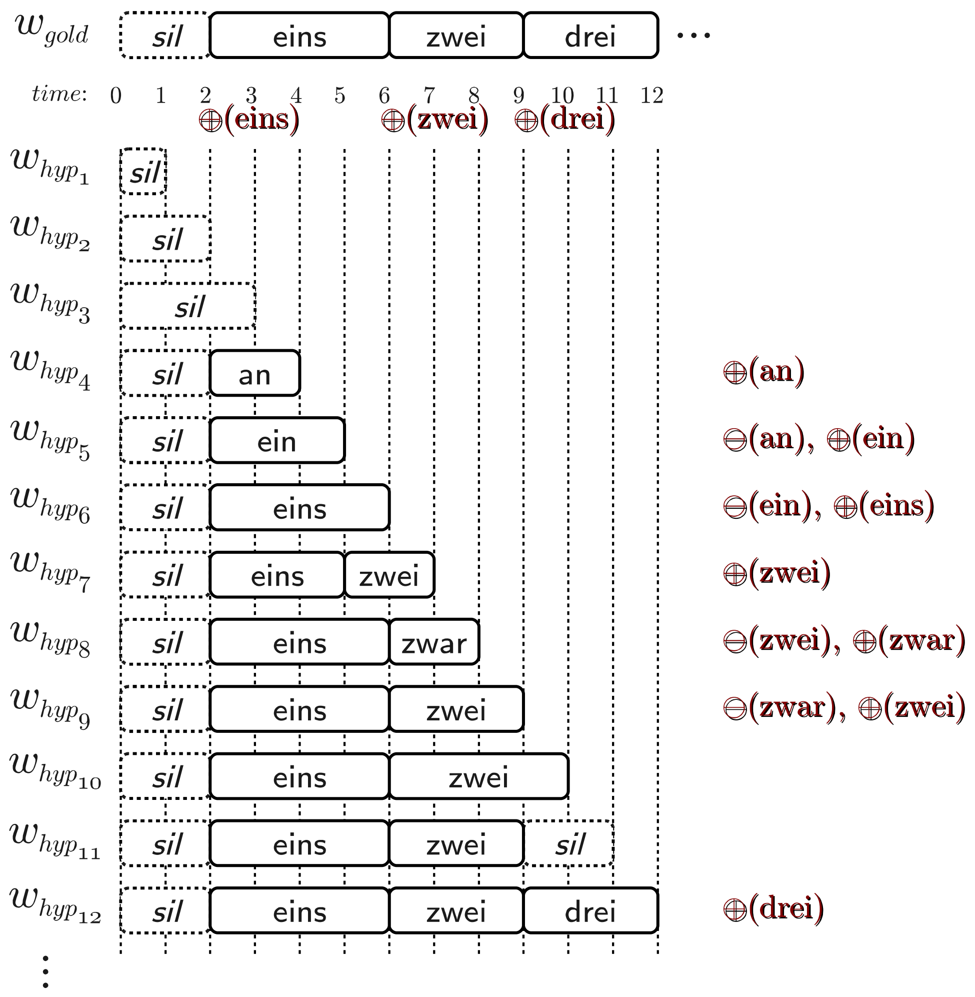
first

final

Measuring Timing

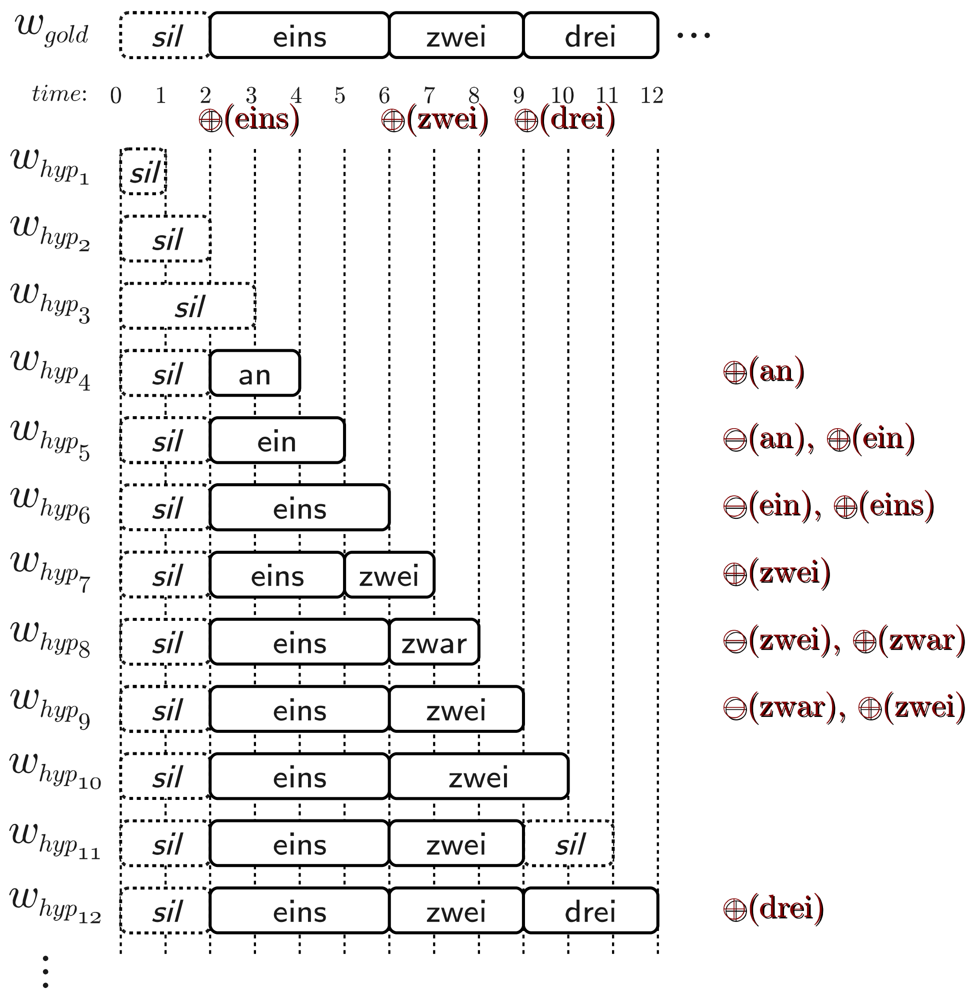
- In general (not just for words):
  - measure the first detection of an occurrence relative to the true beginning of the underlying event
  - measure the final decision for an occurrence relative to the true ending of the underlying event
- depending on the use case we may care for:
  - if we want to assume as soon as posisble → low FO
  - if we want to know as soon as possible → low FD

Edits: a way of measuring stability



w_{gold} sil eins zwei drei …
time: 0 1 2 3 4 5 6 7 8 9 10 11 12
⊕(eins) ⊕(zwei) ⊕(drei)
w_{hyp_1} sil
w_{hyp_2} sil
w_{hyp_3} sil
w_{hyp_4} sil an ⊕(an)
w_{hyp_5} sil ein ⊖(an), ⊕(ein)
w_{hyp_6} sil eins ⊖(ein), ⊕(eins)
w_{hyp_7} sil eins zwei ⊕(zwei)
w_{hyp_8} sil eins zwar ⊖(zwei), ⊕(zwar)
w_{hyp_9} sil eins zwei ⊖(zwar), ⊕(zwei)
$w_{hyp_{10}}$ sil eins zwei
$w_{hyp_{11}}$ sil eins zwei sil
$w_{hyp_{12}}$ sil eins zwei drei ⊕(drei)
⋮

Edits: a way of measuring stability



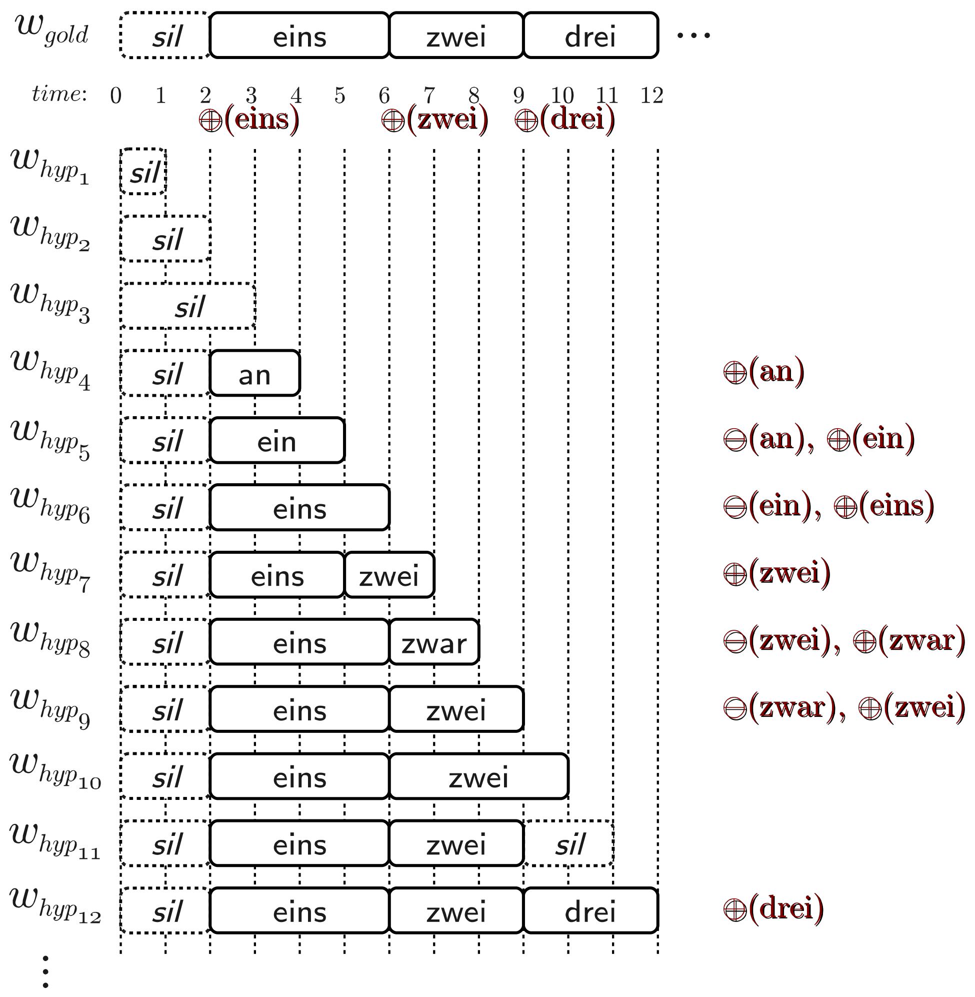
w_{gold} sil eins zwei drei …
time: 0 1 2 3 4 5 6 7 8 9 10 11 12
⊕(eins) ⊕(zwei) ⊕(drei)
w_{hyp_1} sil
w_{hyp_2} sil
w_{hyp_3} sil
w_{hyp_4} sil an — ⊕(an)
w_{hyp_5} sil ein — ⊖(an), ⊕(ein)
w_{hyp_6} sil eins — ⊖(ein), ⊕(eins)
w_{hyp_7} sil eins zwei — ⊕(zwei)
w_{hyp_8} sil eins zwar — ⊖(zwei), ⊕(zwar)
w_{hyp_9} sil eins zwei — ⊖(zwar), ⊕(zwei)
$w_{hyp_{10}}$ sil eins zwei
$w_{hyp_{11}}$ sil eins zwei sil
$w_{hyp_{12}}$ sil eins zwei drei — ⊕(drei)
⋮

changes to the hypothesis:
add, *delete* (maybe *revise*)

ideally: one *add* per word
in fact: **edit overhead**

$$\mathbf{EO} = \frac{|\textit{unnecessary edits}|}{|\textit{edits}|}$$

Edits: a way of measuring stability



w_{gold} sil eins zwei drei ⋯
time: 0 1 2 3 4 5 6 7 8 9 10 11 12
⊕(eins) ⊕(zwei) ⊕(drei)
w_{hyp_1} sil
w_{hyp_2} sil
w_{hyp_3} sil
w_{hyp_4} sil an ⊕(an)
w_{hyp_5} sil ein ⊖(an), ⊕(ein)
w_{hyp_6} sil eins ⊖(ein), ⊕(eins)
w_{hyp_7} sil eins zwei ⊕(zwei)
w_{hyp_8} sil eins zwar ⊖(zwei), ⊕(zwar)
w_{hyp_9} sil eins zwei ⊖(zwar), ⊕(zwei)
$w_{hyp_{10}}$ sil eins zwei
$w_{hyp_{11}}$ sil eins zwei sil
$w_{hyp_{12}}$ sil eins zwei drei ⊕(drei)
⋮

changes to the hypothesis:
add, *delete* (maybe *revise*)

ideally: one *add* per word
in fact: **edit overhead**

$$\mathbf{EO} = \frac{|\mathit{unnecessary\ edits}|}{|\mathit{edits}|}$$

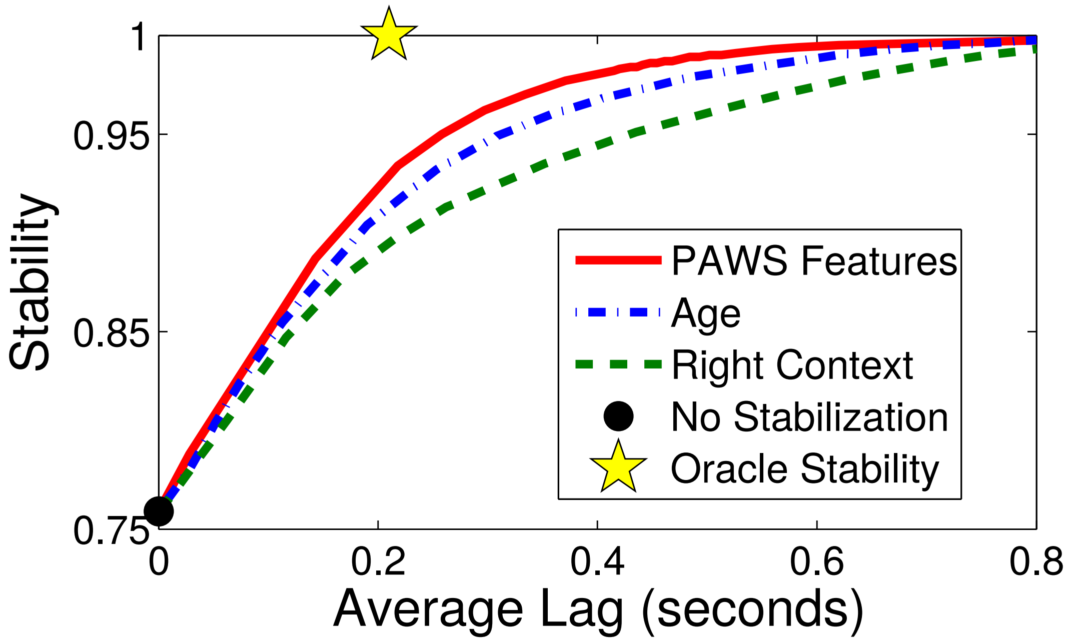
- typically, there is a trade-off:
  reducing edit overhead results in timing deterioration

Measuring Stability

- In general (not just for words):
  - count the minimal number of edits (additions of incremental units) that are necessary to reach the final results
  - compare this to the actual number of edits needed
- fewer edits → higher stability
- improve stability:
  - skip or defer edits until you're more certain about them
  - but: fewer edits → fewer incremental results
  - it's better to pass on all edits and to tag their reliability (downside: higher computational cost)

How to reduce edit overhead

- simple: hold back any edit until it has reached a certain age (or has been cancelled in the meantime)
  - set the age threshold according to your desired edit overhead
- hard: do a lot of machine learning to get slightly better



Baumann et al. 2009 (NAACL), McGraw&Gruenstein 2012 (Interspeech)

Reliability of partial results

- quick hypotheses come at the cost of making (intermittent) mistakes
- we want hypotheses to be reliable (or even better: have an estimate of reliability)
- Edit Survival Rate:
  - an edit that is hypothesized and remains in the result „lives forever“
  - other edits “die off” in favour of alternate edit-hypotheses after a certain time
  - we plot the survival rate over time and use the age of an edit as a reliability estimate

Baumann et al., 2009 (NAACL-HLT), Selfridge et al., 2011 (SigDial)

Experiment:
Off-the-shelf ASRs in a dialog domain

Baumann et al., 2016 (IWSLT)

The Setup

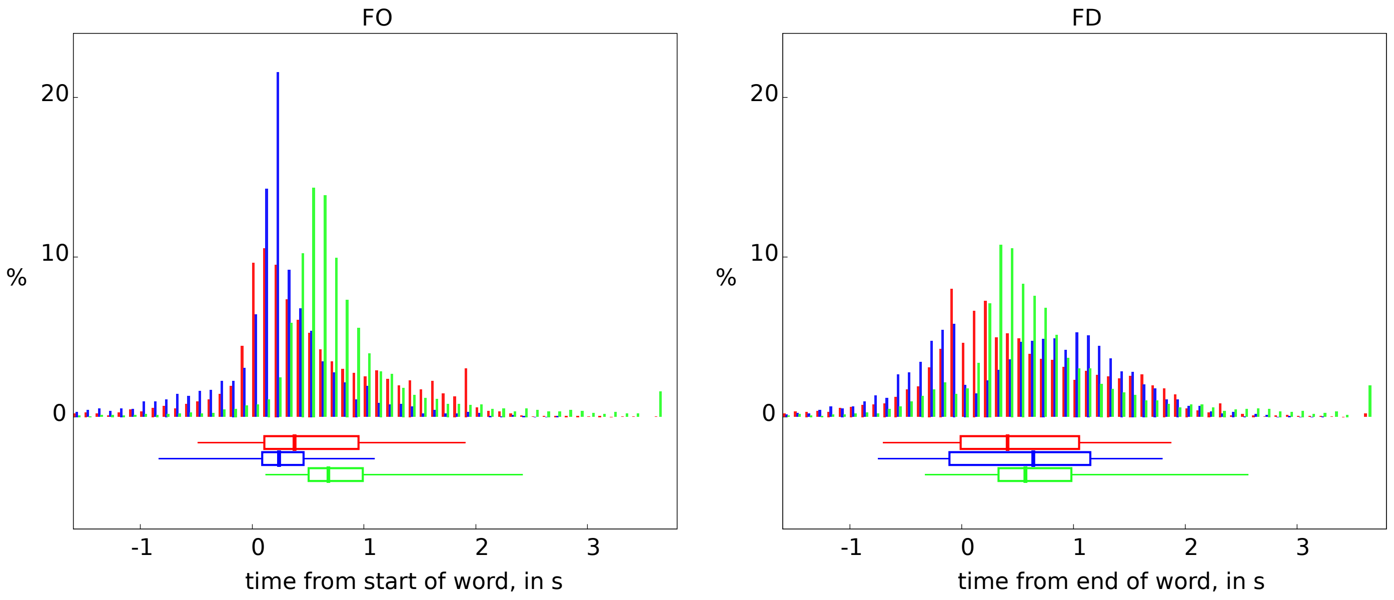
- Google Speech API
- Sphinx-4 with most recent off-the-shelf models (5.2PTM, generic English LM)
- Kaldi server trained with the Voxforge recipe (both acoustic and language models)

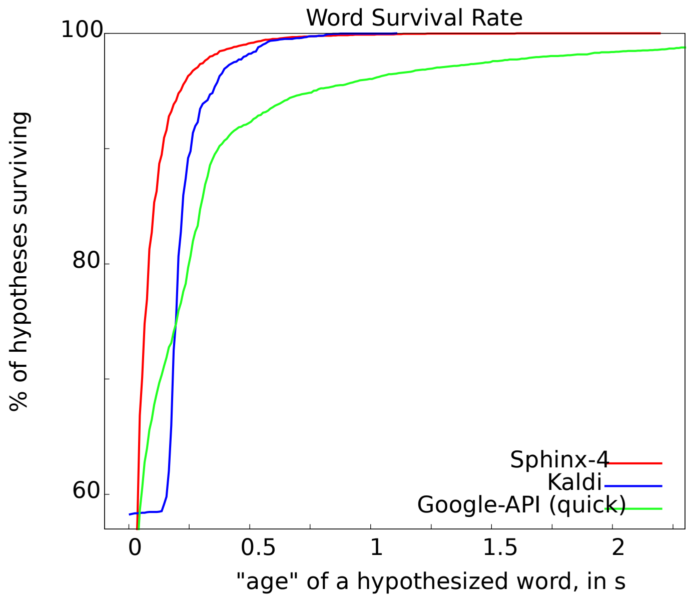
- uniformly available via InproTK

- English test data from a (human-human) dialog domain

Google is limited to ~500 incremental API calls per day
Baumann & Schlangen, 2012; inprotk.sf.net

Incremental Metrics





- Sphinx and Kaldi somewhat earlier than Google
- Google has many very late changes
- Sphinx results become reliable quickly
- Kaldi seems to do some internal age-thresholding as can be seen in the survival rate (cmp. Baumann et al., 2009)

Baumann et al., 2016 (IWSLT)

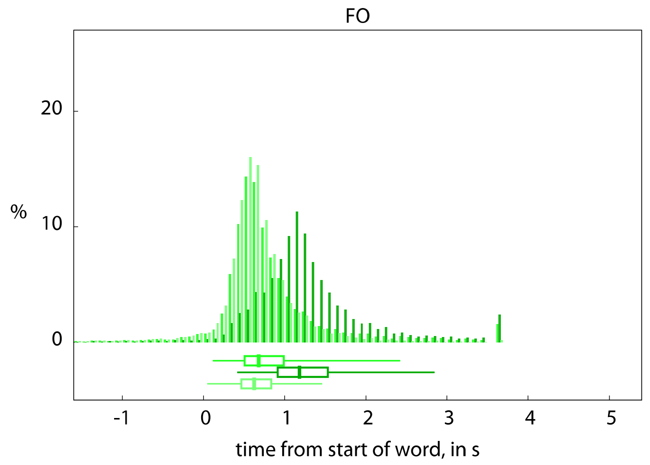
A close look at Google's results

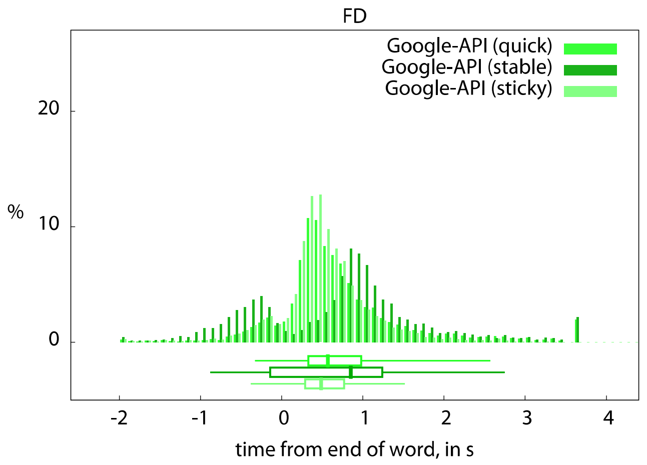
Google divides its results into a "stable" and an "unstable" part

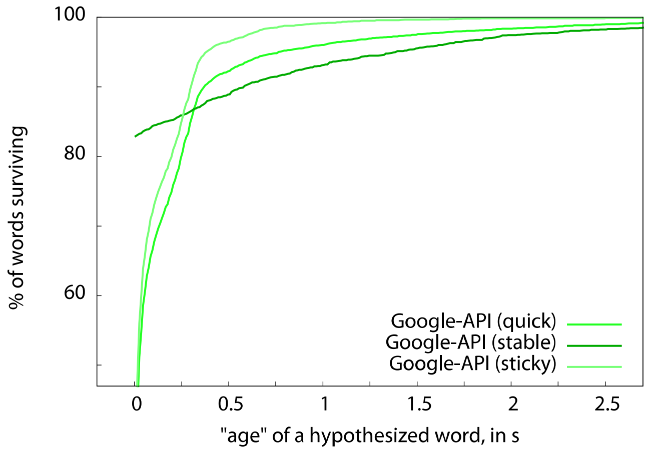
- so far we had been looking at everything

Google apparently rescores the result post-hoc

- this explains the extremely late changes
  - ignoring them has little impact (2%) on WER







Baumann et al., 2016 (IWSLT)

A second example:
Incremental part-of-speech tagging

Köhn (2009)

POS-Tagging

- Straight-forward HMMs

  The man etc. pp.

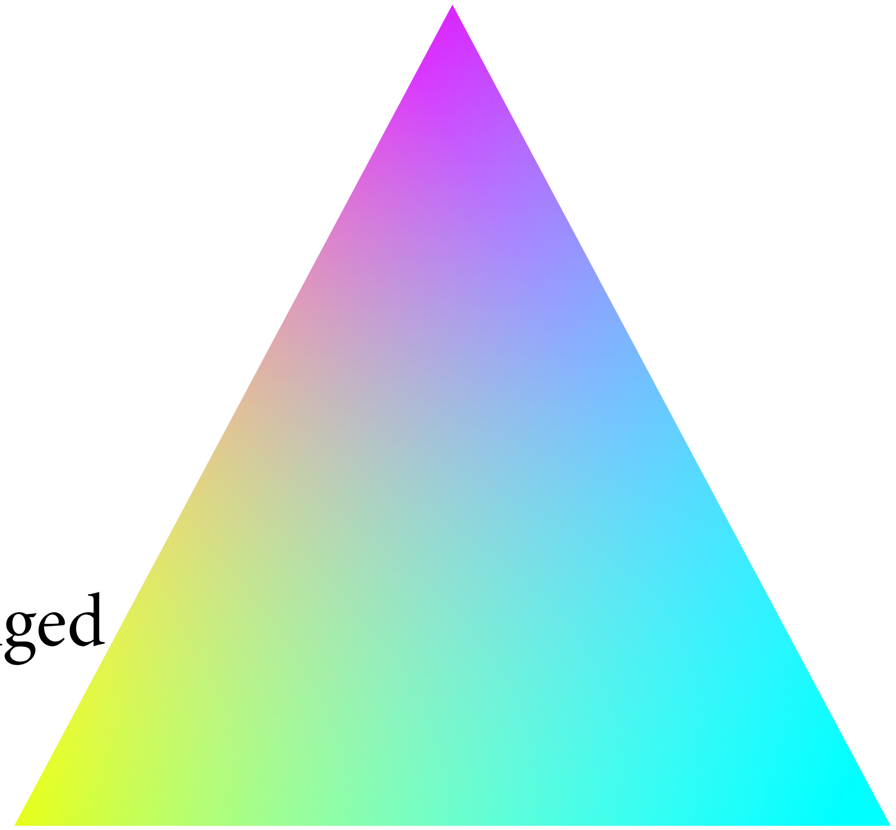
  DET NN etc. pp.

- Decoding techniques for HMMs:

	Given	Predict	Output
– Filtering	$o_1 \ldots o_k$	s_k	prob. dist.
– Smoothing	$o_1 \ldots o_n$	s_k	prob. dist.
– Viterbi	$o_1 \ldots o_n$	$s_1 \ldots s_n$	best sequence

Incremental POS Tagging

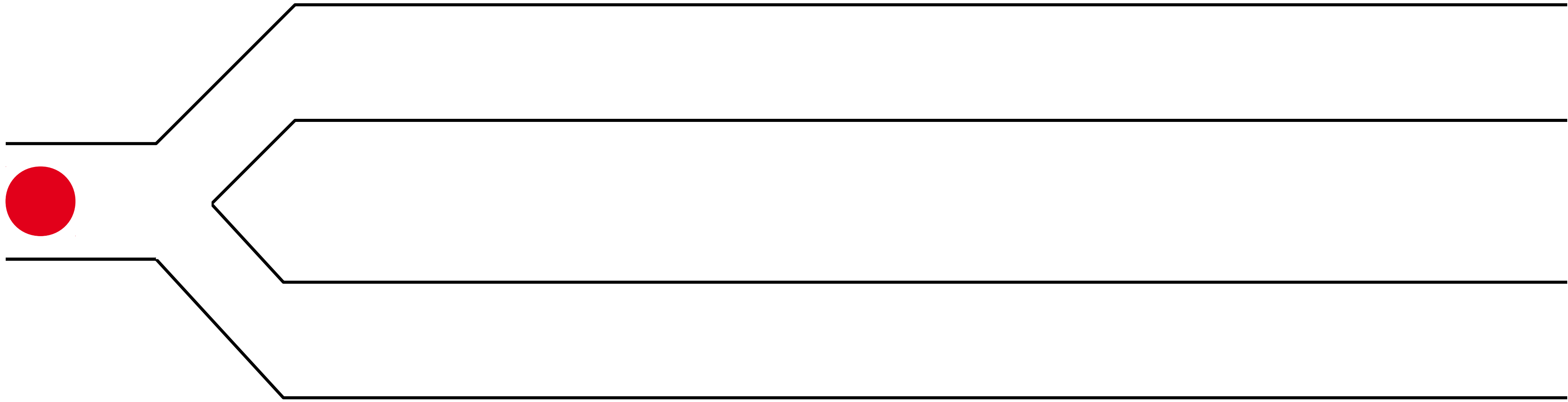
- Non-incremental POS tagging: nearly solved, boring
  - State of the art: ~97.4% Majority baseline: ~90%
- Incremental: What do we lose?
- Timely & Monotonic:
  - Accuracy drop 0.7-2.5%
- Monotonic & Accurate:
  - Delay of 1-2 words
- Timely & Accurate
  - 2.7%-6.9% chance of output changed
- OR: pass on 2-best POS tags



Beuck et al. (2011)

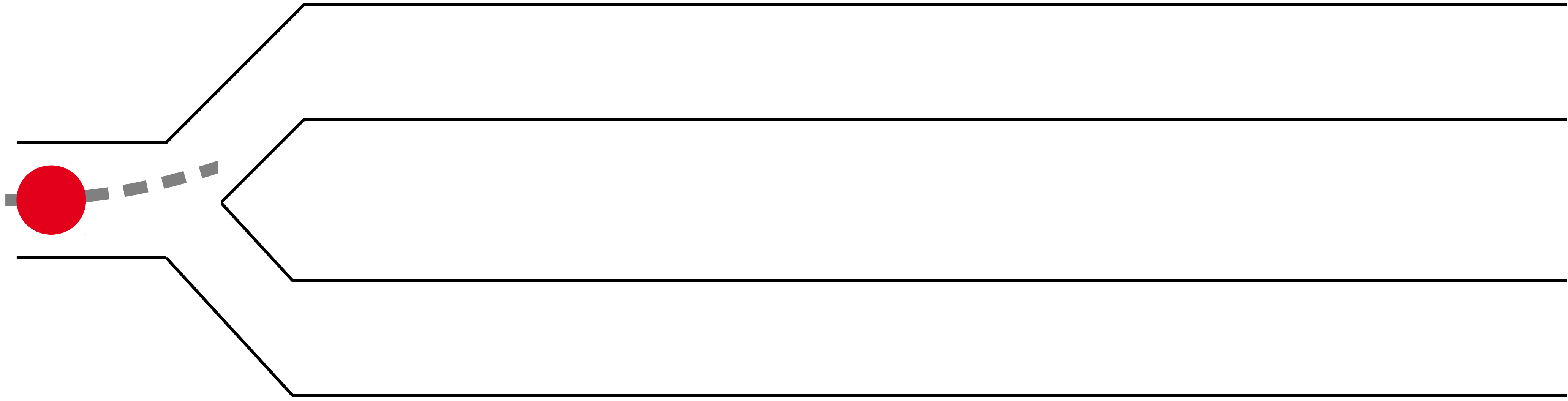
Decoding Strategies by Example

- You walk in the nice hills of Tirol
- Your GPS device tells you where you are
- A sensor provides it with raw data



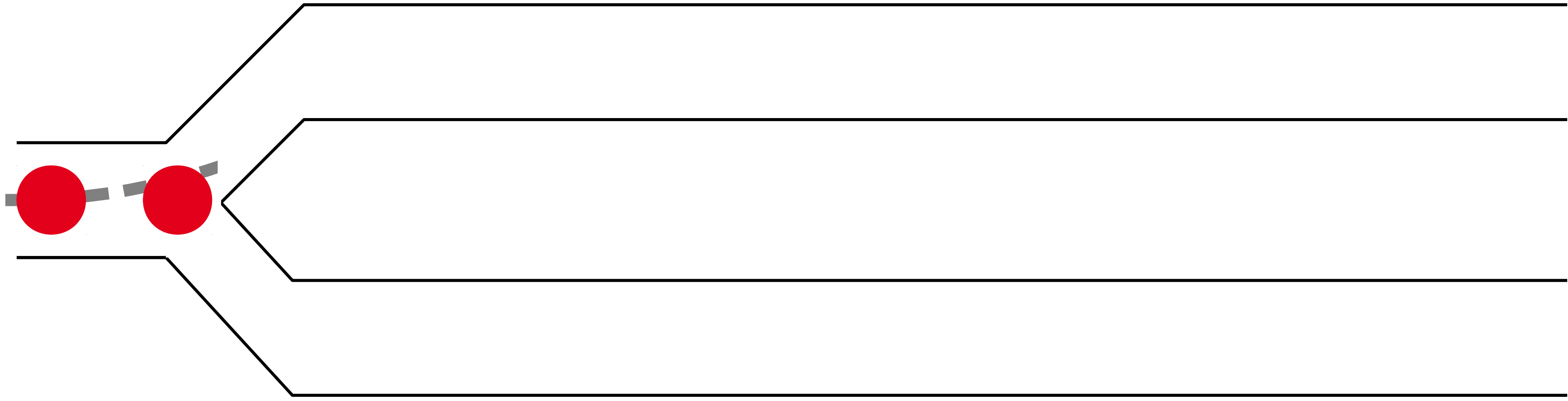
Decoding Strategies by Example

- Filtering:



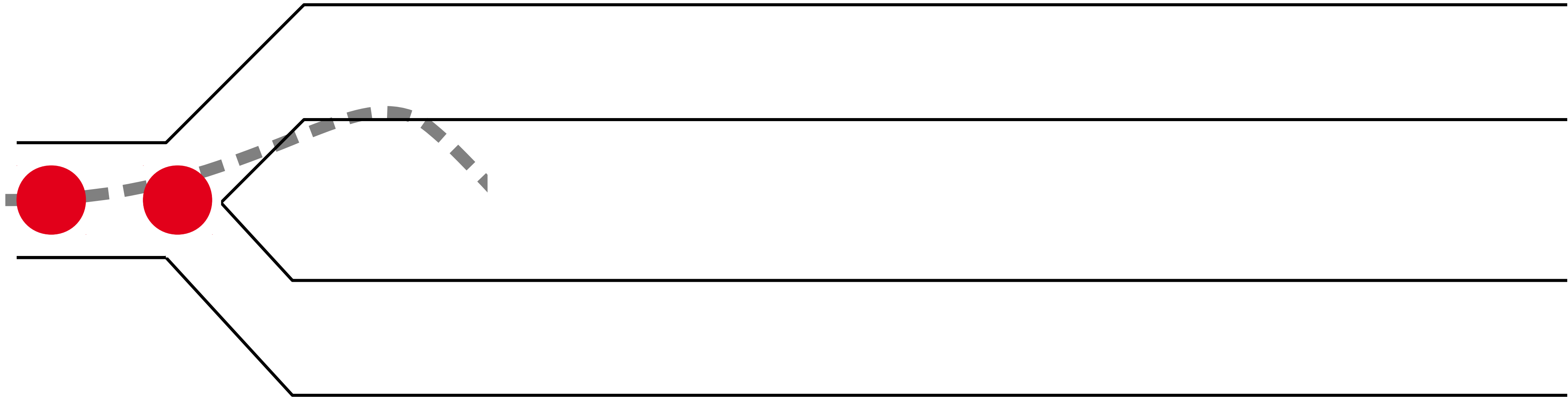
Decoding Strategies by Example

- Filtering:



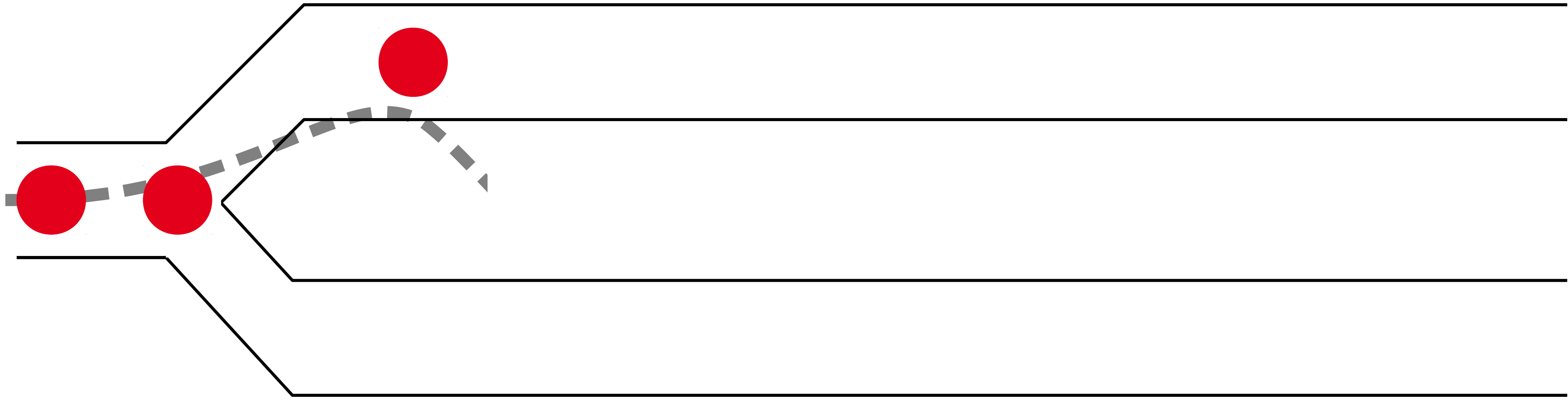
Decoding Strategies by Example

- Filtering:



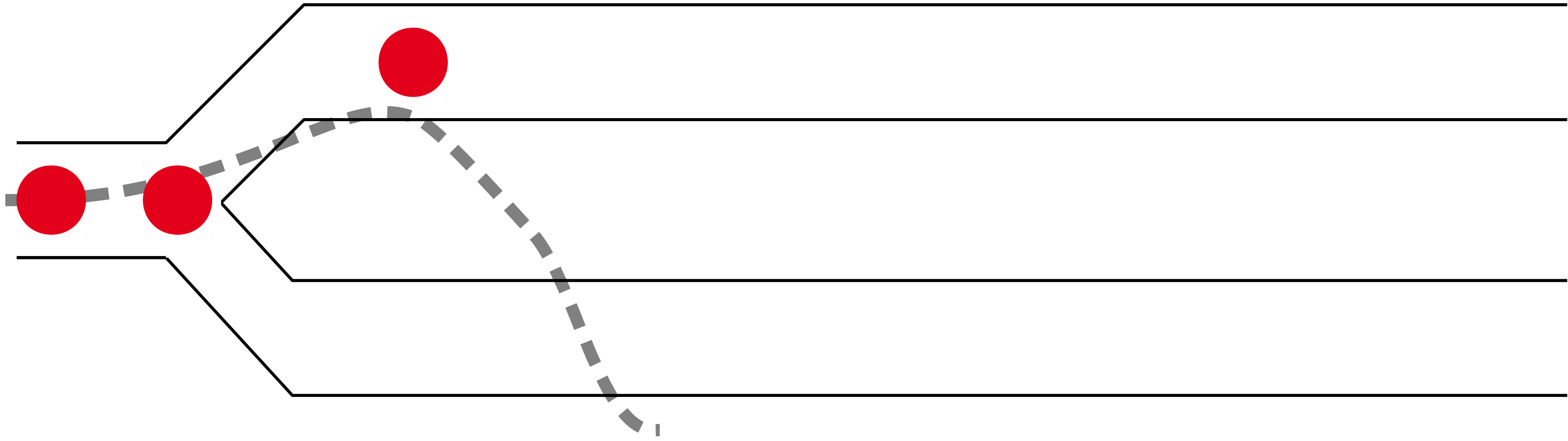
Decoding Strategies by Example

- Filtering:



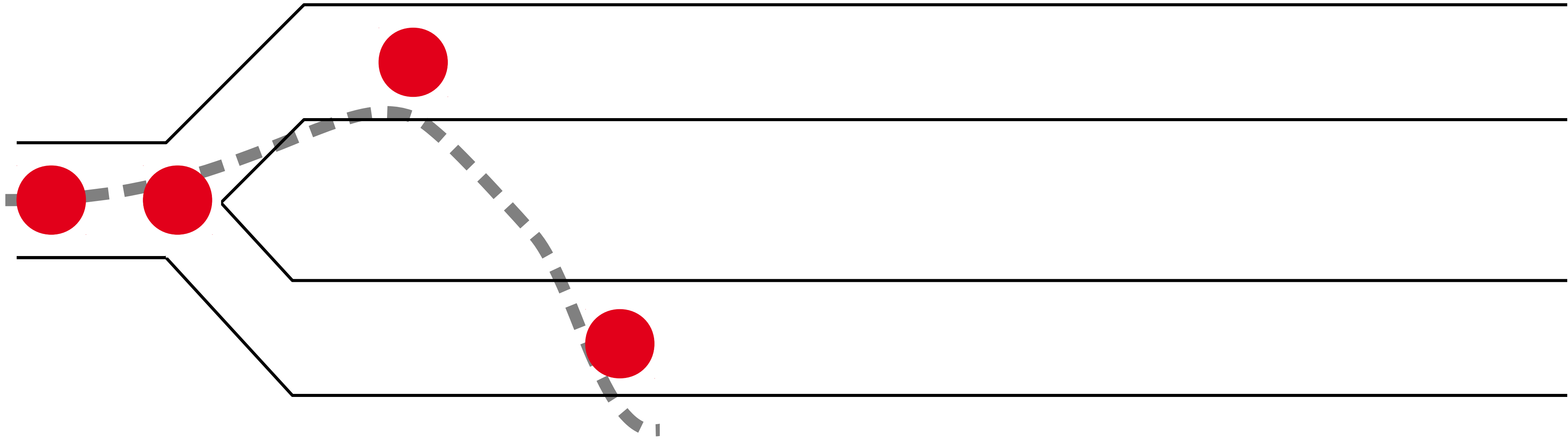
Decoding Strategies by Example

- Filtering:



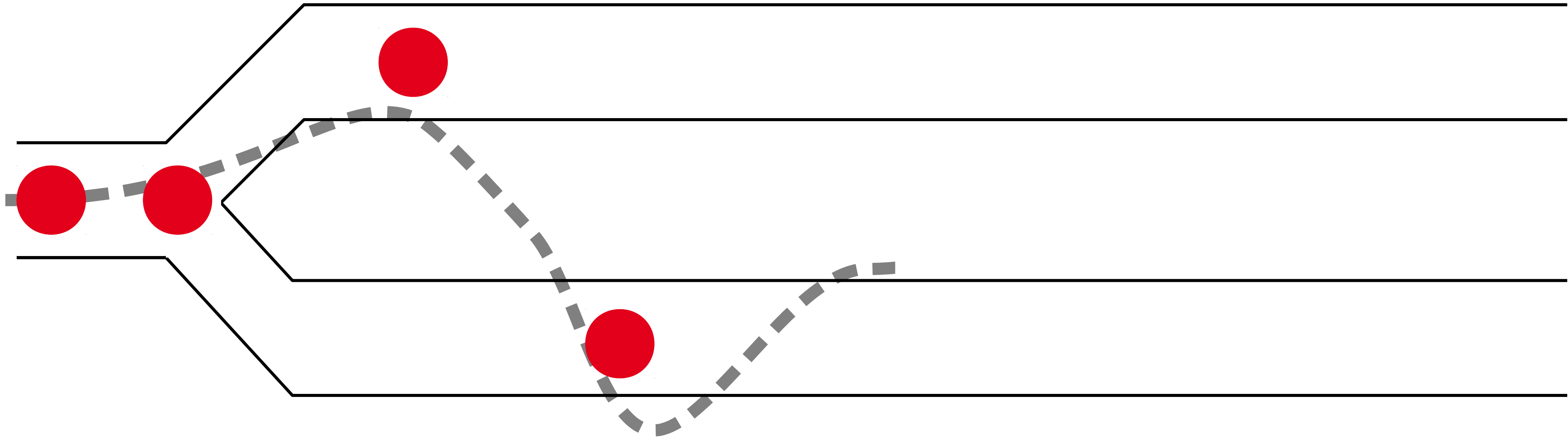
Decoding Strategies by Example

- Filtering:



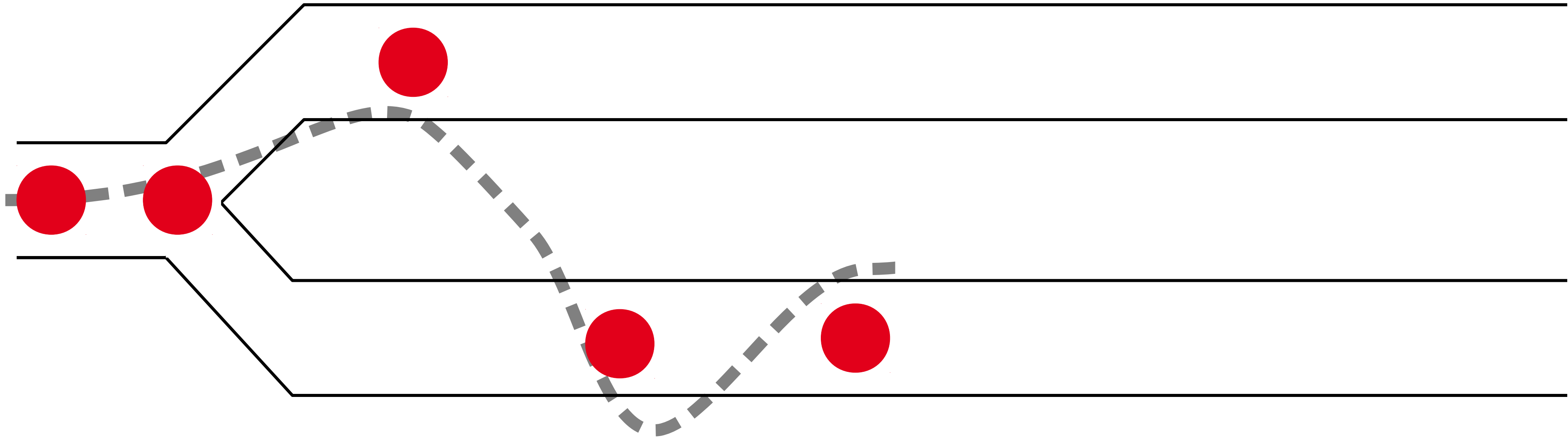
Decoding Strategies by Example

- Filtering:



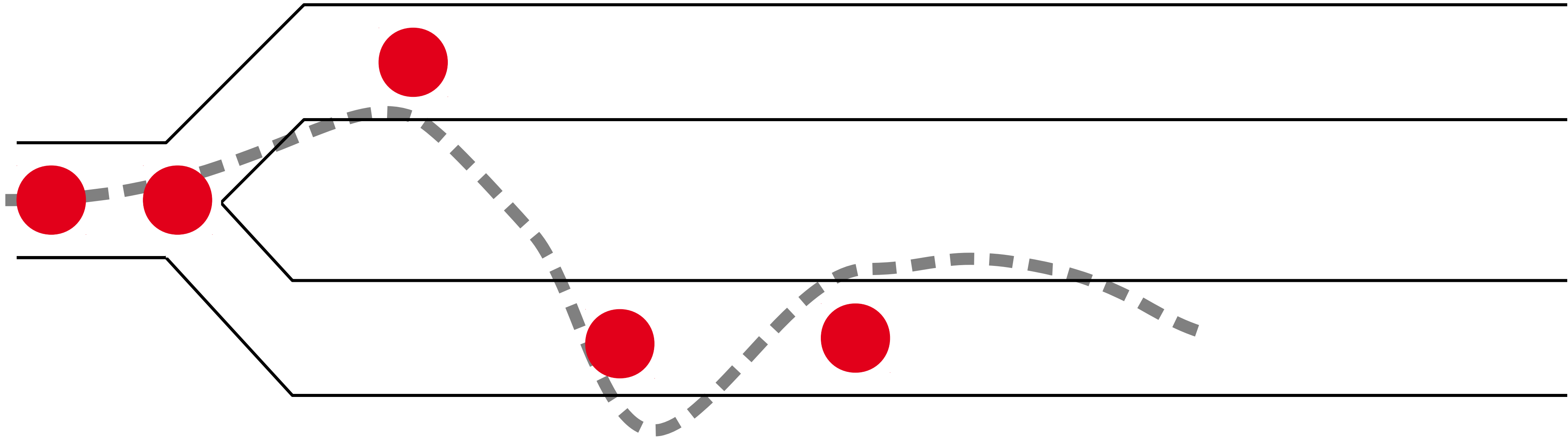
Decoding Strategies by Example

- Filtering:



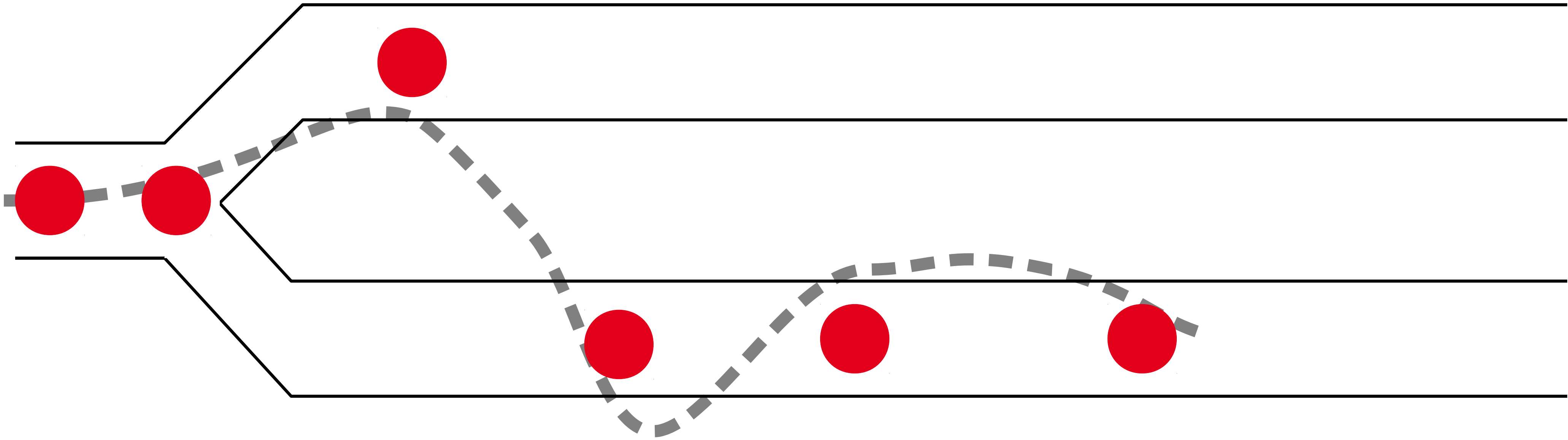
Decoding Strategies by Example

- Filtering:



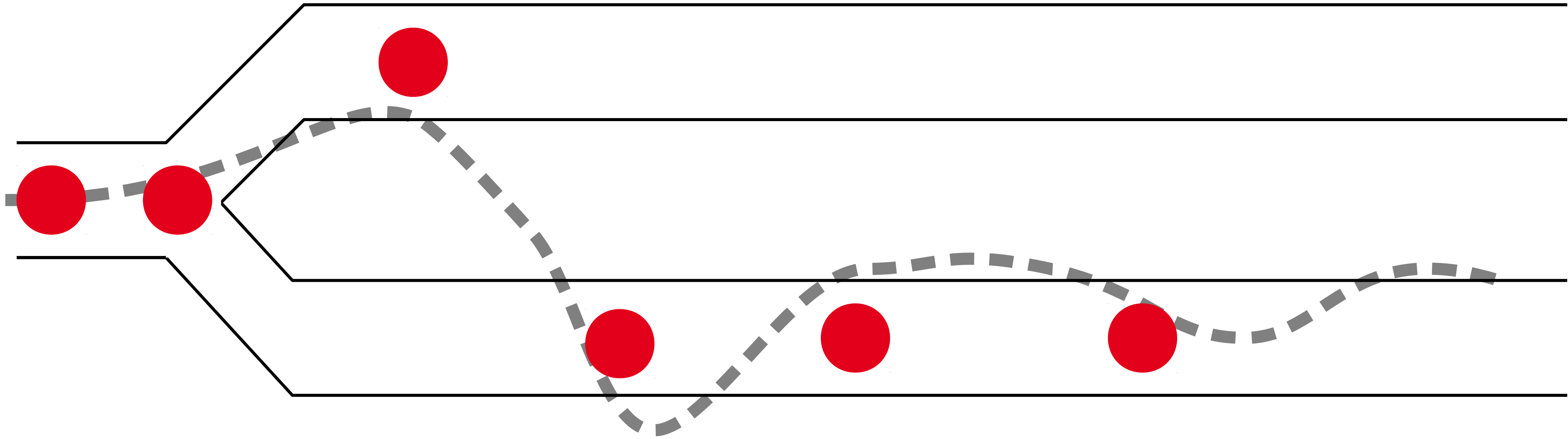
Decoding Strategies by Example

- Filtering:



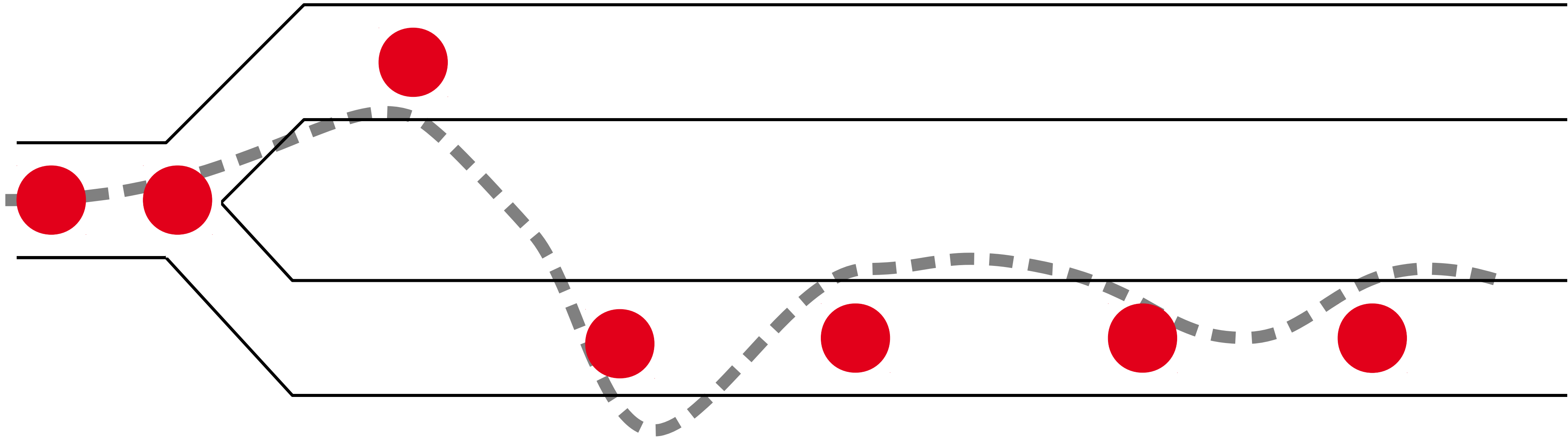
Decoding Strategies by Example

- Filtering:



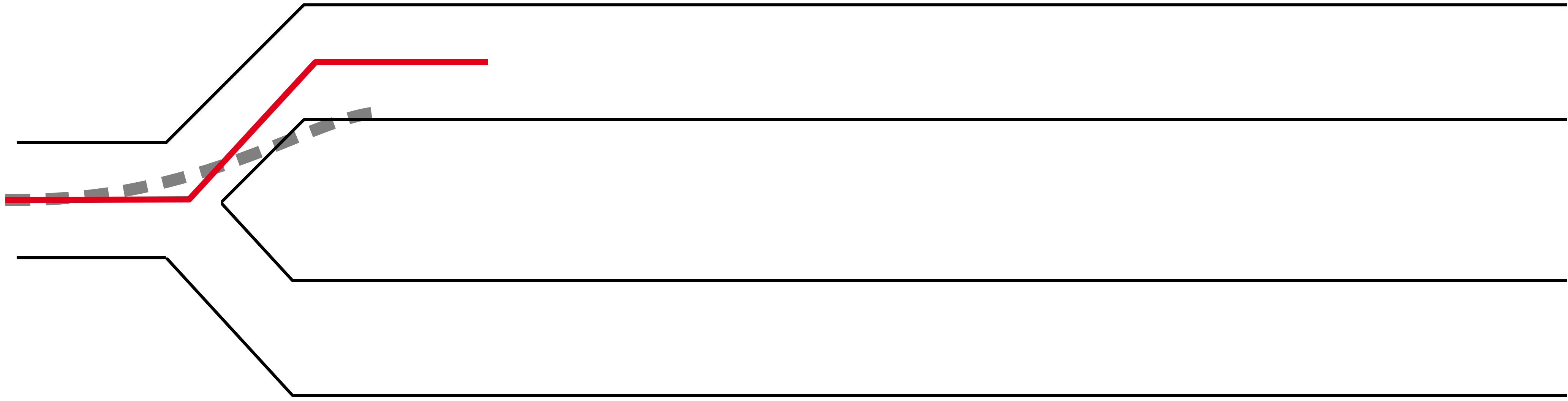
Decoding Strategies by Example

- Filtering:



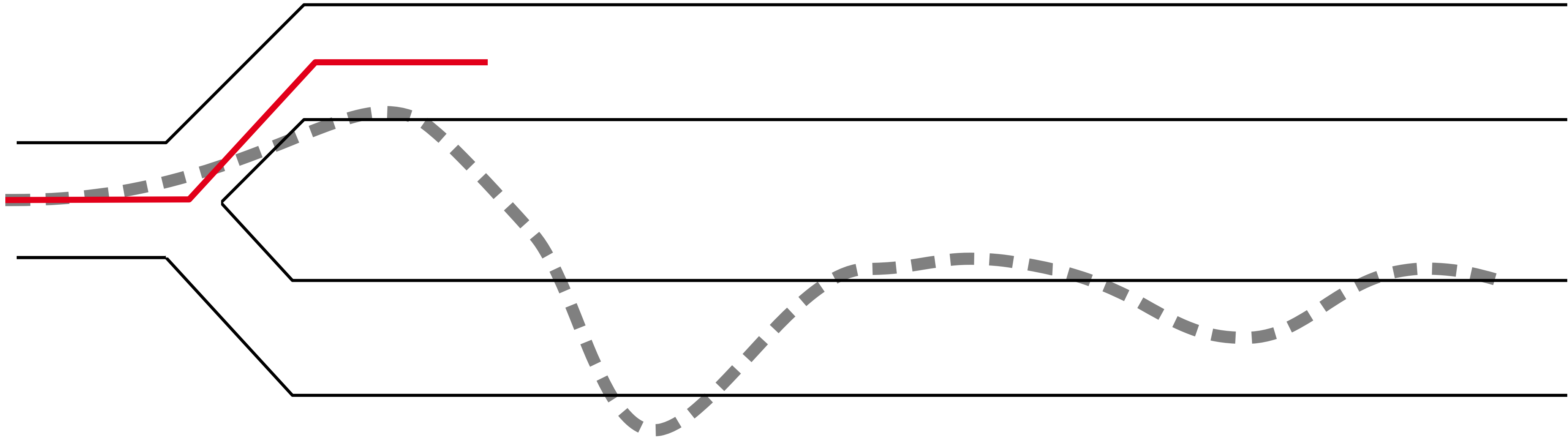
Decoding Strategies by Example

- Viterbi:



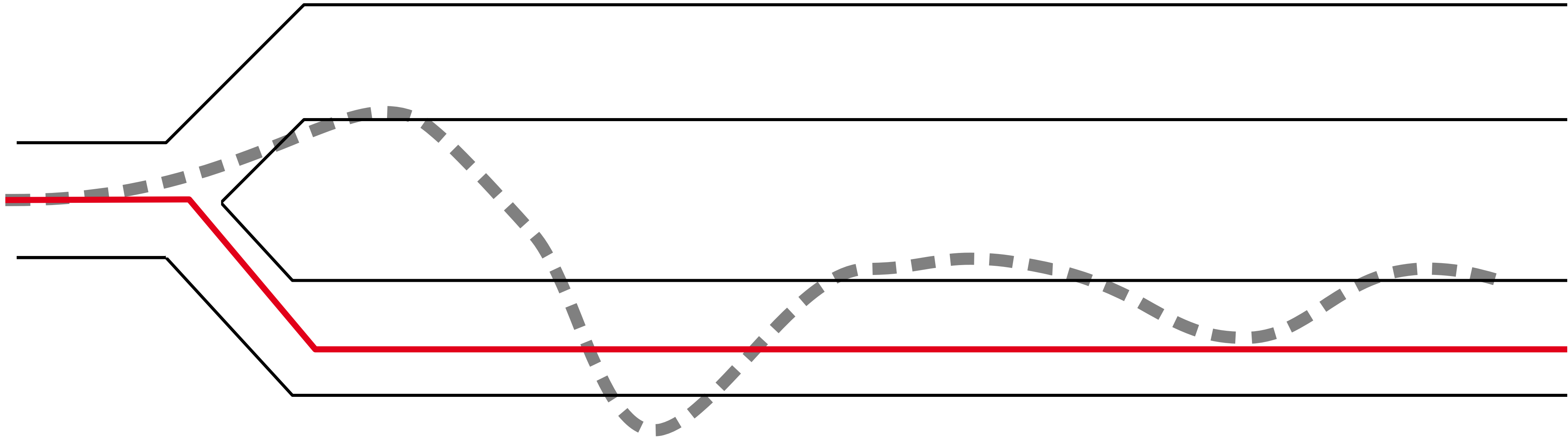
Decoding Strategies by Example

- Viterbi:



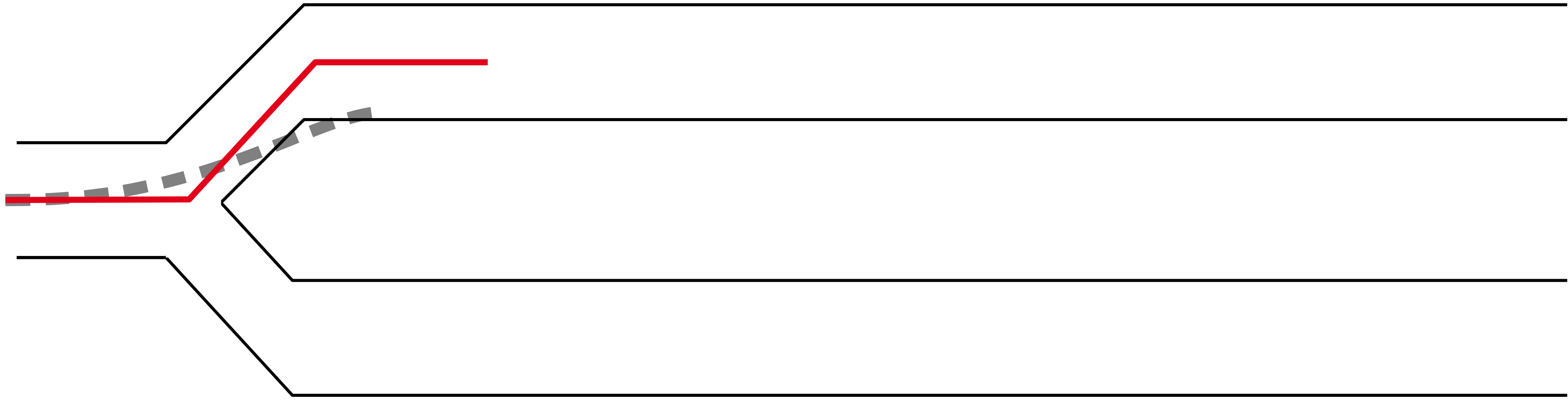
Decoding Strategies by Example

- Viterbi:



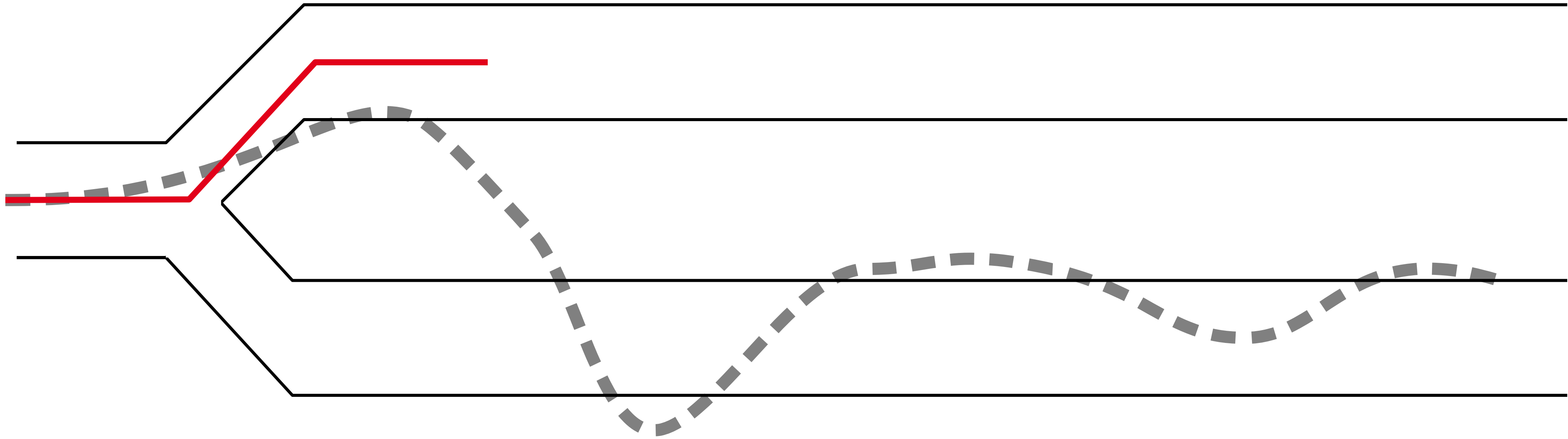
Decoding Strategies by Example

- Monotonic decoding:



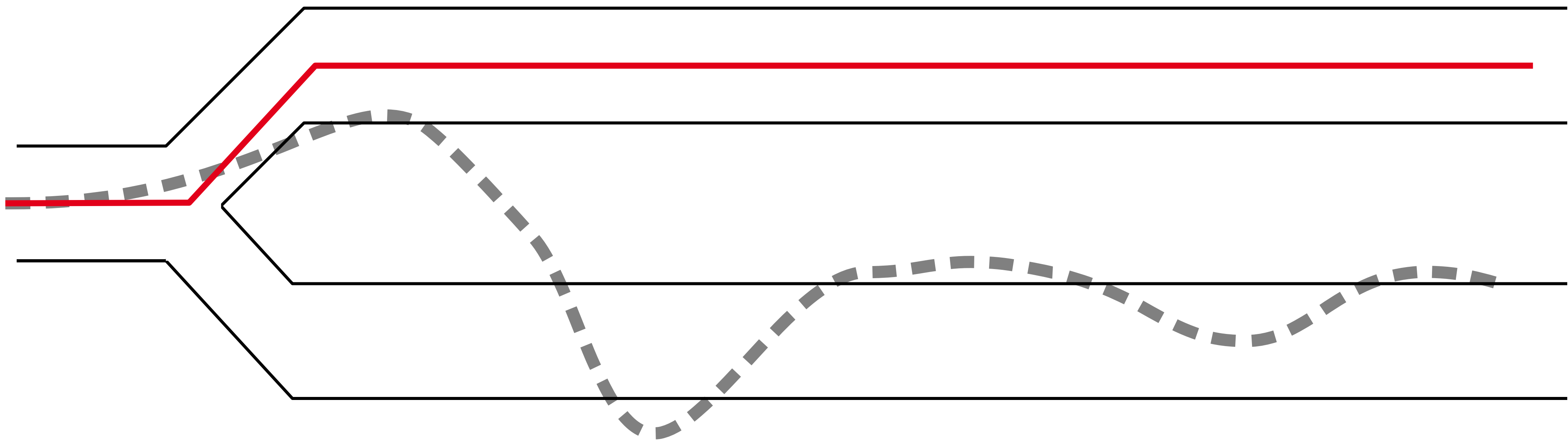
Decoding Strategies by Example

- Monotonic decoding:



Decoding Strategies by Example

- Monotonic decoding:



Guarantees on the Ouput

- smoothing: best output at any state
  but output can be erratic
- Viterbi: consistent output for state sequence
  (e.g., not two full verbs in garden-path sentence)

- what you want depends on your application
  (you might want to live with suboptimal states but get "smooth" transitions between states)
  - e.g. in speech synthesis: it's better to have smooth transitions than to have "jumps" between what would be locally optimal

Trivial Incremental Output

G2P as an example of „incrementalizing“ any simple problem

- grapheme-to-phoneme conversion is the task of turning (written) words into the corresponding (spoken) phonemes
  - G R AE F IY M T UW F OW N IY M K AH N V ER ZH AH N ...
- most tools work word-by-word in full words
- can I use such a tool to work character-by-character?
  -

c	K
ch	K
cha	K AE
char	K AE R
chara	K AE R IH?
charac	K AE R IH K
charact	K AE R IH K T
characte	K AE R IH K T
character	K AE R IH K T ER

Restart-Incremental Processing

- yes:
  - just re-run the tool with all the prefixes after one another
  - the tool need not be aware that prefixes belong to each other

- downsides: the tool's optimization criteria will mismatch the task. E.g., the end of a word may be significant to the tool, but insignificant for all the prefixes
  - possible remedy: retrain by enlarging the training data

    (this is not trivial, as you'll need to map input parts to output parts to generate reasonably enlarged training data)

results of trivial restart-incremental processing

- optimum

c	K
ch	K
cha	K AE
char	K AE R
chara	K AE R IH?
charac	K AE R IH K
charact	K AE R IH K T
characte	K AE R IH K T
character	K AE R IH K T ER

- actual

c	S IY
ch	CH
cha	CH AH
char	CH AA R
chara	CH AA R AH
charac	K EH R IH K
charact	K EH R IH K T
characte	K EH R IH K T
character	K EH R IH K T ER

- far from optimum
- but much better than non-incremental (useful results 4 characters before the end of the word)

Conclusion for today

- sequence-to-sequence problems
  - often already decoded incrementally internally
  - however global optimizations (partially) break down for incremental output
- incremental evaluation
  - early results are better
  - results that remain stable are better
  - unchanged from non-incremental: correct results are better
- fundamental timeliness/stability trade-off
  - delaying results for a little while improves stability (but hurts timeliness)
  - estimate reliability based on edit survival rates

Thank you.

{baumann,koehn}@informatik.uni-hamburg.de
get the code at inprotk.sf.net.

Desired Learning Outcomes

- sequence problems are often decoded using (Hidden) Markov models and decoding these is incremental as-is (yet, software interfaces may need to be rewritten)
- students know the fundamental timing/stability trade-off and understand that it can be controlled by time-based smoothing