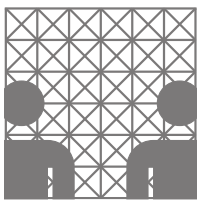
Recognizing Conversational Speech: What an Incremental ASR Should Do for a Dialogue System and How to Get There

Timo Baumann, Casey Kennington, Julian Hough, David Schlangen

baumann@informatik.uni-hamburg.de



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Universität Bielefeld

Motivation

Speech recognition is becoming a commodity – so, which to choose for a conversational SDS?

- we evaluate 3 systems (Google, Sphinx, Kaldi)
- focus on conversational speech
- focus on requirements for smooth interaction
- beyond Morbini et al. (2013):
  - evaluate incrementality of the recognizers
  - more detailed look at disfluent speech

Content

- Desiderata for conversational dialogue systems
- Mapping to evaluation criteria
- Description of the evaluated systems
- Results and discussion

Recognizing Conversational Speech

Recognizing Conversational Speech

- ASRs cater for non-conversational use-cases
  - mostly training with read speech, e.g. Voxforge and audio books for open source systems
  - performance on conversational speech varies
- today's dialogue systems are task-based
  - “Computer Talk” (Fischer 2006): fewer disfluencies, word choice, pronunciation
  - systems will be conversational/“interactional”/social rather than “transactional” in the future

➔ we focus on incrementality and disfluencies

ASR-Desiderata for Conversational Speech Recognition

- *correct*: low CER, low WER
- *quick*: realtime
- *incremental*: partial results as early as possible
- *reliable*: no/few spurious partial results
- *maximally informative*: n-best, confidence, word-timings, preservation of disfluent speech

ASR-Desiderata for Conversational Speech Recognition

- *correct*: low CER, low WER
- *quick*: realtime
- *incremental*: partial results as early as possible
- *reliable*: no/few spurious partial results
- *maximally informative*: n-best, confidence, word-timings, preservation of disfluent speech

ASR-Desiderata for Conversational Speech Recognition

- *correct*: low CER, low WER
- *quick*: realtime
- *incremental*: partial results as early as possible
- *reliable*: no/few spurious partial results
- *maximally informative*: n-best, confidence, word-timings, preservation of disfluent speech

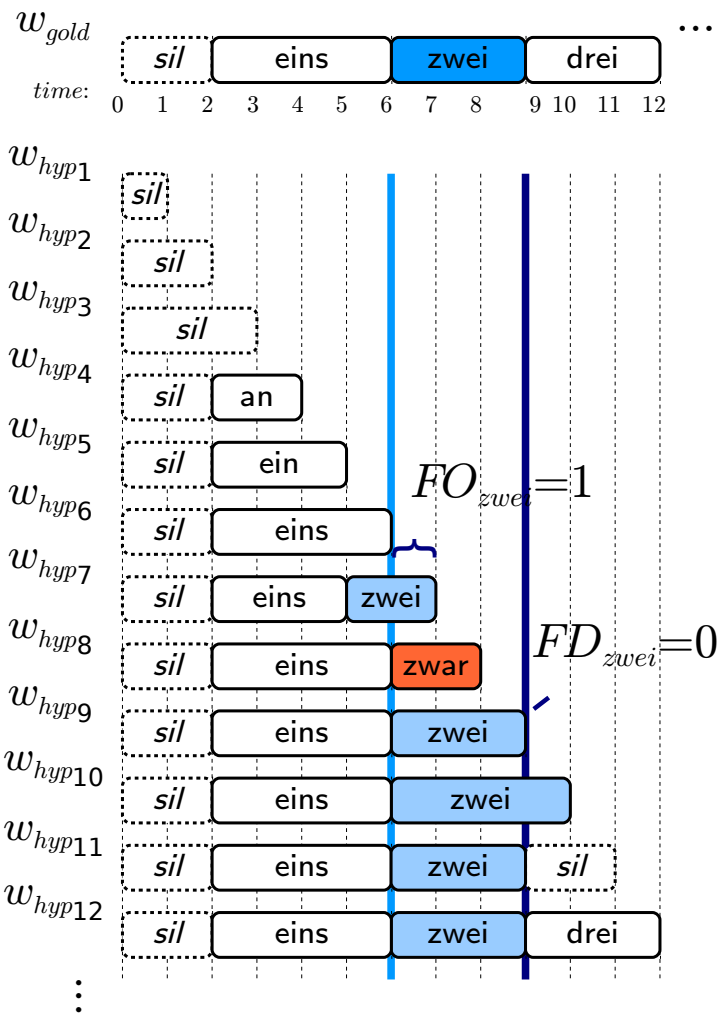
Evaluating Incremental ASR

- when do words first occur in the output (FO)
  - the ASR can decide to change this (repeatedly)
- when are words first decided (FD)
  - the ASR does not change its opinion anymore

- FO tells us when (on average) the ASR *considers* a word, FD tells us when it *decides on* a word

Baumann et al., 2009 (NAACL-HLT)

Evaluating Incremental ASR



for "zwei":

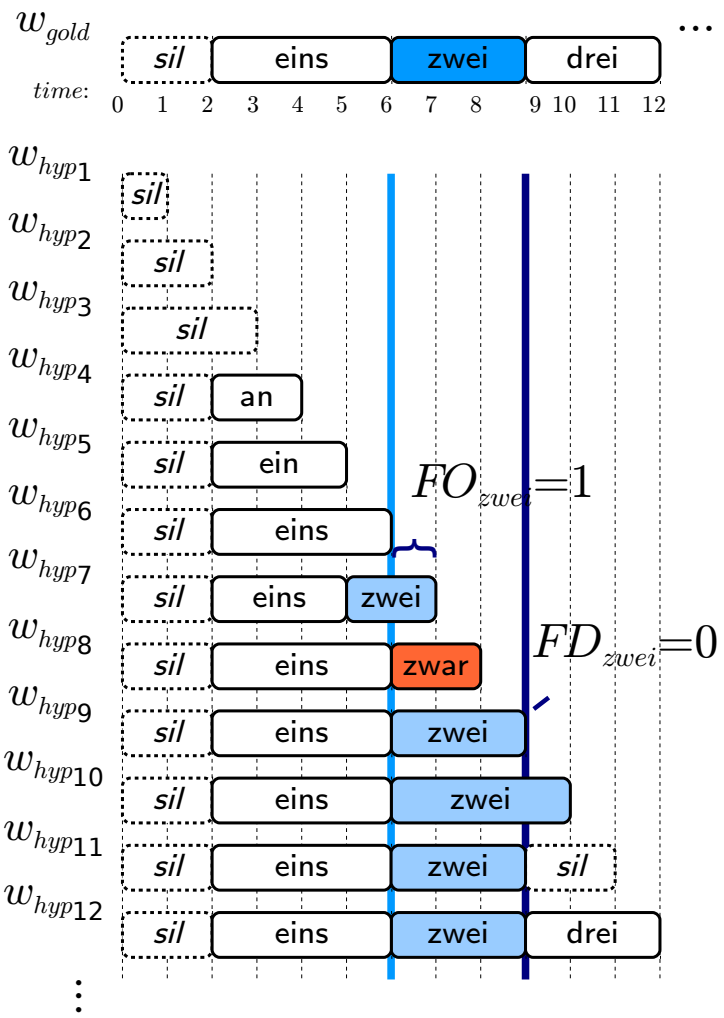
first correct at $t = 7$

first decided at $t = 9$

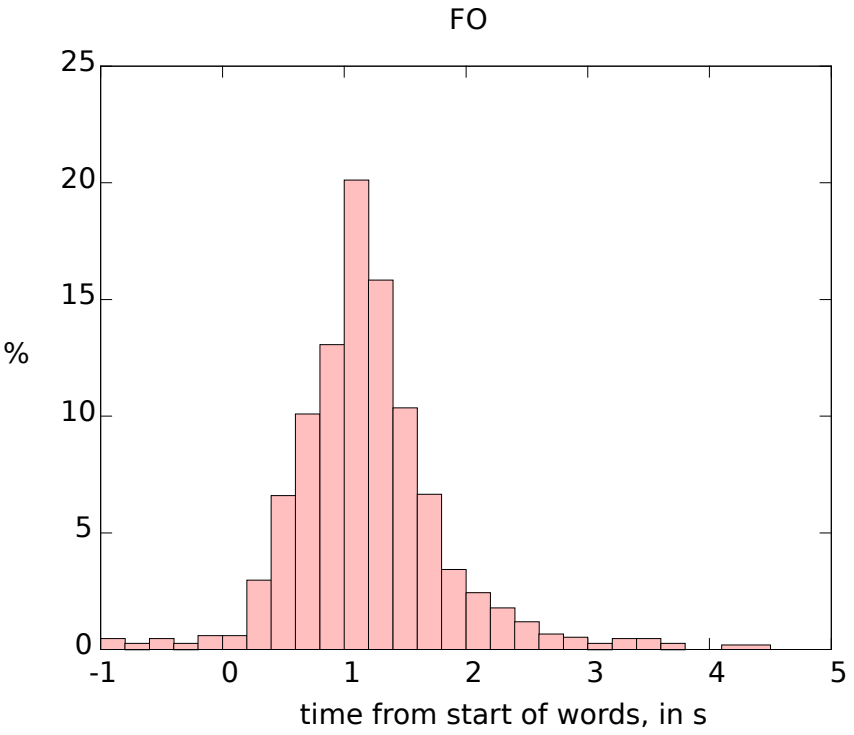
$\mathbf{FO_{zwei}} = 1$

$\mathbf{FD_{zwei}} = 0$

Evaluating Incremental ASR



compare distributions over all words to find the snappyness of the partial results



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)
- Word Survival Rate:
  - a word that is hypothesized and remains in the result „lives forever“
  - other words “die off” in favour of alternate word-hypotheses after a certain time
  - we plot the survival rate over time and use the age of a word as a reliability estimate

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

How to deal with disfluencies

How to deal with disfluencies

- simple: filter out *uh*s and *uhm*s
  - John *uh* likes Mary → John likes Mary
- better?: find and filter reparandum
  - John *likes uh* loves Mary → John loves Mary
- but not good enough:
  - *John likes uh* he loves Mary → He loves Mary (who?)
- filtering correctly requires more than any ASR could be capable of (understanding, context, ...)
  - do not filter but defer to a later module (potentially: mark-up possible disfluencies)

How to deal with disfluencies

- simple: filter out *uh*s and *uhm*s
  - John *uh* likes Mary → John likes Mary
- better?: find and filter reparandum
  - John *likes uh* loves Mary → John loves Mary
- but not good enough:
  - *John likes uh* he loves Mary → He loves Mary (who?)
- filtering correctly requires more than any ASR could be capable of (understanding, context, ...)
  - do not filter but defer to a later module (potentially: mark-up possible disfluencies)

How to deal with disfluencies

- simple: filter out *uh*s and *uhm*s
  - John *uh* likes Mary → John likes Mary
- better?: find and filter reparandum
  - John *likes uh* loves Mary → John loves Mary
- but not good enough:
  - *John likes uh* he loves Mary → He loves Mary (who?)
- filtering correctly requires more than any ASR could be capable of (understanding, context, ...)
  - do not filter but defer to a later module (potentially: mark-up possible disfluencies)

How to deal with disfluencies

- simple: filter out *uh*s and *uhm*s
  - John *uh* likes Mary → John likes Mary
- better?: find and filter reparandum
  - John *likes uh* loves Mary → John loves Mary
- but not good enough:
  - *John likes uh* he loves Mary → He loves Mary (who?)
- filtering correctly requires more than any ASR could be capable of (understanding, context, ...)
  - do not filter but defer to a later module (potentially: mark-up possible disfluencies)

Filtering Disfluencies

```
this is how I discovered, you know,
this is how I discovered  --- -----

[ the que- + the Prime Ministers', you
- --- ---- - --- ----- mr smith    ---

know, question ] and answer period
----- question   and answer period
```

Filtering Disfluencies

```
this is how I discovered, you know,
this is how I discovered  --- -----

[ the que- + the Prime Ministers', you
- --- ---- - --- ----- mr smith    ---

know, question ] and answer period
----- question   and answer period
```

- it seems like “you know” and the surroundings of the (silent) hesitation are filtered out

switchboard 4859.11; Google-ASR

Evaluating Disfluency Filtering

- annotate disfluencies
- correct the transcript for disfluency (i.e., remove reparanda and interregna)

```
orig: John likes uh loves Mary
filt: John loves Mary
```

- compare the ASR's result with the full and filtered transcripts:
  - WER increases indicate no filtering
  - WER decreases indicate proper filtering

```
goog: John loves Mary
sphx: John yikes uh loves Mary
```

Experiment 1:
Off-the-shelf ASRs in a dialog domain

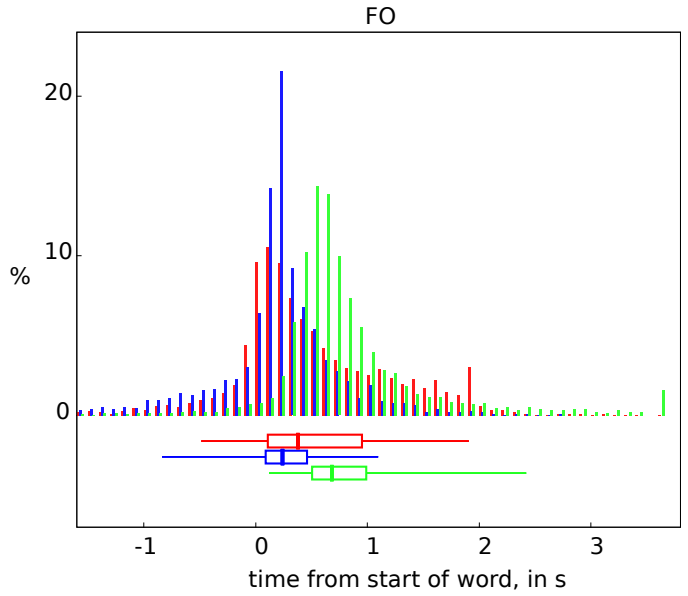
The Setup

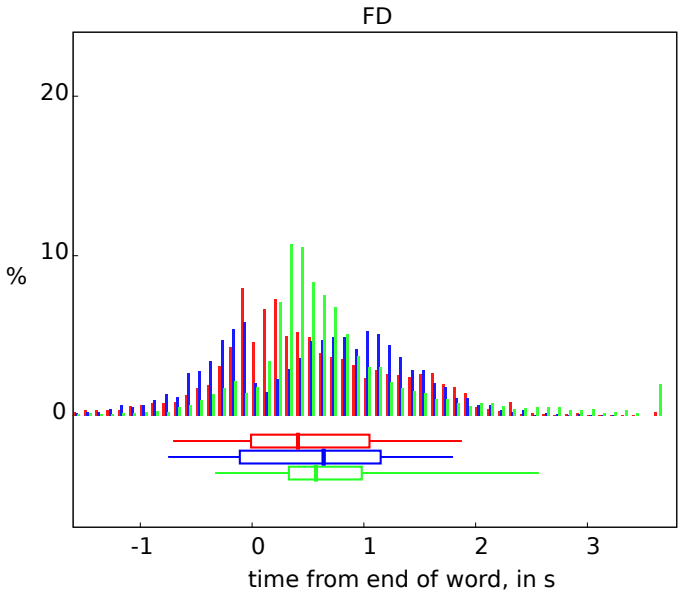
- Google Speech API
- Sphinx-4 with most recent off-the-shelf models (5.2PTM, generic English LM)
- Kaldi server trained with the Voxforge recipe (both acoustic and language models)

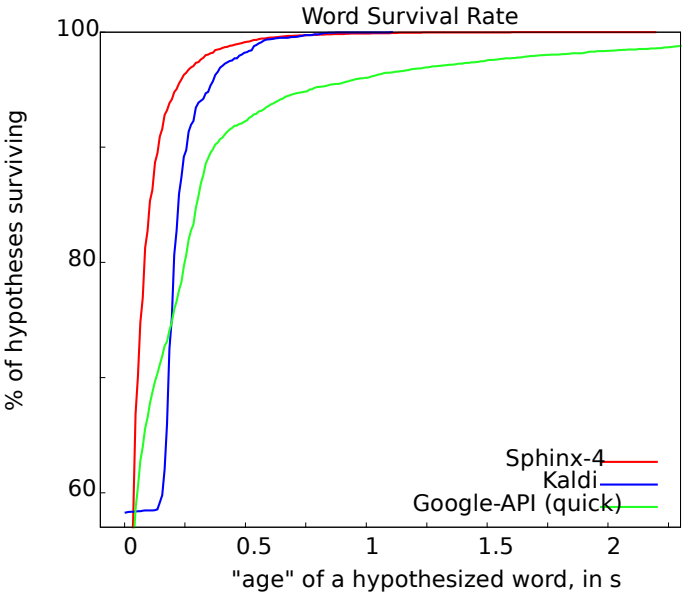
- uniformly available via InproTK

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 smoothing as can be seen in the survival rate (cmp. Baumann et al., 2009)

word error rates

System	US English speakers		All English speakers	
	WER (all)	disfluency filtered	WER (all)	disfluency filtered
Google-API-stable/quick	25.46	28.16 (+2.70)	40.62	41.60 (+0.98)
Google-API-sticky	26.08	29.29 (+3.21)	41.23	42.82 (+1.59)
Sphinx-4	57.61	62.31 (+4.70)	72.08	75.34 (+3.26)
Kaldi	71.31	73.38 (+2.07)	77.57	79.05 (+1.48)

- Google fares well – at least on US English, far worse on British English (the third author was disappointed)
- Sphinx and Kaldi are too bad to be useful (we don't know why)
- all have difficulties with disfluencies

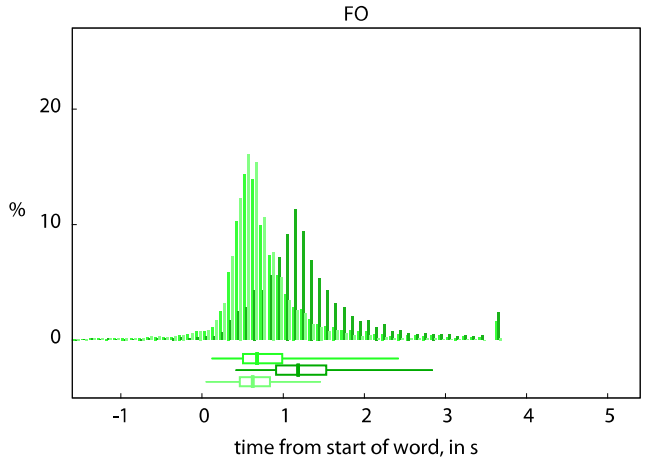
A close look at Google's results

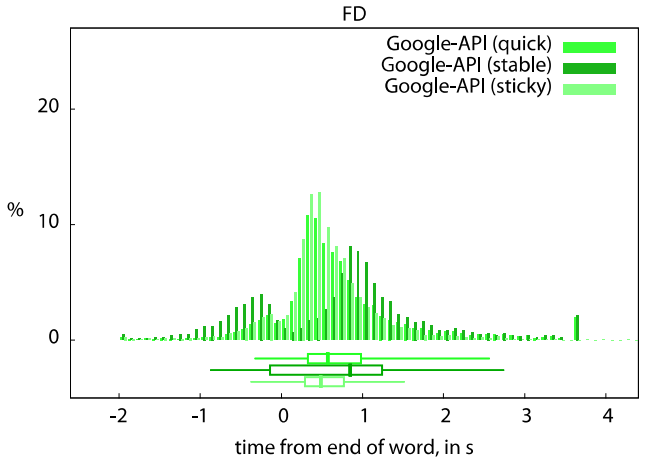
Google divides its results into a “stable” and an “unstable” part

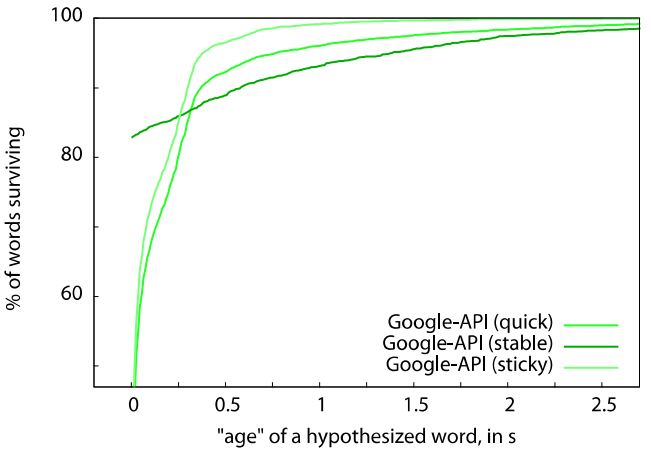
- so far we had been looking at everything

Google apparently rescores the result post-hoc

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







Experiment 2: In-domain training for conv. dialog

- Sphinx and Kaldi on ~11h German in-domain dialog (Pentomino puzzle domain)
  - much more competitive (30% WER)
  - Google for German is better than for English (20% WER) for our data
  - incremental aspects unchanged (advantage for Kaldi+Sphinx)

Conclusion

- Kaldi/Sphinx are snappier than Google
  - reasonable performance on in-domain data
  - improves with more data
- Google offers superior overall performance
  - but relatively slow
  - post-hoc rescoring has no positive effects
- Disfluencies are a problem for all recognizers
  - just errors for Kaldi/Sphinx
  - filtering of disfluent speech with Google
  - we'd prefer markup over filtering

Thank you.

supported by a Daimler-and-Benz-Foundation PostDoc grant