Dialogue and Discourse 2(1) (2011) 113-141 doi: 10.5087/dad.2011.106

Evaluation and Optimisation of Incremental Processors

Timo Baumann TIMO@LING.UNI-POTSDAM.DE
Okko Buß OKKO@LING.UNI-POTSDAM.DE
Department for Linguistics
University of Potsdam
Germany

David Schlangen DAVID.SCHLANGEN@UNI-BIELEFELD.DE
Faculty of Linguistics and Literature
Bielefeld University
Germany

Editor: Hannes Rieser

Abstract

Incremental spoken dialogue systems, which process user input as it unfolds, pose additional engineering challenges compared to more standard non-incremental systems: Their processing components must be able to accept partial, and possibly subsequently revised input, and must produce output that is at the same time as accurate as possible and delivered with as little delay as possible. In this article, we define metrics that measure how well a given processor meets these challenges, and we identify types of gold standards for evaluation. We exemplify these metrics in the evaluation of several incremental processors that we have developed. We also present generic means to optimise some of the measures, if certain trade-offs are accepted. We believe that this work will help enable principled comparison of components for incremental dialogue systems and portability of results.

Keywords: evaluation, incrementality, spoken dialogue systems, incremental processing, timing

1. Introduction

Recent work (Aist et al. 2007, Skantze and Schlangen 2009, Skantze and Hjalmarsson 2010) has shown that *incremental dialogue systems*, that is, dialogue systems which process user input while it is still ongoing, offer the potential to increase user satisfaction through more natural behaviour. Research on how to build such systems, however, is in its infancy, and there are so far no agreed upon standards for how to develop, nor how to evaluate such systems or—as dialogue systems are typically built in a modular fashion—their components. We address the latter question, of how to specify and evaluate such components, which we call *incremental processors*, in this article.

Typical processing components in dialogue systems are speech recognition, parsers (often with grammars that are more semantically than syntactically motivated), dialogue act recognition, dialogue management, language generation and text-to-speech synthesis (Allen et al. 2000, Larsson and Traum 2000). Of these, all except dialogue managers are used in, and are often most actively developed for, other computational linguistics tasks. Even in non-incremental dialogue systems, adaptation and combination of processing components developed for other tasks is not trivial. Turning such

 Submitted 1/10; Accepted 3/11; Published online 5/11

processors into *incremental processors*, i. e. enabling them to generate partial results given partial input that may come from other incremental components in the system, poses additional challenges: How can the desired behaviour with respect to the abilities of accepting partial, possibly uncertain and subsequently revised input, and of producing incrementally growing and improving output, be described, and actual behaviour be evaluated? Incrementality adds a temporal dimension to the evaluation problem: There is no longer just one final output of a processor that needs to be evaluated; potentially, there is a sequence of such outputs, whose appropriateness given the itself partial input must be judged, as well as its "stability" over time and the timeliness of its production.

In this article, we approach these challenges 'from the outside', as it were, by abstractly describing requirements for incremental processors, and deriving from this description metrics for evaluating their performance. We bring together in a unified framework our own previous work on evaluating incremental processors we have developed for our own purposes, and show how this framework can be used to compare this work to attempts by other groups. We also demonstrate how our conceptualisation of the desired outputs allows for the development of generic optimisation techniques that can improve incremental processors along certain parameters. It is our hope that this specification of an evaluation framework will be a valuable contribution to the developing field of incremental dialogue systems, by offering metrics that make work comparable and hence may help achieve the goal of building a toolbox of plug-and-play components.

The remainder of the article is structured as follows: We discuss previous work addressing the evaluation of incremental processing components and similar work in the following section. In Section 3, we explain our notion of incremental processing and some of the challenges involved in devising incremental processors. Sections 4 and 5 are the main part of this article where we discuss our evaluation methodology (in Section 4) and how it can be put to use to characterise processing components, namely several processors that we have built (in Section 5). In particular, we discuss *two different kinds of gold-standards* in Section 4.1 (those including or not including incrementality information) that can be used in the evaluation of incremental processors. We then discuss how these different kinds of gold standards can be used in evaluations that target different aspects of incremental processing: *Similarity Metrics* (Section 4.2.1), i. e. measures of equality with or similarity to the gold standard, *Timing Metrics* (Section 4.2.2), i. e. measures of the timing of relevant phenomena w. r. t. the gold standard, and *Diachronic Metrics* (Section 4.2.3), which measure the change of incremental hypotheses over time. We give a *summary of the metrics* (Section 4.4) which also contains a *tabular overview* (Table 1). Additionally, we exemplify all the metrics presented in the *evaluations* (Section 5). Using the considerations about the characteristics of incremental processing from the previous two sections, we present in Section 6 some generic ways to optimise incremental processors with respect to some of the metrics. We close with a discussion of possible future work in Section 7.

2. Previous Work

Accounts of incrementality and incremental processing have been given before (e. g. (Amtrup 1999, Guhe 2007)). However, most of that work does not systematically deal with the fact that incremental processing requires that a component be able to "change its mind", i. e. it requires the possibility for an incremental processor to revert previously output hypotheses in the light of additional input. This requirement is described by Schlangen and Skantze (2009), who introduce a model of incremental processing which supports changes and revocation of previously output hypotheses of an individual

processor. This allows a processor to output an hypothesis as soon as possible, while keeping the option of changing or revoking this hypothesis in favour of a different, better one, later on. Significant changes and extensions in evaluation methodology become necessary for incremental processing following this approach.

Many problems in spoken language processing have traditionally been tackled incrementally (e. g. Young et al. (1989)), partially to keep memory requirements low. There are even examples of early incremental speech understanding systems (Reddy et al. 1976). However, in the system cited above, incrementality was only seen as a means to improve non-incremental system performance, and no evaluation of incremental aspects took place. "Consciously incremental" processors for many different tasks have also been described in the literature (e. g. Wachsmuth et al. (1998), Sagae et al. (2009), and others mentioned below), and these descriptions usually also include evaluations. An often-used method here is to use standard metrics (such as word error rate (WER; Hunt (1990)) for speech recognizers) for comparing a non-incremental process and an incremental processor which never change their own previously output hypotheses. As this kind of incremental processing is limited in right context, its results will be worse than non-incremental processing, and the trade-off between incrementality and degraded results is assessed (e. g. Wachsmuth et al. (1998)).

A notable exception from the tradition of using standard metrics for the evaluation of incremental processors is (Kato et al. 2004) in the field of incremental parsing, which deals with the evaluation of incrementally generated partial parse trees. Kato et al. define a partial parse's validity given partial input and implement a method for withholding output hypotheses for a given length of time (similarly to our methods to be presented in Section 6), measuring the imposed average *delay* of this method as well as loss in precision compared to non-incremental processing. While this evaluation method goes some way towards capturing peculiarities of incremental processing, it still cannot account for the full flexibility of the incremental model by Schlangen and Skantze (2009), which allows for incremental processors to revise previously output hypotheses so that they may eventually produce final results that are as good as those generated by their non-incremental counterparts. Such an incremental processor may trade timeliness of incremental results for quality of final results. As we will argue here, for this kind of incremental processing evaluation metrics are needed that capture specifically the incremental aspects of the processors and the *evolution over time* of incremental results. What needs to be measured is *what happens when*, as simply comparing final results of incremental and non-incremental settings is not enough.

In our own previous work on incremental processors, we have developed metrics—out of a need to do justice to the complexity of the problem of incremental processing, which wasn't captured by standard metrics—for evaluating incremental ASR (Baumann et al. 2009a) and for balancing incremental quality and responsiveness; for evaluating incremental reference resolution (Schlangen et al. 2009), where the focus was on measuring the "stability" of hypotheses; for evaluating incremental natural language understanding more generally (Atterer et al. 2009), looking at the distribution of certain important events (correct hypothesis first found, and final decision reached); and finally for n-best processing (Baumann et al. 2009b). It is our goal in the present article to generalise these metrics; we return to this previous work in Section 5, where we revisit the results from these papers to illustrate our generalised metrics.

There are similarities between the problems posed by incremental processing and those posed by what is often called "anytime processing" (Dean and Boddy 1988). Particularly relevant from that field is Zilberstein (1996), who discusses the evaluation of anytime processing. *Anytime processing* is concerned with algorithms that have some result ready at every instant, even when all processing

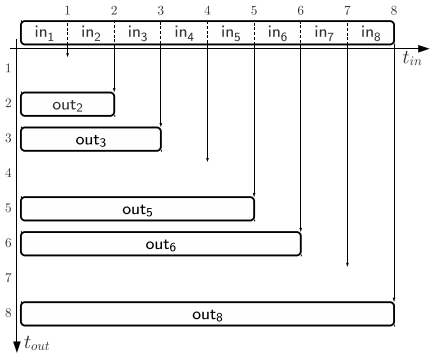

Figure 1: Schematic view of relation between input increments and growing output in incremental processing

possibilities have not yet been exhausted. As noted by Schlangen and Skantze (2009), incremental processing can be seen as a generalisation of anytime processing, where responses are required also for partial input. In anytime processing, there exists a trade-off between the processor's *deliberation time* (when to stop a heuristic search) and the quality of results. In incremental processing, the deliberation time can be seen as the amount of input that the processor awaits before returning a partial result. Likewise to us arguing that incremental evaluation can not be condensed to one single performance metric, Zilberstein (1996) notes for anytime processing that "[t]he binary notion of correctness is replaced with a multivalued quality measure associated with each answer." He also introduces *quality maps* and *performance profiles* as ways to characterise the "expected output quality with execution time *t*" for anytime processors, a notion which we will adjust to relate output quality to amount of context.

3. Our Notion of Incremental Processing

Incremental processing is concerned with the processing of minimal amounts of input (we call these *input increments*) in a piece-meal fashion as soon as they become available (Guhe 2007). Additionally to consuming input incrementally, we also see production of output while the input is still ongoing as an integral part of incremental processing (Kilger and Finkler 1995)—without this condition, any incrementality a processor might possess internally would be unobservable to the outside world.

In this article, we call a processor that consumes input incrementally and generates output incrementally an *incremental processor*. We are solely concerned with the evaluation of individual incremental processors and do not consider the combination of processors to systems. As a consequence, we do not have to consider the inter-play of loops when processors feed back information to predecessors. In the sense of Wirén (1992, as cited by Guhe (2007)), our processing is ordered *left-to-right*. We will detail the input and output of an incremental processor and data structures suitable for this purpose in the following subsections.

3.1 Incremental Processors

A schematic example of input and output of an incremental processor is given in Figure 1. The input, consisting of a sequence of eight input increments, is shown in the top row and the output after the processing of each input increment is shown in the subsequent rows. The time spanned by the input increments is shown on the horizontal axis, while the times at which the outputs have been produced are indicated on the vertical axis. We here assume that the output at some instant t is based on all input up to t and represent this correspondence also through the width (and alignment) of the bars representing output in the figure. As an example, output out_5 is the output generated at time t_5, after all input increments up to and including in_5 have been consumed. It is incidental in the figure that input increment times are equidistant: in many cases, there will be irregular intervals between increments.

In the discussions in this article, we ignore processing delays imposed by the actual consumption of input and generation of output by the processor. (In reality, t_5^{out}, the time at which output out_5 has been created would be slightly after t_5^{in}, the time at which input in_5 was consumed.) In the terminology of Guhe (2007), we assume processing times to be *bounded*, i. e. constant per input increment, and assume that it is negligible compared to the time spanned by each increment.

Notice that not always is new output being generated after consuming an input increment (there is no new output after consuming in_1, in_4, and in_7). This is called *moderate* (as opposed to *massive*) incrementality by Hildebrandt et al. (1999). For anytime processing, Zilberstein (1996) calls the property of having a result ready at all times *interruptibility* and notes that any non-interruptible processor can be made interruptible by caching the most recent result. This holds analogously for incremental processing.

The figure illustrates that there are two temporal dimensions that are relevant when talking about incremental processing: One is the time when the output was made, the other is which segment of the input that output is based on. When evaluating, accuracy of the output should only be determined relative to the input available, and timeliness of the output should be determined relative to the relevant input.

3.2 Representing Incremental Data

In Figure 1 we deliberately left out the question of how output should be structured. Finkler (1997) introduced the distinction between *quantitative* and *qualitative* incrementality. In the former, all output is repeated after every input increment and hence the question of output structure is irrelevant for evaluation methodology. At the same time, quantitative incrementality does not facilitate building modular incremental systems, as output from one component cannot easily be fed to a subsequent component in a piece-meal fashion, as the link between individual bits of information is lost. Contrastingly, in qualitative incrementality output increments build on the output of the preceding processing step and the relation between successive outputs can be traced: Does a later output simply extend an earlier one, or does it (partially) contradict it? Or is there no difference and hence no update by successive processors is necessary? For this we need to structure the output of an incremental processor into *output increments*. Furthermore, we may be interested in tracing which parts of the input have lead to a certain output increment in order to make more informed decisions in a later module.

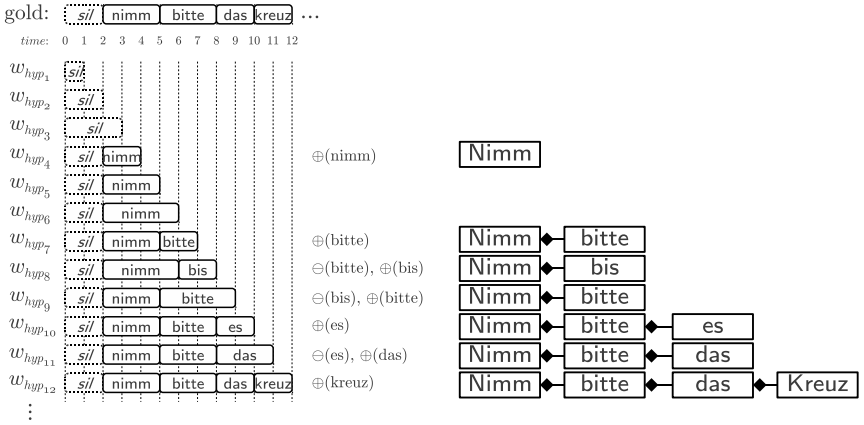

Figure 2: ASR hypotheses during incremental recognition. Raw ASR hypotheses are shown on the left, the corresponding IU network on the right, and step-wise edits in the center.

We use the notion of *incremental units* (IUs), introduced in (Schlangen and Skantze 2009)[1] to arrive at a parsimonious representation for both input and output. IUs are the smallest 'chunks' of information that can trigger a processor into action. IUs typically are part of larger units, for example as individual words are parts of an utterance. This relation of being part of the same larger unit is recorded through *same level links*; the information that was used in creating a given IU is linked to it via *grounded in links*.[2] As IUs are connected with these links, an *IU network* emerges that represents the current output of the processor at the given instant. During incremental processing, the processor incrementally builds up the network of IUs that will eventually form its final output, once all input has been consumed.

Figure 2 shows (stylised) output of an incremental speech recognition (ASR) component. In the figure, ASR is recognising the utterance "Nimm bitte das Kreuz [Take please the cross]" with intermittent misrecognitions substituting "bis" ("until") for "bitte" ("please") and "es" ("it") for "das" ("the"). The representation on the left of the figure is similar to that of Figure 1 above, but here the actual content and structure of the output hypotheses is shown. The rightmost column shows the IU network as it emerges after each processing step. Each IU contains a word hypothesis and IUs are connected via same level links, eventually forming an utterance. The middle column lists the changes that occur between consecutive states of the output IU network.

The *current hypothesis* (or set of hypotheses), i. e. the current 'total output' of an incremental processor during incremental processing can be read off the output IU network by following the same level links from the newest IU(s) backwards. Figure 3 shows IU networks for ASRs in different configurations; for one-best output, there is one final node, for n-best lists there are n final nodes

1. For an extended version see also (Schlangen and Skantze 2011) in this volume.
2. The sequential relationship expressed by same level links is represented implicitly in the input sequence in Figure 1 simply by placing the increments in sequence; Figures 2 (right side), 3 and later figures make them explicit with diamond-headed arrows between IUs. Informational dependency between input and output is represented in Figures 1 and 2 (left side) through horizontal alignment of inputs and outputs while the input has been left out in Figure 3 and no dependencies are shown. In later figures, grounded-in links for informational dependency will be expressed by arrows with regular heads.

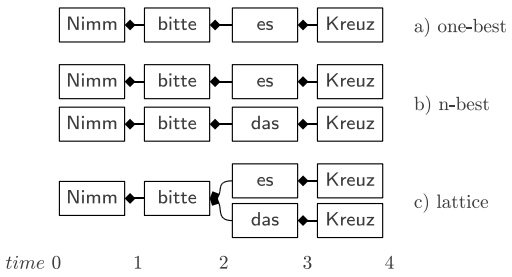

Figure 3: One-best (a), n-best (b), or lattice (c) IU output of an ASR in the IU framework.

with unconnected networks and for lattice-/tree-like structures paths are connected in a common root, ending possibly in several final nodes. The network representation enables a parsimonious representation of the changes to the current hypothesis from one time step to the next (i. e., the real output increments): extension of a previous output is represented as addition of linked IUs to a network, while (partial) contradiction is represented as the un-linking of IUs from a network and the linking of other IUs in their place. Finally, when looking at the evolution of the IU network during incremental processing, we are able to notice when certain IUs are added or removed.

Using this representation, both the current full or 'quantitative' outputs as well as the changes between consecutive outputs can be traced. This representation along with how it changes over time allows us to define metrics that describe (a) the quality of the results encoded in the network, (b) when the individual contributions that form this result were created by the processor, and (c) what else has been hypothesized by the processor along the way. We will describe how to evaluate these three aspects of incremental processing in the following section, after briefly discussing the types of gold standard information that are required.

4. Evaluating Incremental Processors

In the sense that will concern us in this article, evaluation is the comparison of a processor's actual output to some ideal output in order to assess the processor's performance.[3] Such ideal output (often called *gold standard*) can be manually constructed or automatically generated using additional knowledge that is not available to the processor that is being evaluated. The ideal output of a process will look differently, depending on the task, and the specific goals set for a processor. Take as an example the disfluent utterance "Pick up the fork and the knife, I mean, the spoon." Even though "knife" is shown to be uttered by mistake with the interregnum "I mean" and later corrected by the reparans "spoon", it should form part of the output of an incremental speech recognizer as these words have undeniably been spoken. Similarly, a simple semantic analyzer should probably generate output for both "knife" and "I mean", though the latter should be marked as being an interregnum. However, a pragmatic interpreter should *not* output `take(knife)`, not even intermittently, as this meaning is not what the speaker intends. Such an interpreter's prudence will come at the cost of timeliness, because output can only be generated when it is reasonably certain. These considerations should be taken into account when designing the gold standard for a specific application.

3. Another paradigm for the evaluation of dialogue systems is to *subjectively* evaluate the end-to-end performance of whole systems interacting with users. We will not deal with that methodology in this article, but see Section 7 for some remarks.

Comparing ideal to actual output is easy if one is only interested in exact correspondence of the two: Either they are equal, or they are not. Evaluation is much more difficult if some kind of mismatch between actual and ideal output is allowed and different kinds of mismatches should be rated differently, that is, if what is being measured is the *similarity* between actual output and gold standard. What counts as similarity between two different possible outputs is highly task-dependent and many different similarity metrics have been developed for different tasks. (To pick two examples, see (NIST Website 2003) for metrics for evaluating utterance segmentation, and (Papineni et al. 2002) for a metric for evaluating the output of machine translators.)

Incremental evaluation concerns the evaluation of incremental processors, that is, the comparison of actual outputs of such processors to ideal outputs. As explained in the previous section, an incremental processor produces a sequence of partial outputs (one per input increment). Hence, there is more to do than just comparing one actual output to one ideal output: We want to compare the content of incremental outputs, the timing of contributions to the result, and the evolution of the output over the course of processing the input sequence. Before defining metrics for these three aspects, we will first have to turn to the targets for comparison: the gold standard.

4.1 Two Types of Gold Standards for Evaluation

We have said above that evaluation (of the kind we are aiming at in this article) is the comparison of *actual* output of a given processor to output that we would ideally like to see. In the previous section, we have shown how we can represent output of an incremental processor in a way that makes all dimensions of incremental information (content, timing, and evolution of incremental results) easily accessible. The question now is how we can get ideal output data that is also in this format. Sometimes, this is easy to achieve; in other cases, some information used in this representation cannot be recovered from typical evaluation resources, and an approximation has to be found. We deal with the former case first, and then with the approximation case.

4.1.1 EVALUATION WITH INCREMENTAL GOLD STANDARDS

Figure 2 from the previous section showed the state of an incremental processor's output IU network after each input increment consumed. This is the format in which we would like a gold standard for evaluation to be. Luckily, the information required for this is often available in existing ASR evaluation resources: For the content, we need the sequence of output IUs, which in this case is a sequence of words which is provided by the transliteration of the input (either done manually, or automatically with an ASR). We also need the link between input increments and output increments which in this case means that we need an alignment between words and audio signal. Again, this is often provided by ASR evaluation resources, and if not, can be produced automatically via forced alignment.

In Figure 2 (left side), an aligned gold standard sequence is shown in the top row (labelled "gold"). We can see this as the final state which our incremental processor should ideally reach; but what about the intermediate stages? These can be created from the final state by going backwards through the input IUs and removing the current rightmost output IU whenever we go past the input IU that marks its beginning (e. g. at time 10 we would remove "Kreuz", at time 8 "das", and so on.) Following this method, the resulting gold standard demands that an output increment be created as soon as the first corresponding input increment has been consumed; e.g., a word-IU should be produced by an ASR as soon as the first audio frame that is part of the word in the gold standard is

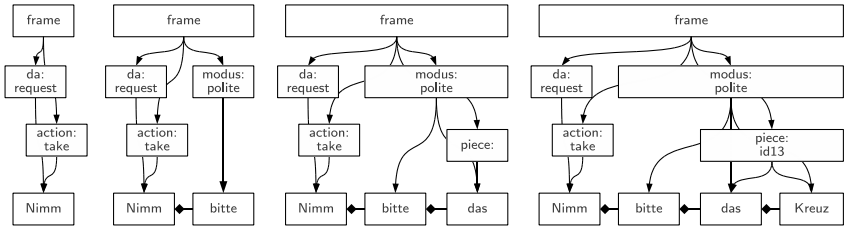

Figure 4: Four subsequent outputs for an incremental semantics component as input words are being processed.

received. While this will often be impossible to achieve, it provides us with a well-defined upper boundary of the performance that can be expected from an incremental processor. We call the resulting intermediate stages the *current gold standard* relative to a given input increment.

This method is directly transferable to other kinds of input and output. Figure 4 shows the incremental growth of a network representing a frame-semantics[4]; this time the input increments (in this case words, not bits of audio) are shown in the bottom row, and the grounded-in links which relate output to input are represented by arrows. (As the networks in this example are more complex, steps are drawn next to each other and not in rows as in the previous figures.) If we have available a corpus of utterances annotated with their final semantics together with information about which words are responsible for which bits of that final semantics, we can use the same method to go backwards through the input IUs and create the full corresponding set of IU states for partial inputs. However, such resources are rare, as making the link between what should be known based on partial input may not even be easy for human annotators (but see (Gallo et al. 2007) for an effort to create such a resource). Typically, only the final correct semantics is available, with no indication of how to create it (see e. g. the ATIS corpus as used in (He and Young 2005)). In such a case, the intermediate outputs must be approximated from the final state; we will explain how in the next section.

4.1.2 EVALUATION WITH NON-INCREMENTAL GOLD STANDARDS

Figure 5 shows a situation in which a fine-grained link between input and output increments cannot be recovered from the available evaluation resource. We then simply assume that all output IUs are grounded in all input IUs, which is the equivalent of saying that every input increment contributed to every output increment. The figure only shows the final state, we again derive the incremental steps from this by going backwards through the input IUs, as above. However, no desired output will disappear from the gold standard because every output is already grounded in the very first input increment (as we don't know what input increment some output increment *logically* depends on). Viewed in the direction of time this means that the gold standard is demanding that all output increments be known from the beginning; this is clearly an unreasonable assumption, but as it is kept constant, it allows to measure the gradual approach towards this ideal.

Such a representation then of course gives us less information, and hence an evaluation based on it can only give a coarse-grained insight into the processor's performance. If we assume that in reality not all output information is available immediately, the best a processor can do against such

4. In the example given, the slot filling for "modus" depends on "bitte" and all following words. This is because the modus could easily turn out to be e. g. "sarcastic" if some other word had been added later on.

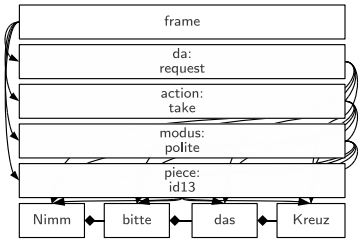

Figure 5: A semantic frame represented in the IU framework without specific dependencies between slots and words.

a gold standard is that it fares better and better as more input comes in, and as more and more of what will be the final representation is recovered. Likewise, we loose the ability to make fine-grained statements about the timeliness of each output increment.

There is a third common case that can be subsumed under this one. Sometimes one may want to build a processor that is only incremental on the input side, producing for each input increment an output of the same type as it would for a complete, non-incremental input. An example for this would be a processor that predicts a 'complete' utterance meaning based on utterance prefixes. (This has recently been explored by Sagae et al. (2009), Schlangen et al. (2009) and Heintze et al. (2010).) In IU-terms, the output IU is grounded in all input IUs and hence such a processor can be evaluated against non-incremental gold standards without loss of information.

4.2 Metrics for Evaluation of Incremental Processors

We now discuss metrics that quantify differences between actual and ideal output (i. e. the gold standard).[5] We identify three categories of metrics: Overall *similarity metrics* (measures of equality with or similarity to a gold standard), *timing metrics* (measures of the timing of relevant phenomena w. r. t. the gold standard) and *diachronic metrics* (measuring change of the incremental hypotheses over time), which we will look at in turn. These metrics illuminate the different aspects of incremental performance of a processor, but they are not completely independent of each other (e. g. timing can only be measured if something is correct, absolute correctness entails perfect timing and evolution, etc.). Interrelations of metrics will be further discussed in Section 4.3.

4.2.1 SIMILARITY METRICS

Similarity metrics compare what should ideally be known at some point in time to what *is* known at that point. The only difference that incremental evaluation brings with it is that the comparison is not done only once, for the final output given complete input, but also for all stages that lead to this final output. An incremental similarity evaluation hence will result in a sequence of results per full input token (e. g. per utterance), where non-incremental similarity evaluation yields only one. To be able

5. When describing our metrics in the following subsections, we will not give fully formalised definitions. First, we believe that the chosen level of abstraction communicates our ideas in a flexible, yet precise manner that allows reproduction and transfer to different situations. Secondly, a full formalisation of our metrics would first require the development of a formalism that would need to be capable of expressing subtle details about incremental processing (including revocation of previously output hypotheses, concurrency of processing components and possible delays in message passing). This would be, and hopefully will be, the topic of another paper.

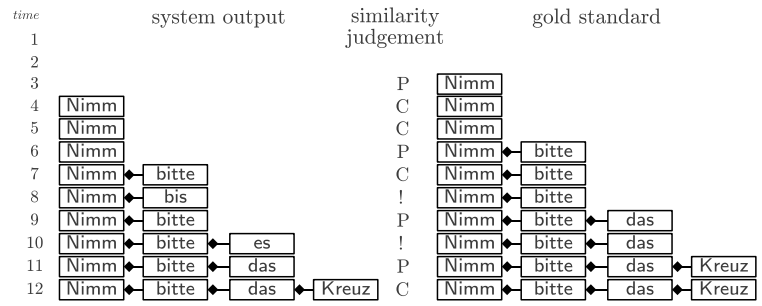

Figure 6: An ASR's actual incremental output (left), the incremental gold standard (right) and a basic similarity judgement (center): correct (C), prefix-correct (P), or incorrect (!).

to evaluate after every input increment, we need a gold standard that covers the ideal outputs after every input increment, as explained in the previous subsection. Figure 6 shows such an incremental gold standard and the IU network (for the same utterance as in Figure 2) produced by an incremental ASR.

The most basic measure of similarity is `correctness`: We simply count how often the output IU network is identical to the current gold standard and divide this by the total number of increments. In Figure 6, the output is correct four times, resulting in a `correctness` of 40 % (ignoring the empty, trivially correct hypotheses 1 and 2 in the calculation). Incremental processors often lag behind in producing output for recent input. If this delay (Δ) is known in advance, we can take it into account, defining a delay-`discounted correctness` which, if measured at time t only expects a correctness relative to the gold standard at time $t - \Delta$. However, the processor's lag will often vary with the input, which a fixed Δ cannot account for. In this case, we propose counting the number of times that the processor's output is a *prefix* of the ideal output at that instant and call the corresponding metric `p-correctness`. In Figure 6, `p-correctness` is 80 %.

`Correctness` has a shortcoming, however, namely that many processors generate output that is often not correct: Even the final output—that is, the output when all input increments have been considered—may contain errors compared to the gold standard. (Imagine that the ASR in the example from Figure 6 had generated "nehme" instead of "nimm"; this would have rendered all output increments incorrect in this strict understanding.) In such a case, a processor's non-incremental deficiencies (i. e. deficiencies that also show in non-incremental processing) block the incremental evaluation. Secondly, comparing `correctness` relative to an independent gold standard conflates the overall accuracy of the processor with the quality aspects of incremental processing. There are two possible remedies for the problem: Relaxing the equality condition to some task-dependent similarity measure (that is, changing what counts as correct), or changing the gold standard in such a way that it is guaranteed that final results are regarded as correct. We discuss the latter approach first as it allows for a clean separation between incremental and non-incremental performance.

If we want to ensure that the final output of a processor counts as correct, we can simply use this final output as the gold standard and derive from it the set of current gold standards. To distinguish this from `correctness` as compared to an independent gold standard, we call the measure `r-correctness` (for *relatively* correct, relative to the processor's final output). This separates the evaluation of incremental and non-incremental quality aspects: To learn about the overall (non-incremental)

quality of the processor, we use standard metrics to compare its output given a complete input with a "true" (externally generated) gold standard; to focus on the incremental quality, we use `r-correctness`. As a general rule of thumb, incremental performance will improve with improvements to non-incremental performance.

The alternative approach of relaxing the equality condition leads us to using task dependent evaluation techniques that make it possible to measure the similarity (and not just identity) of IU networks. Which non-incremental measure should be used as the basis for such incremental performance measure depends on the task at hand: For example, incremental speech recognition can be evaluated with WER or CER (Boros et al. 1996), syntactic parsing with measures for tree comparison (Carroll et al. 1998), semantics by calculating f-measure between frame pairs, or, as a more complex example, specific metrics for natural language generation (Reiter and Belz 2009). We can 'incrementalize' such metrics simply by making comparisons at each input time step, comparing actual output and current gold standard, as explained above. Typically, we will be interested in the average of this metric (how similar in general is the output, no matter how partial the input?), but we may also want to explore results at different grades of completeness (does the processor perform worse on smaller prefixes than on longer ones?), or the performance development over time.

Also, we may want to allow for certain output differences over time, something that is unique to incremental processing. For example, an ASR may produce in sequence the hypotheses "grün", "grüne", and finally "grünes" as it successively consumes more input. In certain settings, already the first hypothesis may be close enough so that the consuming processor can start to work, and the subsequent hypotheses would not count as contradictions. In such a case, we can set up the similarity metric so that it would allow all variants in the comparison with the gold standard, creating a kind of *incremental concept error* metric which weighs "sensible" mistakes as less serious than non-sensible mistakes.

We close with a discussion of two more similarity metrics. `F-score`, the harmonic mean of precision and recall, is a useful metric for similarity when the output representation is a 'bag' of concepts and no timing information is available in the gold standard. In such a setting, we can expect the score to be very low at the beginning (hardly any slot, compared to the correct representation for the complete input, will have been filled in the beginning) and to rise over time, as more slots are being filled (hopefully correctly). In this case, the shape of `f-score curves` plotted over different grades of input completeness can become an informative derived measure.

Finally, we can value mistakes differently at different stages through the input. This is especially appropriate for processors that predict complete outputs (see discussion above at the end of Section 4.1.2). Defining a `time-adjusted error`, we can value certain events (e. g. a processor deciding on the special class "undecided") differently, depending on how much of the input has been seen: Indecision is understandable at the beginning of input, but the more one has seen, the more not making a decision becomes just like making a wrong decision.[6]

4.2.2 TIMING METRICS

Timing metrics measure *when* some notable event happens in incremental output relative to some reference time from the gold standard. All other things being equal, we prefer incremental processors

6. There is a moral for life here.

that give good and reliable results as early as possible; our timing metrics allow us to give a precise meaning to this.

As mentioned above, a characteristic of incremental processing is that hypotheses may have to be revised in the light of subsequent input. This revision means that there are two events specifically that are informative about the performance of the processor: When information becomes available, and when it is not revised any more. We capture this with the following two metrics:

- timing of the first occurrence of an output increment (FO),
- timing of the final decision for an output increment (FD).

The statistics of these events over a test corpus convey useful information: The average FO tells us when (on average) consuming components can usefully start to work on their input, the average FD tells us when a decision becomes stable and can be relied upon. In principle, we might be specifically interested in these metrics for certain tokens in our corpus, or would like to take a close look at the distributions of FO and FD in the corpus. However, we restrict the analyses in Section 5 to reporting means, medians and standard deviations to describe the distributions.

To determine timing measures, we use the time difference between the occurrence of the IU in the output relative to the gold standard IU's timing. There are two issues with this that have to be resolved: One is that the gold standard by definition only contains timing information for correct output, and hence, we can only compute these timing measures for increments that we find in the gold standard; the other is that for those correct outputs, we need to decide on how to anchor the timing comparison.

For the problem of (possibly) missing references in the gold standard for incorrect increments we can use an automatic alignment between output and gold standard to find references for non-matching increments. Alternatively, the final (possibly incorrect) output of the processor may be used to derive the incremental gold standard (as we did for similarity metrics). Again, this separates the question of how well the processor performs compared to an external standard from the question of how well it performs incrementally; here, the measures will tell us how fast and how stable the processor is in making the decisions it will ultimately make anyway.

Regarding the anchoring of the metrics, we measure FO from the beginning of the gold IU; this again encourages *speculation* to happen as soon as some of the cues relevant for the IU become available. The 'faster' a processor is, the lower its average FO will be—some processors may even achieve a negative average FO, if they often predict output before any (direct) evidence for it was available. Conversely, for FD (the time until an IU is not withdrawn anymore) it makes sense to anchor in the end of the gold IU, which is the moment when all information regarding this IU has been seen. The more reliable a processor is, the lower its average FD will be. (FD can only be measured after processing has finished; during processing it is unknown whether an IU will be withdrawn later on.)

There will be detailed examples of the application of these metrics in the following section, but as a first illustration, for "bitte" in Figure 6 above, the first occurrence is at step 7, while it should have appeared at time 6, hence FO(bitte) is 1. As "bitte" is temporarily replaced by "bis", the final decision is at step 9. Looking back at the time-alignment in Figure 2 we see that the alignment of "bitte" actually changes between step 8 and 9 (and is only correct at step 9). Depending on whether we include this difference in our evaluation FD(bitte) is 1 or 2.

Whether timing metrics should be measured on an absolute or relative scale will depend on the task at hand and the granularity of the available gold standard. For example, we evaluate timing of ASR output in milliseconds, but timing of reference resolution as utterance-percentages in Section 5.

4.2.3 DIACHRONIC METRICS

In some sense, the types of metrics presented so far cast a static look on the processing results, with the similarity metrics telling us *whether* a result (at a given time step) is correct or not, and the timing measures telling us *when* correct results become available, given the whole set of outputs. The metrics we discuss now round out the set of metrics by telling us what happens over the course of processing the input—the *diachronic evolution* of the final results.

We measure this diachronically in the processor's incremental output by counting the *edits* to the IU network that are made over time, and in particular, counting the proportion of unnecessary edits. As discussed above, from one processing step to the next, there may be additions, revocations, and substitutions of IUs in the network. (This was illustrated in the middle column in Figure 2 above.) To build a final result comprised of n increments, it takes at least n addition operations. If an incremental processor ever 'changes its mind' about previous output, it will need more edit operations to revoke or substitute a previously output increment. We call the proportion of unnecessary or even harmful edits among the overall edits the processor's *edit overhead* (`EO`).[7] (In our running example from Figure 2 above, there are 10 edit operations for a total of 4 words, resulting in an `EO` of 60 %.)

Why does edit overhead matter, and why do we need another metric to cover this? Remember that in incremental systems, the idea is that subsequent components start to work immediately with incremental outputs of their predecessors. Hence, unnecessary edits mean unnecessary work for all consumers (and, possibly, for their consumers). A processor that frequently changes previously output hypotheses (we call such changes *jitter*) may still go unpunished in similarity metrics (as it may produce the same amount of correct intermediate results, but in a "harmful" order) and while there are influences on timing metrics (see next subsection), edit overhead allows for a direct quantification.

Another kind of diachronic metric can be derived from timing metrics: We call the (average) difference between `FO` and `FD`, i. e. the time it takes for an increment to be first hypothesized and to have settled, `correction time`. We can base an external confidence measure on this statistic, combined with the 'age' of an increment (the time that it has survived without being recalled): For example, if a processor is known to have a correction time of less than 500 ms for, say, 90 % of its output IUs, then we can be certain to a degree of 90 % that some IU will not change any more once it has been around for 500 ms. While this does not tell us whether an IU is final or not, it helps us to make probabilistic judgements.

4.3 Interrelations between Metrics

In general, one wants a processor to perform as good as possible in all the metrics presented above. In practice, this may often be impossible. To begin with, there are some simple interrelations: Perfect `correctness` is equivalent to perfect `FO` and `FD`. Both entail zero `edit overhead`, however the reverse does not hold as the 'good' edits may have happened at incorrect processing steps.

A processor that is not always correct will always be faced with trade-offs: Improving timeliness (`FO`) by making it hazard guesses earlier will mean that it is more likely to get something wrong (hurting both `correctness` and `EO`). Reducing `EO`, while being one of our main objectives in Section 6, usually also delays decisions, hurting `FO`. Contrastingly, `FD` may improve when effort is put into

7. Depending on the operational costs for the *consumers* of the incremental output, we may want to assign different costs to addition, revocation and substitution. For example, in the example evaluations discussed in the next section, we assign a cost of 2 to the substitution operation.

Table 1: Summary of metrics for incremental evaluations

name	characterisation	see also
similarity metrics		Sec. 4.2.1
correctness	proportion of ideal intermediate results	Sec. 5.2
r-correctness	incrementally correct **r**elative to the non-incremental result	Sec. 5.1
discounted (r-)correctness	results at time t are compared to gold standard at time $t - \Delta$	Sec. 6.1
p-correctness	results that are a **p**refix of their gold standard	Sec. 5.1
time-adjusted error	weighs indecision depending on how much has been seen	Sec. 5.4
f-score curves	average development of f-score over time	Sec. 5.3
WER/CER curves	development of task-specific metrics over time	Sec. 5.4
timing metrics		Sec. 4.2.2
FO	**f**irst **o**ccurrence of an output increment	
FD	**f**inal **d**ecision for an output increment	
	– measured in milliseconds	Sec. 5.1
	– measured as utterance percentage	Sec. 5.2
diachronic metrics		Sec. 4.2.3
EO	**e**dit **o**verhead, proportion of unnecessary edits	Sec. 5.1, 5.2
correction time	FD – FO, time it takes for an increment to become final	Sec. 5.1, 6.2

reducing spurious edits. High `correctness` is not a suitable target *per se*—a clairvoyant processor outputting the correct final result right from the beginning would even be incorrect against an incremental gold standard up until the very end. However, similarity is a good indicator of whether processing is running as expected and the development of a measure over time can give insights in both processing and properties of the corpus.

As a general rule of thumb, improving a processor's non-incremental performance (that is, the result for the complete input sequence) will also improve the incremental properties, as non-incremental performance is an upper bound for all incremental processing. Finally, good performance in some metric may be more important than in another for a given application. This has to be taken into account when comparing different processors (or post-processing techniques as described in Section 6) for a given task.

4.4 Summary of Metrics

Table 1 gives a short characterisation for the metrics described in this section. We first described `correctness`, as base-measure for similarity and explained how it can be extended to full-blown, task-dependent *similarity metrics* based on traditional, non-incremental similarity metrics, with `f-score` being the most generic, and that it is helpful to plot curves of these metrics over time. Also, we noted that using the processor's final output as gold standard enables the separation of the evaluation of incremental performance from non-incremental performance (`r-correctness`). We also proposed a `time-adjusted error` if a fully incremental gold standard is not available. We then described how the *timing* of a processor's output can be evaluated by measuring when increments first occur (`FO`) and when the final decision for them happens (`FD`). We also discussed anchoring points and temporal scales for these timings. Finally, we discussed the diachronic dimension of incremental processing and used `edit overhead` (possibly with differently weighted edit operations to account for consumer's needs) and `correction time` as our *diachronic metrics*.

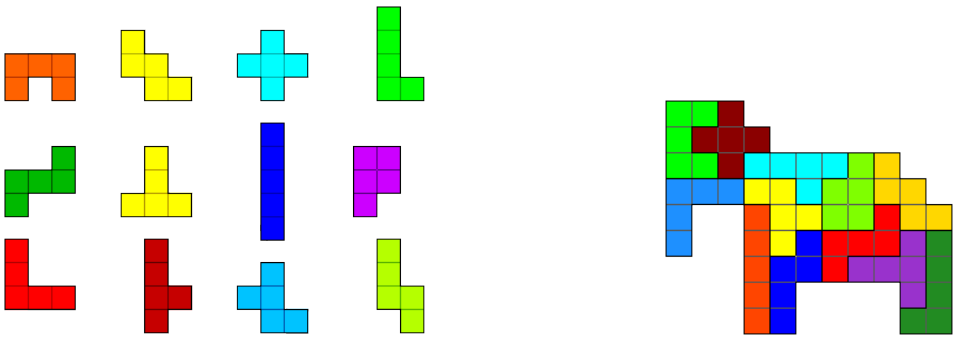

Figure 7: The twelve Pentomino pieces individually (left) and arranged to form an elephant.

We will instantiate these metrics and give exemplary evaluations in the following section before showing how to make trade-offs by exploiting the interrelations between metrics in Section 6. As a quick guide, we have also noted the most relevant subsections for every metric in Table 1.

5. Example Evaluations

To give concrete examples of how the metrics developed in the previous section can be used to make statements about the performance of incremental processing modules, we now present evaluation experiments we have performed on incremental processors that we have developed.[8] The presentation is in order of increasing complexity of the IU networks that are output by the processors. We start with examples of our incremental speech recognizer (which produces sequences of IUs) and proceed to our semantic components, first a reference resolver (which outputs just one current IU) and then a semantic chunker (which creates hierarchically ordered sets of IUs). The last part of this section is dedicated to a look at n-best processing, which includes results from both ASR as well as the semantic chunker.

5.1 Evaluation of Incremental ASR

In this section we report our evaluation effort of incremental speech recognition. We first show how to capture performance with our metrics, and then investigate how the metrics are affected by variations to the quality of the underlying speech recognition.

5.1.1 CAPTURING THE PERFORMANCE OF INCREMENTAL SPEECH RECOGNISERS

We use the large-vocabulary continuous-speech recognition framework Sphinx-4 (Walker et al. 2004) for our experiments, using the built-in LexTree decoder, extended by us to provide incremental results. We trained acoustic models for German and a trigram language model that is based on

8. These data have been published before (for Section 5.1 in (Baumann et al. 2009a), for Section 5.2 in (Schlangen et al. 2009), for Section 5.3 in (Atterer et al. 2009), and finally for Section 5.4 in (Baumann et al. 2009b)) but we reformulate the results in the light of the unified presentation of the metrics outlined above in Section 4. We describe the processor's details and experimental setups only as is necessary for this article's topic; for details see the aforementioned papers.

Additionally, please note that all our experiments were conducted with German data. At first blush, this may be seen as problematic for a claim of general applicability. However, we do not see any principled reason why the methodology should not transfer to other languages. Concrete results (e. g. timing tendencies) may be different *across different languages*, for example due to differences in canonical word order, but the evaluative function of the metrics when applied *within one language* should be preserved, as higher measures in a metric still signal better performance.

Table 2: Base measurements for our incremental ASR component

SER (non-incremental)	68.2 %
WER (non-incremental)	18.8 %
r-correctness	30.9 %
p-correctness	53.1 %
edit overhead	90.5 %
FO: mean, stddev, median	0.276 s, 0.186 s, 0.230 s
FD: mean, stddev, median	0.004 s, 0.268 s, –0.06 s
mean word duration	0.378 s
immediately correct	58.6 %

spontaneous instructions in a puzzle building domain. The test data is also from that domain. In our domain, which is also the basis for the other evaluations reported below, twelve geometric puzzle pieces (*Pentominos*) are to be described, moved, rotated or otherwise manipulated by the experiment participants. In some of the experiments, puzzle pieces are coloured; in dialogue and Wizard-of-Oz experiments, an instructor tells a follower (or the wizard) how to place the pieces, most often to form an elephant as in Figure 7.

For this evaluation, we tried to separate incremental from non-incremental performance. Hence, we give the non-incremental word error rate (and sentence error rate) and calculate incremental `r-correctness` relative to the ASR's final results. Additionally, we observed that the ASR often lagged behind at word boundaries: The previous word was usually extended over a few hundred milliseconds, before the next word started. To account for such errors, we also measure `p-correctness`.

Table 2 shows the results for our evaluation. The ASR's non-incremental WER is 18.8 % (which is a fair value for spontaneous speech (Finke et al. 1997)), and of all incremental hypotheses, 30.9 % are correct w. r. t. the final hypothesis, and 53.1 % are at least a prefix of what would be correct. Still, we see a substantial `edit overhead` of 90.5 %, meaning that there were ten times as many edits as there would be in a perfect system.

Timing metrics were calculated in absolute wall-clock time from the left (`FO`) respectively right (`FD`) word boundaries. Their means, standard deviation and median are also given in Table 2. On average, the correct hypothesis about a word becomes available 276 ms after the word has started (`FO`). With a mean word duration of 378 ms this means that information becomes available after roughly $3/4$ of the word have been spoken. On average, a word becomes final (i. e. is not changed any more) when it has just about ended (mean(FD) = 0.004).

We also analysed the `correction time` for `r-correct` words (`FD`−`FO`). Of all words, 58.6 % were immediately correct. The distribution of correction times for correct words is given in Figure 8, giving the percentage of words with correction times equal to or lower than the time on the x-axis. While this starts at the initial 58.6 % of words that were immediately correct, it rises above 90 % for a correction time of 320 ms and above 95 % for 550 ms. Inversely this means that we can be certain to 90 % (or 95 %) that a current correct hypothesis about a word will not change anymore once it has not been revoked for 320 ms (or 550 ms respectively). (Notice that we should calculate correction time for *all* and not just for r-correct word IUs. This is, however, a computationally complex task, as in that case timings of *all* intermediate hypotheses have to be computed. Our experience is that wrong hypotheses die off quickly, so that a curve of correction times for all hypothesized words – not just correct ones – should be even steeper.)

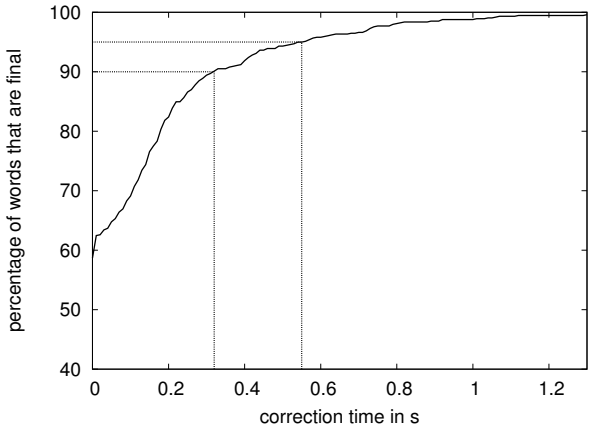

Figure 8: Distribution of correction times (FD − FO).

We concluded from this experiment that our metrics give us useful information about the performance of the component. For example, we now know when (on average) we can expect incremental output in relation to when it has been spoken, and how certain we can be about the 'finality' of this output.

5.1.2 STABILITY OF THE INCREMENTAL MEASURES

To test the dependency of our measures on details of the specific setting, such as audio quality and language model reliability, we varied these factors systematically, by adding white noise to the audio and changing the language model weight relative to the acoustic model (LM weight was 8 in the experiment mentioned above). We varied the noise to produce signal to noise ratios ranging from hardly audible (−20 dB), through annoying noise (−10 dB) to barely understandable audio (0 dB).

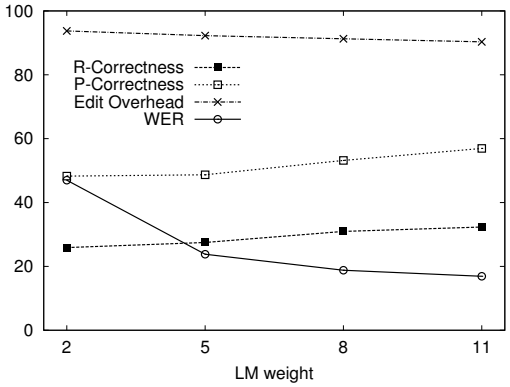

Figure 9: Correctness, EO and non-incremental Word Error Rate (WER) with varied language model weights.

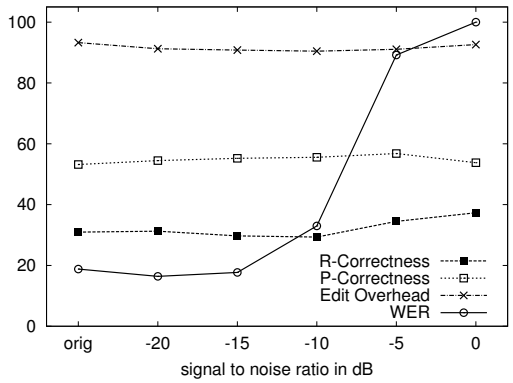

Figure 10: Correctness, EO and non-incremental Word Error Rate (WER) with additive noise.

Figure 9 gives an overview of the performance with different LM weights and Figure 10 with degraded audio signals. Overall, we see that `r-correctness` and `EO` remain remarkably stable with different LM and AM performance and correspondingly degraded non-incremental WER. A tendency can be seen that higher LM weights result in higher `correctness` and lower `EO`. A higher LM weight leads to less influence of acoustic events which dynamically change hypotheses, while the static knowledge from the LM becomes more important, thus reducing jitter.

We conclude that non-incremental and incremental measures are highly decoupled and both give fairly independent views of the processor's overall performance across the variations of the setting.

5.2 Evaluation of Incremental Reference Resolution

In this section we evaluate a component for incremental reference resolution. We define incremental reference resolution (IRR) as the task of inferring the referent referred to in an utterance, while this utterance is going on. The component we evaluate uses a data-driven methodology, and was trained (and cross-validated) on a corpus of 684 utterances, each referencing one of the puzzle pieces in descriptive language.

The IRR processor consumes word IUs. For evaluation, these were derived from text transcriptions of the test corpora used. Employing a referent-specific language model trained with the SRI LM package (Stolcke 2002), the processor arrives at a likelihood belief distribution over the twelve pieces in the domain after each word IU. This distribution was updated with each incoming IU and the top-most referent was evaluated against the gold standard.

In terms of the metrics defined here, the study determined `edit overhead`, `FO` and `FD`. Additionally, we looked at how `correctness` improved with utterance completion. Notable experimental conditions included using n-best lists and various language models trained with and without pseudo-words for hesitations and silences to test robustness against disfluencies. In addition, an adaptive threshold strategy was employed such that a new belief decision is only made if the maximal value after the current word IU is above a certain threshold, where this threshold is reset every time this condition is met.

The complete set of experimental results is displayed in Table 3. Best results for `FO` and `FD` were found at 30.43 % (n-best with disfluencies) and 70.89 % (adaptive threshold, no disfluency) of utterance completion respectively. `Correctness` was topped at 37.81 % using n-best processing with disfluencies. `EO` was low at 69.61 % using adaptive threshold strategy.

These figures allow some initial performance analysis along the incremental dimensions. Timing metrics `FO` and `FD` show promising results, as the processor was able to correctly hypothesise user

Table 3: First occurrence, first decision, edit overhead and correctness for our incremental reference resolution component with (w/ h) and without hesitations (w/o h) used by the processor.

model:	n-best		rnd-nb	adapt		max		random
measures	w/ h	w/o h		w/ h	w/o h	w/ h	w/o h	
FO	30.4 %	33.7 %	29.6 %	53.9 %	55.3 %	46.6 %	49.3 %	42.6 %
FD	87.7 %	85.0 %	97.1 %	71.2 %	70.9 %	96.1 %	94.3 %	98.4 %
EO	93.5 %	90.7 %	96.7 %	69.6 %	67.7 %	92.6 %	89.4 %	93.2 %
correctness	37.8 %	36.8 %	23.4 %	23.0 %	26.6 %	17.8 %	20.2%	7.8%

intention on average as early as after a third of the input utterance and offer a stable hypothesis after roughly two-thirds (albeit under different experimental conditions). Also, reference resolution `edit overhead` stayed far below that of the ASR, discussed above. We conclude that the metrics were useful in describing the performance of this component, and allow us to perform meaningful comparisons between the variations of the model.

5.3 Evaluation of Incremental Semantics Construction

In addition to the statistical IRR component described above, we evaluated a rule-based semantic chunker (described in (Atterer and Schlangen 2009)) on this task. The chunker also consumes word IUs and, based on chunks defined in a grammar, fills slots in a semantic frame. We present results from two evaluations of this component, one in which the input words came from corpus transcriptions and another in which they were taken from actual ASR output.

5.3.1 EVALUATION WITH IDEAL INPUT

The initial text-only evaluation using data from task-oriented corpora in the puzzle domain described above provided timing and similarity measures for all slots in a semantic frame (piece involved, action, possibly parameters of that action). `FO` and `FD` were determined for each utterances' frame interpretation. The frame `FO` was below 40 % of utterance completion for almost all utterances, with gold standard transcriptions filling at least one slot. `FD` was generally achieved between 85 % utterance completion and the utterance's end.[9]

The results from the study are re-represented in Figure 11 as `recall`, `precision` and `f-score`. The gold standard available for evaluation did not include linkage information between individual input and output IUs, so precise reasoning about each input IU's performance with respect to correctness or timing was impossible. For this reason, we reverted to using `f-score` which provides a natural workaround for this shortcoming.

Portability of experimental results is one of our aims in re-presenting this data here. As such, this representation provides a point of comparison to (Sagae et al. 2009), who present a similar approach employing similar measures. While training of the incremental semantic models for the two approaches varies, the experimental findings are very similar. Both models exhibit an initial `f-score` of approximately 20 % progressing to an asymptotical final of just below 80 %, exhibiting similar gains along the way. This similarity provides some preliminary conclusions about the nature of stability and accuracy of partial complex NLU hypotheses. Building on the work by Sagae et al. (2009), DeVault et al. (2009) train a classifier to decide on when the interpretation of an utterance is likely to not improve any further. This is similar in spirit to our output optimisation techniques discussed in Section 6.

5.3.2 EVALUATION WITH ASR OUTPUT

We also evaluated the chunker with actual ASR output (and including ASR timing information) using a sub-domain in which users were asked to describe one of the puzzle pieces using descriptive language.[10] We collected an evaluation corpus in a Wizard of Oz study; the gold standard for the

9. Note that these numbers are based on word counts rather than wall clock timing, which would be less meaningful for a word-based processor.

10. Since we are only interested in one of the slots filled by the chunker, the one referring to the piece in question, the task may perhaps more aptly be described as chunk-based reference resolution.

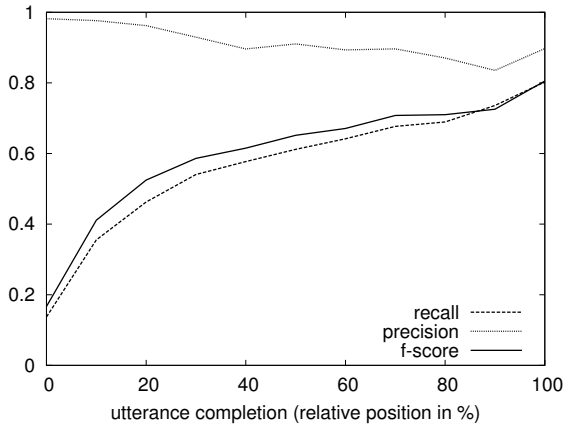

Figure 11: Recall, precision and f-score of the semantics component at different positions in the utterance (from Atterer and Schlangen (2009), metrics recalculated).

ASR was hand-transcribed, while gold for the chunker was derived from meta-information (each scene of the study was associated with a single puzzle piece.) The one-best case in that study is available as a baseline, resulting in FO at 51.2 %.

This is in line with some of the findings from evaluating our IRR component in Section 5.2. This observation is relevant in several respects. The FO data is a measure of response time. By seeing similar results in both components for responsiveness, we validate our hypothesis that something can be known about a user's intention early on in the utterances. In addition, as we noted when comparing our results with (Sagae et al. 2009), we now have a rough indication what this responsiveness might be (or at least an operating point on which to improve, see Section 6 below). Lastly, since these two components can be charged with solving overlapping tasks (in our case resolving references quickly), and since our intention here is to enable informed comparisons of two or more components, we now have the tools to do so from a timing perspective.

5.4 Evaluation with Incremental ASR N-Best Results

The ASR results of the study described in the previous subsection also included n-best lists, which we used to gauge the overall effect, beneficial or adverse, of considering recognition alternatives on the three dimensions of incremental evaluation: similarity, timing, and evolution.

For each incremental ASR result obtained we measured WER, as well as correctness as a concept error rate, CER, as per the reference resolution task.[11] Notice that we made CER *time adjusted*, i. e. an unfilled concept was counted as a small error early on in the utterance, but as a full error towards the end. We then calculated oracle and anti-oracle scores for WER and CER. An *oracle* error rate for an n-best list is the error rate of the best hypothesis contained in the list; correspondingly, an *anti-oracle* score is the worst score in the list. This gave us potential gains in WER, CER, and FO, associated with being able to identify the best hypothesis among the N, as well as the risk associated with picking the worst instead. In addition, by observing the output for all N over time, we were able to observe

11. For performance reasons we limited the ASR's beam width and set N to a maximum of 100,000.

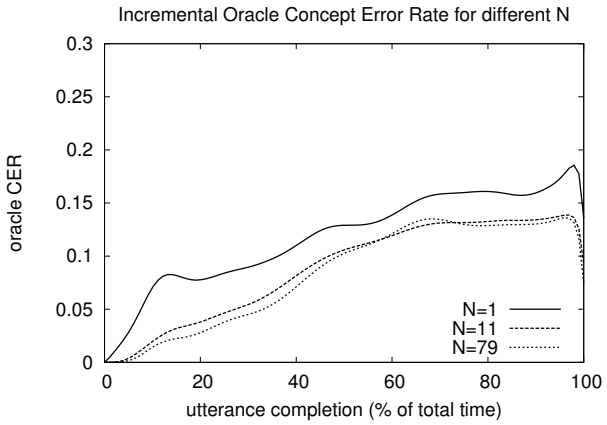

Figure 12: Incremental oracle concept error rate for selected N in n-best processing.

the effect of n-best processing on `edit overhead`, as presumably the number of unnecessary changes would increase.

Results from ranking the n-best list for `oracle WER` were sobering. Using the recognition results associated with `oracle WER`, provided little or no gain in `CER`. However, there were timing benefits: We observed a 20 % potential gain in `FO` from a baseline 51.2 % to an oracle 41.0 % (i. e. ASR results in the n-best list existed that resolved references on average that much sooner than a one-best result would have). However, this gain came with a significant (150-fold) increase in `edit overhead`, which is clearly unacceptable and results from the sporadically huge n-best lists during recognition.

We repeated the evaluation with capped n-best lists. Already low values of $maxN$ promised similar (but smaller) gains in `correctness`, the same reduction in `FO`, but a significant reduction in `edit overhead` (e. g. at a low $maxN$ of 11, it halved to 60-fold over the one-best baseline). Graphs for various settings of $maxN$ and corresponding `oracle CER` are shown in Figure 12. Note that `CER` rises because we employed an `adjusted error` as mentioned above. The final drop can be attributed to a syntactic phenomenon in the data set.

Without a method for determining the best result in an n-best list the benefits just outlined remain hypothetical. We will now discuss methods that *are* available for improving incremental processors.

6. Optimising the Output of Incremental Processors

In the previous section, we showed some example incremental processors and how they fared in evaluations of their respective tasks. In this section, we demonstrate simple (incremental) post-processing techniques to improve the output of an incremental processor.

These techniques are aimed mainly at the reduction of `edit overhead`, as this was one of the major problems that came up in our evaluations. So far, they are mostly only tested on ASR output (where the problem of edit overhead is most severe), but apply equally to other incremental processors.[12] The graphs presented in these subsections are similar in spirit to Zilberstein's (1996)

12. This is especially true for other kinds of 'input' processors, that receive input from the outside world (e. g. gesture and face recognition).

performance profiles allowing to analyse where some delay can hopefully result in the largest improvements of an overall system.

Finally, we give a theoretical discussion of how post-processing is helpful even if edit overhead is not an issue, as is typically the case in what we call 'beat-driven' systems.

6.1 Right Context

Incremental processing is hard due to the fact that a processor is expected to output a result as soon as some input becomes available. Obviously, some results will be wrong, as the correct final output can not yet be determined from the very beginning (cmp. the discussion in Section 4.2.3 above). A simple strategy to improve results is to allow the processor some *right context* which it can use as input, but for which it does not yet have to provide any output. For example, typical ASR systems use this strategy internally at word boundaries (with very short right contexts) in order to restrict the language model hypotheses to an acoustically plausible subset (Ortmanns and Ney 2000).

In the experiment described here, we allow the ASR a larger right context of size Δ by taking into account at time t the output of the ASR up to time $t - \Delta$ only. That is, what the ASR hypothesises about the right context is considered to be too immature and is discarded, while the hypotheses about the input up to $t - \Delta$ have the benefit of a lookahead up to t. This reduces the *jitter*, which is found mostly to the very right of the incremental hypotheses. Thus, we expect to reduce the `edit overhead` in proportion with Δ. On the other hand, allowing the use of a right context leads to the current hypothesis *lagging behind* the gold standard. (Correspondingly, `FO` increases by Δ.) Obviously, using only information up to $t - \Delta$ has averse effects on correctness as well, as this measure evaluates the word sequences up to time t which may already contain more words (those in the right context). To account for this lag (which is known in advance), we also report a `discounted r-correctness`.

Figure 13 details the results for the data from Section 5.1 with right context between 1.5 s and -0.2 s. (The x-axis plots Δ as negative values, with 0 being "now". Results for a right context (Δ) of 1.2 can thus be found 1.2 to the left of 0, at -1.2.) We see that at least in the discounted measure, fixed lag performs quite well at improving both the processor's `correctness` and `EO`. This is due to the fact that ASR hypotheses become more and more stable when given more right context. However even for fairly long lags, late edits cannot be ruled out entirely.

To illustrate the effect of a system that does not support editing of hypotheses but immediately commits itself to an hypothesis we plot the *`fixed` `WER`* that would be reached by such a system after a right context of Δ. As can be seen in the figure, it is extremely high for low right contexts (exceeding 100 % for $\Delta \leq 400$ ms) and remains substantially higher than the non-incremental `WER` even for fairly large right contexts. Actually, the `WER` plot by Wachsmuth et al. (1998) looks very similar.

As expected, the analysis of timing measures shows an increase with larger right contexts with their mean values quickly approaching Δ (or $\Delta -$ mean word duration for `FD`), which are the lower bounds when using right context. Correspondingly, the percentage of immediately correct hypotheses increases with right context reaching 90 % for $\Delta = 580$ ms and 98 % for $\Delta = 1060$ ms.

Finally, we can extend the concept of right context into negative values, predicting the future, as it were. By choosing a *negative* right context, in which we extrapolate the last hypothesis state by Δ into the future, we can measure the correctness of our hypotheses correctly predicting the near future. The graph shows that 15 % of our hypotheses will still be correct 100 ms in the future and 10 % will

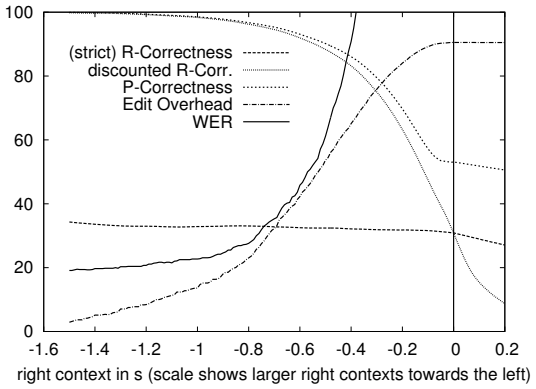

Figure 13: Correctness, EO and fixed-WER for varying right contexts Δ.

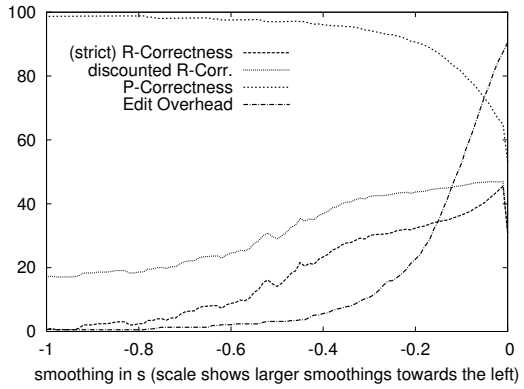

Figure 14: Correctness and Edit Overhead for varying smoothing lengths.

still be correct for 170 ms. Unfortunately, there is no way to tell apart hypotheses that will survive and those which will soon be revised.

6.2 Hypothesis Smoothing

In the previous method we suppressed wrong edits by avoiding the recognition jitter that the ASR produces in the 'youngest' parts of its recognition results. In this section, we look at the diachronic evolution of the results and use the age of an hypothesized increment as a cue. (This is similar in spirit to using `correction time` to determine the certainty that an IU will last.) We simply only pass on an IU if it reaches a certain age N, that is, it is part of N consecutive hypotheses. To illustrate the process with $N = 2$ we return to Figure 2. Neither of the words "bis", or "es" would ever be output, because they are only present for one time-interval each. Edits would occur at the following times: $\oplus(\text{nimm})$ at t_6, $\oplus(\text{bitte})$ at t_{10} (only then is "bitte" the result of two consecutive hypotheses) and $\oplus(\text{das})$ at t_{12} and so on. With a smoothing factor of $N = 2$, no words are revoked in the example, because all revocations last for only 1 time-step. Yet, in general, changes can of course still occur, namely if it is called for by at least N consecutive time-steps.

For this strategy, as can be seen in Figure 14, `edit overhead` falls rapidly, reaching 50 % (for each edit necessary, there is one superfluous edit, `EO` *parity*) with only 110 ms and 10 % with 320 ms. (The same thresholds are reached through the use of right context at 530 ms and 1150 ms respectively as shown above.) Likewise, the prefix correctness improvements are better than when using right context, indicating the effectiveness of the technique. At the same time, `r-correctness` is poor. This is due to correct hypotheses being held back too long due to the hypothesis sequence being interspersed with wrong hypotheses (which only last for few consecutive hypotheses), resetting the counter until the add message (for the prevalent and potentially correct word) is sent. Still, the positive effects of hypothesis smoothing outweigh its disadvantages, as we reach `EO parity` with N at 110 ms instead of $\Delta = 530$ ms leading to an increase in `FO` of 140 ms and in `FD` of 67 ms. When comparing to increases of at least 530 ms using right context, results are rather impressive.

6.3 Optimising for Beat-Driven Systems

When we described the advantages of the above methods, we put a special focus on `edit overhead`, as this is a very important metric in our event-based spoken dialogue system architecture (Schlangen and Skantze 2009, Schlangen et al. 2010).

Some incremental spoken dialogue systems (DeVault et al. 2009, Raux and Eskenazi 2009) follow a different approach to incremental processing, which we here call the *beat-driven approach*. In such systems, processing is done repeatedly (at a fixed or flexible rate) on the partial input available so far. While the system and its components may or may not know about the fact that they are repeatedly processing partial (and unfolding) input, they do not usually take that into account while processing. Instead, they completely reprocess all input at every beat.[13] Obviously, for such a system edit overhead is not an issue, because all intermediate results are recomputed at every beat regardless of previous input. In this section, we want to show how an incremental processor may optimise its output for a beat-driven consumer.

Again, we take as an example our incremental speech recognition component. Our processor, as explained above, is suited to only output messages once new information is available. In the previous subsections, we have shown ways of increasing the reliability of such messages, and have mostly been concerned with reducing the `edit overhead` from messages that have to be revoked and changed. Here we are concerned with the increased `correctness` (and `prefix-correctness`) of the output when the above improvements are carried out.

Consider a system with a beat β of 200 ms, which polls the ASR at every 200 ms for its most recent result. In such a setting `correctness` remains the same (as—on average—the results at every 200 ms will be just as good or bad as all other incremental results). However, new output increments will be delayed on average by $\beta/2$ because they are only registered on the next beat (which is, on average, $\beta/2$ into the future). Thus, as timing is already worse due to querying results only when the beat falls, adding an additional small delay through the use of smoothing or right context as described above may be well justified by the resulting increase in correctness.

7. Conclusions and Future Work

In this article we have presented a general approach to representing the output of what we call incremental processors, that is, components in incremental dialogue processing systems. We have discussed how gold standards can be created in the same format, to enable evaluation based on the comparison of actual and ideal output. Sometimes, resources for producing a fully incremental gold standard may not be available; we have discussed ways to approximate one in these cases. We have then presented three types of metrics: (1) timing metrics, (2) similarity metrics and (3) diachronic metrics, measuring incrementality in terms of speed, accuracy and change over time, respectively. Presenting our own evaluations of various incremental modules, we have demonstrated how these metrics can usefully characterise performance and how they can guide optimisation techniques that improve performance without changes to the internal workings of the modules. We hope that these metrics and methods will be of use in the wider community and improve comparability of results and the exchangeability of modules.

13. We do not claim that one or the other method is superior to the other, at least not from a system point of view. (We do think, however, that repeated re-processing is psycho-linguistically rather implausible.) While starting from previous results potentially saves processing cycles, it also incurs book-keeping overheads. Also, the beat-driven approach allows to easily integrate originally non-incremental components into an incremental system.

Our discussion focused on incremental processors that are tasked with input recognition and interpretation. We did not discuss how these metrics can be applied to the evaluating of processors that are concerned with 'later' aspects of dialogue, such as producing system behaviour and generating output. There are some recent efforts in this direction, such as (Skantze and Hjalmarsson 2010, Buß and Schlangen 2010) and (Buß et al. 2010). How exactly the metrics presented here relate to these efforts is an open question we currently explore.

Also missing from our discussions is the issue of how the characterisation of the performance of *components* relates to that of *systems*, or more concretely, how improvements of the former can lead to improvements of the latter. Zilberstein (1996) analyses how *performance profiles* for anytime algorithms can be used to find an optimal resource allocation when combining several components to form a complex system. In incremental processing, a similar analysis of how incremental metrics (e. g. timing) aggregate in a complex system and how and where to best apply optimisation techniques in the system would make for very helpful future work. The profiles which we report for the optimization techniques in Section 6 can be seen as a first step in this direction. Finally, feed-back loops as in Wirén's (1992) *full incrementality* may incurr additional challenges.

On a less technical account, the 'success' of spoken dialogue systems is usually measured in terms such as user satisfaction or task success. The prediction is that that there is at least an indirect connection between such subjective measures and the performance metrics discussed in this article, since user-facing behaviours that are enabled by incremental processing (e. g., fast turn-taking, backchannels; see discussion in (Buß and Schlangen 2010)) should benefit from improvements in incremental processing. The exact nature of this connection, however, and the place of the metrics in system-evaluation frameworks like (Walker et al. 1997) or (Möller and Ward 2008) will have to be explored in future work.

Acknowledgements

This work was funded by a DFG grant in the Emmy Noether programme. We wish to thank the anonymous reviewers for their very helpful comments.

References

Gregory Aist, James Allen, Ellen Campana, Carlos Gallo, Scott Stoness, Mary Swift, and Michael Tanenhaus. Incremental dialogue system faster than and preferred to its nonincremental counterpart. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society*, pages 761–766, Nashville, USA, August 2007.

James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3):213–228, 2000.

Jan Willers Amtrup. *Incremental Speech Translation*. Springer Verlag, Berlin, Germany, 1999.

Michaela Atterer and David Schlangen. RUBISC – a Robust Unification-Based Incremental Semantic Chunker. In *Proceedings of SRSL 2009, the 2nd Workshop on Semantic Representation of Spoken Language*, pages 66–73, Athens, Greece, March 2009.

Michaela Atterer, Timo Baumann, and David Schlangen. No sooner said than done? Testing the incrementality of semantic interpretations of spontaneous speech. In *Proceedings of Interspeech*, pages 1855–1858, Brighton, UK, September 2009.

Timo Baumann, Michaela Atterer, and David Schlangen. Assessing and improving the performance of speech recognition for incremental systems. In *Proceedings of NAACL-HLT*, pages 380–388, Boulder, USA, May 2009a.

Timo Baumann, Okko Buß, Michaela Atterer, and David Schlangen. Evaluating the potential utility of ASR N-best lists for incremental spoken dialogue systems. In *Proceedings of Interspeech*, pages 1031–1034, Brighton, UK, September 2009b.

Manuela Boros, Wieland Eckert, Florian Gallwitz, Günther Görz, Gerhard Hanrieder, and Heinrich Niemann. Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proceedings of the 4th ICSLP*, pages 1009–1012, Philadelphia, USA, October 1996.

Okko Buß and David Schlangen. Modelling sub-utterance phenomena in spoken dialogue systems. In *Proceeding of PozDial, the 14th International Workshop on the Semantics and Pragmatics of Dialogue*, pages 33–41, Poznań, Poland, June 2010.

Okko Buß, Timo Baumann, and David Schlangen. Collaborating on utterances with a spoken dialogue system using an isu-based approach to incremental dialogue management. In *Proceedings of the SIGDIAL Conference*, pages 233–236, Tokyo, Japan, September 2010.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: A survey and a new proposal. In *Proceedings of LREC*, pages 447–454, Granada, Spain, May 1998.

Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54, Cambridge, USA, August 1988.

David DeVault, Kenji Sagae, and David Traum. Can I finish? learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL Conference*, pages 11–20, London, UK, September 2009.

Michael Finke, Petra Geutner, Hermann Hild, Thomas Kemp, Klaus Ries, and Martin Westphal. The Karlsruhe-Verbmobil speech recognition engine. In *Proceedings of ICASSP*, pages 83–86, Munich, Germany, April 1997.

Wolfgang Finkler. *Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen*. Dissertationen zur Künstlichen Intelligenz. infix Verlag, Sankt Augustin, Germany, 1997.

Carlos Gómez Gallo, Gregory Aist, James Allen, William de Beaumont, Sergio Coria, Whitney Gegg-Harrison, Joana P. Pardal, and Mary Swift. Annotating continuous understanding in a multimodal dialogue corpus. In *Proceeding of DECALOG, the 11th International Workshop on the Semantics and Pragmatics of Dialogue*, pages 75–82, Trento, Italy, June 2007.

Markus Guhe. *Incremental Conceptualization for Language Production*. Lawrence Erlbaum Associates, Mahwah, USA, 2007.

Yulan He and Steve Young. Semantic processing using the hidden vector state model. *Computer Speech and Language*, 19(1):85–106, 2005.

Silvan Heintze, Timo Baumann, and David Schlangen. Comparing local and sequential models for statistical incremental natural language understanding. In *Proceedings of the SIGDIAL Conference*, pages 9–16, Tokyo, Japan, September 2010.

Bernd Hildebrandt, Hans-Jürgen Eikmeyer, Gert Rickheit, and Petra Weiß. Inkrementelle Sprachrezeption [incremental language understanding]. In *KogWis: Proceedings der 4. Fachtagung der Gesellschaft für Kognitionswissenschaft*, pages 19–24, Bielefeld, Germany, September 1999.

Melvin J. Hunt. Figures of merit for assessing connected-word recognisers. *Speech Communication*, 9(4):329–336, 1990.

Yoshihide Kato, Shigeki Matsubara, and Yasuyoshi Inagaki. Stochastically evaluating the validity of partial parse trees in incremental parsing. In *Proceedings of the ACL Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 9–15, Barcelona, Spain, July 2004.

Anne Kilger and Wolfgang Finkler. Incremental generation for real-time applications. Technical Report RR-95-11, DFKI, Saarbrücken, Germany, 1995.

Staffan Larsson and David R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3–4):323–340, 2000.

Sebastian Möller and Nigel G. Ward. A framework for model-based evaluation of spoken dialog systems. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 182–189, Columbus, USA, June 2008.

NIST Website. RT-03 Fall Rich Transcription, 2003. URL `http://www.itl.nist.gov/iad/894.01/tests/rt/2003-fall/index.html`.

Stefan Ortmanns and Hermann Ney. Look-ahead techniques for fast beam search. *Computer Speech and Language*, 14(1):15–32, 2000.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, USA, July 2002.

Antoine Raux and Maxine Eskenazi. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of NAACL-HLT*, pages 629–637, Boulder, USA, May 2009.

D. Raj Reddy, Lee D. Erman, Richard D. Fennell, and Richard B. Neely. The Hearsay-I speech understanding system: An example of the recognition process. *IEEE Transactions on Computers*, C-25(4):422–431, 1976.

Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558, 2009.

Kenji Sagae, Gwen Christian, David DeVault, and David Traum. Towards natural language understanding of partial speech recognition results in dialogue systems. In *Proceedings of NAACL-HLT*, pages 53–56, Boulder, USA, May 2009.

David Schlangen and Gabriel Skantze. A general, abstract model of incremental dialogue processing. In *Proceedings of EACL*, pages 710–718, Athens, Greece, March 2009.

David Schlangen and Gabriel Skantze. A general, abstract model of incremental processing. *Dialogue and Discourse*, 2(1):83–111, 2011.

David Schlangen, Timo Baumann, and Michaela Atterer. Incremental reference resolution: The task, metrics for evaluation, and a bayesian filtering model that is sensitive to disfluencies. In *Proceedings of the SIGDIAL Conference*, pages 30–37, London, UK, September 2009.

David Schlangen, Timo Baumann, Hendrik Buschmeier, Okko Buß, Stefan Kopp, Gabriel Skantze, and Ramin Yaghoubzadeh. Middleware for incremental processing in conversational agents. In *Proceedings of the SIGDIAL Conference*, pages 51–54, Tokyo, Japan, September 2010.

Gabriel Skantze and Anna Hjalmarsson. Towards incremental speech generation in dialogue systems. In *Proceedings of the SIGDIAL Conference*, pages 1–8, Tokyo, Japan, September 2010.

Gabriel Skantze and David Schlangen. Incremental dialogue processing in a micro-domain. In *Proceedings of EACL*, pages 745–753, Athens, Greece, March 2009.

Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *Proceedings of the 7th ICSLP*, pages 901–904, Denver, USA, September 2002.

Sven Wachsmuth, Gernot A. Fink, and Gerhard Sagerer. Integration of parsing and incremental speech recognition. In *Proceedings European Signal Processing Conference*, pages 371–375, Rhodes, Greece, September 1998.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of ACL and EACL*, pages 271–280, Madrid, Spain, July 1997.

Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Technical Report SMLI TR2004-0811, Sun Microsystems Inc., 2004.

Mats Wirén. *Studies in Incremental Natural Language Analysis*. Unpublished doctoral dissertation, Linkoping University, Sweden, 1992.

Steve J. Young, NH Russell, and JHS Thornton. Token passing: A simple conceptual model for connected speech recognition systems. *Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR*, 38, 1989.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–83, 1996.