Evaluating Automatic Speech Recognition and Natural Language Understanding in an Incremental Setting

Ryan Whetten
ryanwhetten@u.boisestate.edu
Computer Science
Boise State University

Enoch Levandovsky
enochlevandovsky@u.boisestate.edu
Computer Science
Boise State University

Mir Tahsin Imtiaz
tahsinimtiaz@u.boisestate.edu
Computer Science
Boise State University

Casey Kennington
caseykennington
boisestate.edu
Computer Science
Boise State University

Abstract

Spoken dialogue systems enable people to interact with machines using speech, many of which involve the use of automatic speech recognition and language understanding in order to react to and determine a decision about how to respond. Unlike humans, many systems operate on complete sentences, waiting for a length of silence before attempting to process the input. In contrast, incremental spoken dialogue systems enable faster and more natural interaction by operating at a more fine-grained level. In this work, we evaluate six speech recognizers and RASA for language understanding in an incremental spoken dialogue system. The results suggest that, for speech recognition, online/cloud models can be slower and less stable than local models and we show that incremental language understanding can enable a system to make decisions earlier than waiting for the end of the utterance.

1 Introduction

Interacting with technology using a spoken dialogue system (SDS) has become more prevalent with applications such as voice search, dictation, and virtual assistants (Yu and Deng, 2016). A fundamental step in how these systems process input, whether implemented in a chatbot, on a website, or on a robot, is to understand what is uttered by the user and produce some kind of action, often by responding using speech back to the user. This is usually performed by first transcribing what the user says using *Automatic Speech Recognition* (ASR), followed by using a model of *Natural Language Understanding* (NLU) to map from the ASR's transcript to a computable abstraction, often a semantic frame. Existing models for NLU, including large language models, are becoming more commonplace, but most have an important drawback: they operate on complete sentences.

Incremental systems, in contrast, operate at more fine-grained levels of information, usually at the word-level instead of the sentence-level, and begin to process the input as soon as it is received. Incremental systems have been shown to offer a more natural interaction (Aist et al., 2007; Edlund et al., 2008) likely due to the fact that humans also produce and understand language incrementally (Tanenhaus and Spivey-Knowlton, 1995). However, most existing ASR and NLU models are either non-incremental or have not been evaluated incrementally. With incremental systems offering more natural interactions, it is crucial to evaluate and understand how ASR and NLU perform in an incremental setting.

In the spirit of prior work, which evaluated several existing ASR models and their relationship to NLU to inform the research community (Morbini et al., 2013), in this work, we evaluate six ASR models (two online/cloud and four local). However, in this work, we experiment in an incremental SDS setting. We evaluate on two English datasets using incremental metrics proposed from Baumann et al. (2009, 2016), as well as propose a new metric *Revokes per Second* to observe how frequently the predictions of an ASR model change (section 3.1.1). Moreover, we incrementalize a recent version of RASA, a framework for NLU and building conversational agents, and evaluate its incremental performance on the SNIPS and SLURP datasets (Coucke et al., 2018; Bastianelli et al., 2020a) in conjunction with an ASR model.

Results show that cloud ASRs, although being some of the most accurate, can have a higher latency and change predictions more frequently than the local ASRs. For incremental NLU, results show that even without a perfect transcript (i.e. a transcript generated by an ASR instead of the ground-truth), a system could be ready to take an action up to six words on average before the end of an utterance. The results provide insights when considering which ASR to use and for designing SDSs that are more natural and responsive in their in-

teractions. All of the models are implemented as modules in the Retico framework (Michael, 2020) for ease of use in incremental systems.

2 Background & Related Work

Incremental ASR Many ASR models operate incrementally in that they produce word or sub-word outputs as the recognition unfolds (Morbini et al., 2013). This ability to function incrementally is an important requirement for spoken dialogue systems (SDS), especially ones that are multimodal or part of a robot platform because there is a high expectation of timely interaction from human dialogue partners (Kennington et al., 2020). Although ASR can function incrementally, most ASR models use the word-error-rate (WER) metric for evaluation, even in conversational settings (Morris et al., 2004; Morbini et al., 2013; Georgila et al., 2020). However, WER solely captures the end performance and does not take into account incremental performance and speed. Morbini et al. (2013) mentions the importance of incremental ASR stating, "incremental results allow the system to react while the user is still speaking", yet evaluates ASR performance using only WER. We build on this prior work by using WER as well as metrics to evaluate incremental performance.

Baumann et al. (2009, 2016) proposed metrics for evaluation of incremental performance such as Edit Overhead, Word First Correct Response, Disfluency Gain, and Word Survival Rate. All of the metrics, including WER, can be classified into one of the following three general areas of interest: overall accuracy, stability (which can be thought of as measuring the incremental performance), and speed. However, these metrics focus on discrete word-level output and not the relationship of between incremental performance and speed. To capture the relationship between incremental performance and speed, we propose to measure the number of *Revokes per Second* (introduced in Section 3.1.1).

Incremental NLU NLU maps words onto a meaning representation, such as a semantic frame (see example in Section 3.2). Among many methods for doing this mapping, in this paper, we focus on RASA (Bocklisch et al., 2017) which is open source and has been shown to work well for NLU (Liu et al., 2019). RASA was made to work incrementally in Rafla and Kennington (2019), but subsequent updates to RASA have left the incremental version obsolete and the original evaluation did not include ASR as the evaluation was performed using only text data (i.e. ground-truth transcriptions). In this work, we incrementalize a recent version of RASA that will be more maintainable in the future and we evaluate performance using incrementally produced transcriptions from an ASR as well as the ground-truth transcriptions.

3 Methods

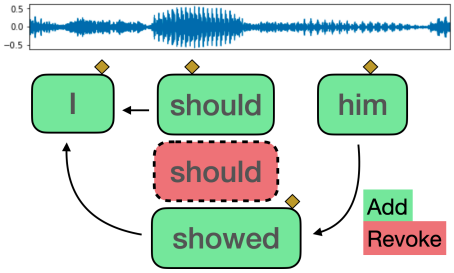

Figure 1: An example of `adds` and `revokes`. The word *should* is added, then `revoked` and replaced by *showed*. The diamonds represent the time when the predictions are made.

The Incremental Unit Framwework We adopt the *Incremental Unit* framework from Schlangen and Skantze (2009) for its flexible design. The framework is built around *incremental units* (IU), discrete pieces of information (e.g., a chunk of audio, a word, an image), that are produced by specific modules. These modules process IUs as input and can pass IUs that they produce to other modules. For example, a microphone module can output chunks of audio as IUs that are passed to an ASR module that outputs individual words as IUs which can in turn be passed on to an NLU module, and so on.

The IU framework has provisions for handling cases where a module's output was found to be in error, given new information. To handle these cases, there are three operations for IUs: `add` (to mark an IU to be added to the output), `revoke` (to mark an IU to be removed from the output), and `commit` (to mark that an IU will not longer change). A perfect ASR would only `add` new words to the growing list of previously recognized words. But as most ASRs have errors—particularly when they work incrementally—the `revoke` operation allows the ASR module to remove an erroneous IU and replace it (i.e., through another `add` operation) in the recognized output. Importantly, the `revoke`

operation propagates to downstream modules that may have acted on prior input, signalling the error. An example of incremental `add` and `revoke` for ASR is shown in Figure 1. We use Retico, a Python implementation of the IU framework, to implement and evaluate ASR and NLU models (Michael, 2020).

3.1 ASR

We use six different, readily available ASR models: 2 cloud-based and 4 local, chosen due to their respective results and accessibility. The cloud-based models are Google Cloud's Speech-to-Text and Microsoft Azure's Speech. We use Wav2Vec2 (W2V), DeepSpeech (DS), PocketSpinx (PS), and Vosk (Baevski et al., 2020; Hannun et al., 2014; Huggins-Daines et al., 2006).

Due to the limited amount of information given about the online ASR models, we can not go into depth about the architecture and training behind these models. The local models are summarized in Table 1 and described below.

Wav2Vec (W2V): We use Meta's Wav2Vec model from a checkpoint provided by HuggingFace where the model has been pre-trained and fine-turned on 960 hours of Librispeech (Baevski et al., 2020). This architecture is unique in that it is pre-trained on hours of unlabeled raw audio data. While other models first convert the audio into a spectrogram, Wav2Vec operates directly on audio data.[1]

DeepSpeech (DS): Mozilla's DeepSpeech model, is based on work done by Hannun et al. (2014). This architecture uses Recurrent Neural Networks that operate on spectrograms of the audio to make predictions. We use the 0.9.3 model and scorer for predictions. This model was trained using a wider variety of data from Fisher, LibriSpeech, Switchboard, Common Voice English, and approximately 1,700 hours of transcribed WAMU (NPR) radio shows explicitly licensed to them to be used as training corpora.[2]

PocketSphinx (PS): One of the lighter ASRs we tested is CMU's PocketSphinx (Huggins-Daines et al., 2006). PS is a light-weight ASR that is a part of the open source speech recognition tool kit called the CMUSphinx Project. This model was trained on 1,600 utterances from the RM-1 speaker-independent training corpus. Unlike the previously mentioned models, PS does not use neural networks and is instead based on traditional methods of speech recognition by using hidden Markov models, language models, and phonetic dictionaries.[3]

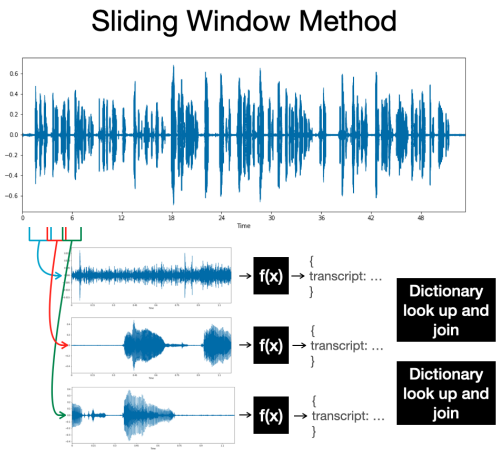

Figure 2: In the Sliding Window method, the ASR model makes predictions on partially overlapping portions of audio. Dictionaries are used to join the incoming predictions together.

Vosk: Alpha Cephei's Vosk (with the vosk-model-en-us-0.22 model) is built on top of Kaldi (Povey et al., 2011), and like PocketSphinx, uses an acoustic model, language model, and phonetic dictionary. Vosk uses a neural network for the acoustic part of the model.[4]

3.1.1 ASR Metrics

As mentioned, all previously proposed metrics for evaluating incremental ASR can be divided into three broad categories: overall accuracy (using WER), speed, and stability. In this section, we describe the specific metrics used and introduce our new metric which combines these last two categories of speed and stability into a single metric.

Overall Accuracy: WER Although there are different metrics to measure overall accuracy as compared in (Morris et al., 2004), we only use the most common metric, Word Error Rate (WER), which is defined by the the number of edits, substitutions (S), insertions(I), and deletions (D), divided by the total number of words (N): $WER = \frac{S+I+D}{N}$.

Predictive Speed: Latency In order to measure the general speed of an ASR model, we measure the time it takes from the time the ASR model gets the

[1] https://huggingface.co/facebook/wav2vec2-base-960h
[2] https://deepspeech.readthedocs.io/en/r0.9/
[3] https://github.com/cmusphinx/pocketsphinx-python
[4] https://alphacephei.com/vosk/

Name (abbreviation)	Model	Training Data
Wav2Vec (W2V)	wav2vec2-base-960h	LibriSpeech
DeepSpeech (DS)	0.9.3	Fisher, LibriSpeech, Switchboard, Common Voice English
PocketSphinx (PS)	N/A	1600 utterances from the RM-1
Vosk	en-us-0.22	N/A

Table 1: Local ASR models along with their used models and training data if available.

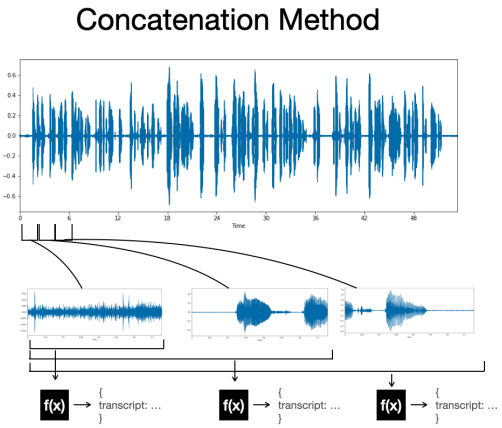

Figure 3: In this method, the incremental audio is concatenated together, and a prediction is made on the entire audio that has been given up to that point.

audio until the prediction is made. We then take this time and divide by the number of words in that particular prediction. With this, we define latency as the average amount of time per word it takes an ASR model to make a prediction: $LAT = \frac{Time}{N}$, where time is measured in seconds and N is the total number words in a given prediction.

Stability: Edit Overhead For measuring stability, we measure the edit overhead (EO). EO is the total number of revokes (R) divided by the total number of edits, or additions (A) and revokes (R), that the ASR model makes. $EO = \frac{R}{A+R}$.

Revokes per Second Our proposed and final metric is the number of *Revokes per Second* (RPS). We propose this metric as a way to capture the relationship between both speed and stability in an interpretable fashion by measuring how often an ASR changes its predictions. In an incremental SDS setting, this is the average number IUs that are labeled as type `revoke` per second. In an online meeting where real-time subtitles are available, this would represent the number of times you could expect a word to change per second in the transcript.

In such settings, a high RPS in a model's incremental predictions could result in confusion in downstream modules in an SDS setting (such the NLU module) or in humans trying to follow an online meeting using the real-time transcript.

We also look at the inverse *Seconds per Revoke* (SPR) as a simple adjustment to this metric to see how many seconds will pass by before one can expect to see a revoke. This SPR value is useful in interpretations when the RPS is low. Taken together, the formulas for these metric are as follows: $RPS = \frac{R}{Time(s)}$ and $SPR = \frac{Time(s)}{R} = \frac{1}{RPS}$

Combining Sub-word Output Both Google and Azure offer incremental ASR results. For these two ASRs, the audio files are sent to the cloud services in chunks, and the service returns a prediction with other meta-information. Google and Azure ASRs handle the concatenation, combining the predictions into a string that grows as the utterance unfolds. For local ASR models, we have control over how the predictions are combined and processed. We apply and compare two methods in this evaluation: Sliding Window and Concatenation.[5]

One limitation of many ASR models is the amount of audio they can process. For longer audio files (> 30 seconds), ASR models will start slow down and even crash. For this reason, we experiment with a sliding window of audio. For this Sliding Window method, we pass the audio from the file in chunks that are a bit longer than one second. These are then concatenated together as an audio buffer and given to an ASR model until it produces a prediction of at least 5 words or when it is indicated that it is the end of that particular audio file. Once a prediction of 5 words is made we remove the first 35% of the audio buffer. This results in a series of predictions on segments of audio. When a prediction is received, it is joined together with previous predictions. Due to overlap in incoming predictions, the way that the predictions are joined together is non-trivial. We used string filtering and matching functionalities to fil-

[5]We used the same PC with a GTX1080TI GPU for the local models.

ter out noise and join predictions appropriately by finding the overlapping string using dictionaries from WordNet and NLTK (Miller, 1995; Bird et al., 2009).

In the audio datasets we use, generally the files are short. Therefore, as a comparison we also implement a more simple Concatenation Method. For the Concatenation method, we present the audio in chunks into an audio buffer in the same manner as the Sliding Window method, except the audio buffer is a concatenation of all the audio (i.e., no audio ever gets removed from the buffer). Essentially, with this method, the ASR model makes a prediction from the very beginning of the file to the most recent audio given to the buffer. This is computationally more expensive and takes more memory because the ASR model has to make predictions on longer pieces of audio as time goes on, but this method eliminates the need for string matching between overlapping predictions. Diagrams showing these two methods can be seen in Figures 2 and 3. We compare these two methods as part of our evaluation.

3.2 NLU

RASA is a NLU framework that is made up of components that work in a sequential pipeline. In RASA, at least three components are usually required: a tokenizer which splits inputs into smaller tokens (usually words), a featurizer that maps words into a vector, and a classifier that maps from vectors to slots, but others can be included.

The output of this classifier becomes a *meaning representation*, which is a semantic *frame* made up of *slots*, with an overarching *intent*. The example below shows how the utterance *I would like a flight from Boston to Berlin* is represented as a semantic frame made up of 3 slots, one being the intent:

`intent`	flight
`source`	Boston
`target`	Berlin

The dialogue designer determines the slot names based on the domain, e.g., `source` for departure airport and `target` for destination airport.

Instead of making each of the individual components in RASA work incrementally, we follow Khouzaimi et al. (2014) by inserting an incremental manager component at the beginning of the RASA pipeline that allows word-level IUs (i.e. word and IU operation type) to be used as input.

Figure 4 shows a typical minimal pipeline for RASA with our incremental manager component added to enable the entire pipeline to process with word-level IUs. This incremental manager component maintains the unfolding utterance by adding each new word (i.e., from an incremental ASR) to a growing utterance prefix, or an incremental cache, that is re-processed at each word (revoked words are removed from the prefix, as needed).[6]

For example, the utterance *from Boston to Berlin* as part of an ongoing dialogue about booking flights is processed word-by-word, but RASA processes each prefix as a separate utterance:

from			
from	Boston		
from	Boston	to	
from	Boston	to	Berlin

Another challenge to incrementalizing RASA is only outputting new updates to the NLU frame. For example, at each word in the above utterance, RASA should only produce the `source:Boston` slot of the frame when the words *from Boston* are uttered, and not again even though the prefix is being reused at each increment. Likewise, the slot `target:Berlin` should only be produced once when the relevant words *to Berlin* are uttered.

For evaluating NLU, we use accuracy and F1 scores at each word increment.

4 Experiment 1: Incremental Evaluation of ASR Models

Data To evaluate, we use datasets from two different domains: LibriSpeech and a recently assembled dialogue dataset of simulated medical conversations (Fareez et al., 2022).[7] The LibriSpeech test-clean dataset contains 5.4 hours of speech from 40 different speakers, 20 male and 20 female. This audio is divided into over 2,600 files with an average of about 20 words per file containing a vocabulary of over 8,100 words. To ensure the audio would work on all of our models, we converted the audio files to WAV files.

[6]This version of incremental processing is called *restart* incremental because it resets the internal model at each word increment. More ideally, we would use a model that could maintain its internal state and work incrementally (known as *update* incremental), but recent language understanding models are not amenable to word-level processing.

[7]We were unable to obtain the Switchboard corpus due to prohibitive costs.

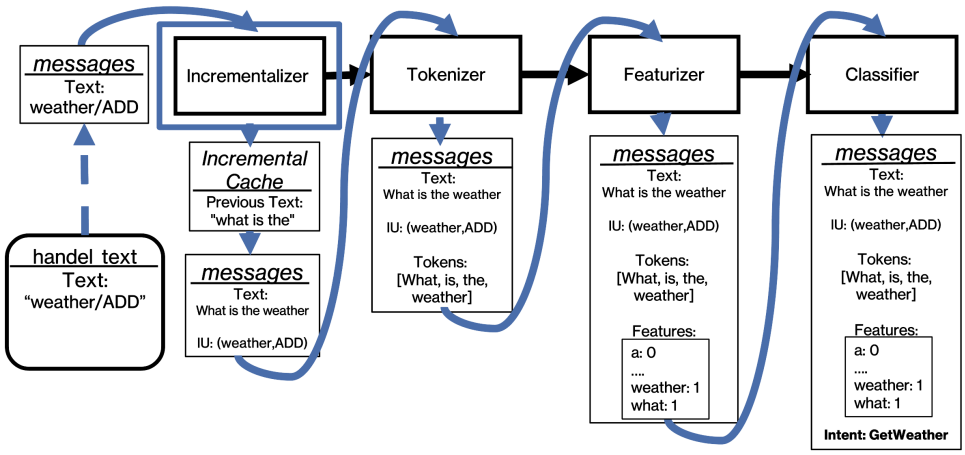

Figure 4: Adapted from rasa.com: our Incrementalizer component at the beginning of any pipeline allows the entire pipeline to process at the word level by managing and caching relevant incremental information.

The medical conversation dataset contains 272 audio files with corresponding transcripts. The purpose of using this dialogue data is 1) to test each model on domain data that presumably none of them have been trained on (since this dataset was just made public in 2022), and 2) to test how each model performs on a dialogue dataset that contains disfluencies such as fillers, corrections, and restarts.

The audio files range from around 7 to 20 minutes in length or about 800 to 2,200 words. Due to the size of these audio files, we split up the files into utterances based on silence and then randomly sample a set of 40 utterances, 17 of which were able to be processed by all 6 ASR models (max 40 seconds, min 0.8 seconds, 6.1 seconds in length on average) due to the length of some of the utterances and the constraints that each model can handle.

Results The results are shown Table 2. When using the Sliding Window method, local models had lower latency (i.e. faster) than both the online models. Some of the local ASR models using the Concatenation method were faster than both of the online ones. However, tests using the Concatenation method was slower and had a higher EO than the Sliding Window method given the same ASR.

Although slower and less stable (as measured by EO), the Concatenated versions performed better than their corresponding Sliding Window version in overall all accuracy or WER. This makes sense as the Concatenation method has access to the entire context to make predictions where as the Sliding Window has only a small portion of the context. Comparing the online models, Google is less accurate and more revoke dependant than Azure. However, Google is considerably quicker which could be crucial in an interactive dialogue setting. The cloud models had surprisingly low latency (though the latency is dependent on the internet speed), but the local ASRs tended to have the lowest latency.

The local ASR model which performed the best in terms of WER was the W2V model using the Concatenation method on the LibriSpeech data and Vosk on the Medical Dialogue data, while the model with the lowest Edit Overhead was the DS model using the Sliding Window method. Though a low WER is generally better, the number of revokes has implications for downstream modules in an SDS; keeping the EO low and Revokes per Second low with a low WER means the model was correct early, which is ideal.

Our results are consistent with previous evaluations on Incremental ASR (Baumann et al., 2016) that show that Google's ASR predictions, although fairly accurate overall, are not as stable as the others, with the highest Edit Overhead of 0.279/0.228 and an average of about 4.5/5.1 Revokes per Second on the LibriSpeech dataset and Medical Dialogue dataset respectively.

The DS and Vosk models' WERs were higher than some of the other models, but the low EO and infrequent number of revokes make them potentially good candidates for an SDS that requires high accuracy as well as low latency and EO, for example in a robotic platform. We suggest Concatenation for live microphones, with voice activity detection or with certain models chunking,[8] to pre-

[8] https://huggingface.co/blog/asr-chunking

Incremental ASR Results on LibriSpeech										
	Google	Azure	W2V	W2V (Con.)	DS	DS (Con.)	PS	PS (Con.)	Vosk	Vosk (Con.)
WER	13.2	9.1	10.6	**4.0**	18.3	8.4	*40.4*	31.8	33.4	6.4
LAT	0.197	0.539	**0.099**	0.127	0.181	*1.443*	0.105	0.220	0.104	0.167
EO	*0.279*	0.065	0.011	0.093	**0.001**	0.013	0.014	0.147	0.072	0.019
R/Sec	*4.564*	0.679	0.141	1.919	**0.008**	0.012	0.178	1.688	0.910	0.143
Sec/R	*0.219*	1.473	7.083	0.521	**123.135**	80.489	5.613	0.593	1.099	7.004

Incremental ASR Results on Medical Dialogue Dataset										
	Google	Azure	W2V	W2V (Con.)	DS	DS (Con.)	PS	PS (Con.)	Vosk	Vosk (Con.)
WER	41.1	**21.0**	47.8	42.3	42.5	38.7	*85.6*	80.0	38.4	23.2
LAT	0.287	0.623	**0.125**	0.217	0.245	*1.452*	0.131	0.394	0.307	1.296
EO	*0.243*	0.055	0.016	0.211	**0.000**	0.014	0.005	0.240	0.048	0.025
R/Sec	*5.944*	0.207	0.253	6.376	**0.000**	0.013	0.046	2.447	0.215	0.079
Sec/R	*0.168*	4.837	3.953	0.157	**inf**	75.616	21.734	0.409	4.649	12.733

Table 2: Summary of results. Local ASRs had lower latency than cloud-based ASRs. The Concatenation method, shown in the columns that contain a *(Con.)*, had higher latency and resulted in a higher EO and RPS, but not as many revokes as the online ASRs. *inf* means zero revokes per second.

vent running out of memory because it is more accurate and does not require string matching.

5 Experiment 2: Evaluation of Incremental RASA

In this section, we explain our experiment to evaluate our incremental version of RASA NLU.

Task & Procedure For this experiment, we were restricted to only use datasets that contain audio, text transcriptions, and annotated frames. Since SNIPS (Coucke et al., 2018) and SLURP (Bastianelli et al., 2020b) datasets have these requirements, they are used for evaluation in this experiment. Both datasets have speech (though in the case of SNIPS, the audio is synthesized). We compare incremental results using oracle (i.e., hand-transcribed) speech from the two datasets as well as ASR output from Google ASR (which had good WER in Experiment 1 and has low latency, but high edit overhead which is desired so RASA has a chance to handle revokes).

The SNIPS dataset contains 14,484 entries with seven categories of intents evenly distributed. There is an average of 9 words per utterance with 66,500 entity annotations with the largest entity representing 8.2% of the annotations. The SLURP dataset consists of 14,488 utterances with 18 intents unevenly distributed with the largest intent representing 14.4% of the data. There is an average of 7 words per utterance with 21,662 entity annotations with he largest entity representing 14.9% of the annotations.

Metrics & Baseline We calculate F1 score of the intent and slots (termed *entities* in RASA; a false positive is when a slot is filled erroneously, and a false negative is when a slot is unfilled). The F1 score at the end of an utterance is the highest possible because at that point it has received all of the audio information. To show how well RASA works incrementally, we show F1 score at the end of the utterance along with show how the F1 score is affected when incrementally removing up to 7 frames/words before the end of the utterance.

We report the F1 score for both hand-transcriptions and ASR output for both datasets. For intent detection, the majority classifier baseline for SNIPS is 14.3% and for SLURPS, 14.4% Similarly the majority classifier baseline for entity detection for SNIPS is 8.2% and for SLURPS entities, is 14.9%. Here we are not attempting to evaluate the RASA model to achieve state-of-the-art results, rather we are trying to evaluate the potential for RASA as an incremental NLU.

Results Figure 5 shows the results for this experiment. Naturally, the closer the utterance was to the end of the utterance, the higher the F1 score. Furthermore, the F1 scores for intent (a single classification of the entire utterance) was higher than the F1 scores for the entities. This too is expected as the single classification task of detecting intent from either seven (SNIPS) or 18 intents (SLURPS), is much simpler than detecting multiple entities from a wider range of categories.

Results moreover show that even when the transcripts are not 100% correct (i.e. come from an ASR, the solid blue line in figure 5), RASA can achieve a higher F1 score than the majority classifier as early as 6 words out for the more difficult task of entities detection. For intent detection, RASA performs significantly better than the

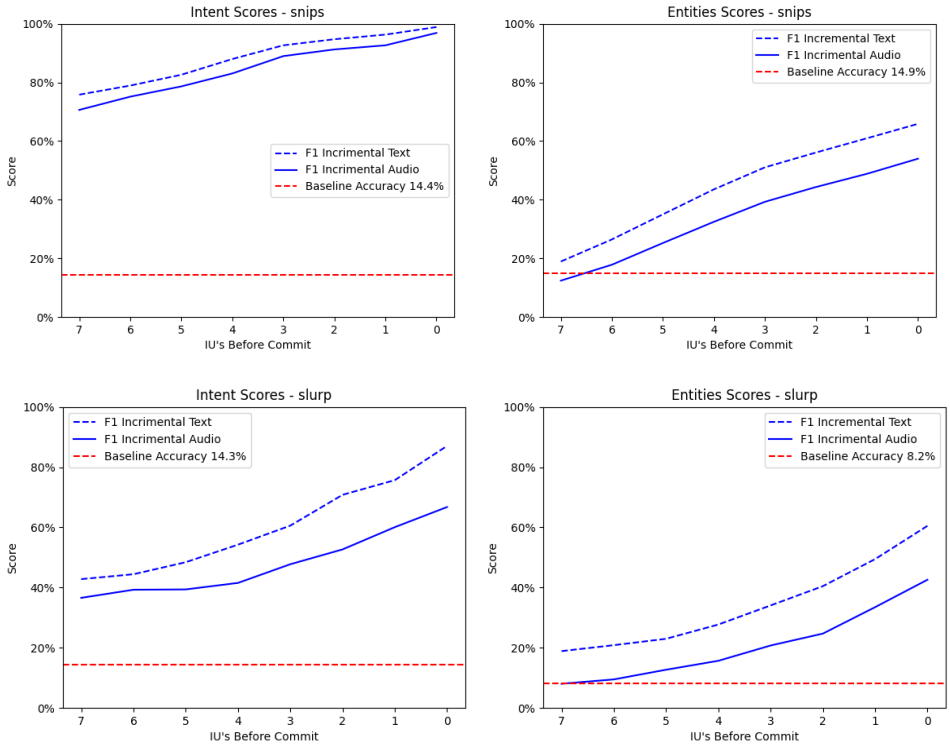

Figure 5: TOP: Incremental results on the SNIPS dataset: transcriptions and ASR. The y-axis is the F1 score, the x-axis is the number of words before the end of the utterance (i.e., before commit). BOTTOM: Incremental results on the SLURP dataset: transcriptions and ASR. The y-axis is the F1 score, the x-axis is the number of words before the end of the utterance.

baseline very quickly into an utterance and on the SNIPS dataset, with imperfect transcripts from the ASR, RASA achieves an F1 score over 80% as early as three words before the end of the utterance.

This evaluation shows the potential for RASA to be used effectively in an incremental setting, allowing a system that uses this incremental setup to be able to make decisions, start acting, or formulating queries before the end of an utterance. This is agreement with with Manuvinakurike et al. (2018) who showed that incremental NLU can be more efficient. For example, an utterance such as *go to the right to pick up...*, a robot could start moving in a predicted direction before the robot even 'knows' that it is to *pick up* and before it 'knows' what to pick up. In the setting of booking an airline or flight, the words *I would like to book...*, the SDS could already begin to start formulating the query to check availability before the end of the sentence.

6 Conclusion

In this work, we tested six different ASR models and RASA for NLU in an incremental setting and we proposed a new metric for incremental ASR, Revokes per Second as an informative addition to existing incremental metrics. We showed that, generally, as might be expected, online ASR (in our evaluation, Google Cloud and Azure cloud services) is not as fast as most of the local ASR models tested, and while the online ASRs are some of the most accurate ASRs we tested, they both have a relatively high number of Revokes per Second and Edit Overhead which, in combination with the latency, could potentially lead to more issues in an incremental setting because high edit rates could require unnecessary processing. Our results are informative as to the *out of the box* performance. Furthermore, we also believe that our proposed metric, Revokes per Second, is an interpretable useful metric that should be used as ASR becomes more prevalent in *live* settings such as in Spoken Dialogue Systems on robots or in live captioning in online meetings.

For NLU, we showed that RASA can work well incrementally, offering designers and users earlier than end-of-utterance predictions of user utterances. This will enable systems to have the option to make earlier decisions and actions, and our changes will

be beneficial for long-term maintenance.

Each of the modules described in this paper are implemented in Retico and will be made public.

Acknowledgments

We deeply appreciate the useful feedback from the anonymous reviewers.

References

Gregory Aist, James Allen, Ellen Campana, Carlos Gomez Gallo, Scott Stoness, and Mary Swift. 2007. Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods. In *Pragmatics*, volume 1, pages 149–154, Trento, Italy.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020a. SLURP: A Spoken Language Understanding Resource Package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020b. SLURP: A spoken language understanding resource package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7252–7262. Association for Computational Linguistics.

Timo Baumann, Michaela Atterer, and David Schlangen. 2009. Assessing and improving the performance of speech recognition for incremental systems. In *Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics*, pages 380–388.

Timo Baumann, Casey Redd Kennington, J. Hough, and David Schlangen. 2016. Recognising conversational speech: What an incremental asr should do for a dialogue system and how to get there. In *IWSDS*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *Proceedings of the 31st Conference on Neural Information Processing Systems*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, pages 12–16.

Jens Edlund, Joakim Gustafson, Mattias Heldner, and Anna Hjalmarsson. 2008. Towards human-like spoken dialogue systems. *Speech Commun.*, 50(8-9):630–645.

Faiha Fareez, Tishya Parikh, Christopher Wavell, Saba Shahab, Meghan Chevalier, Scott Good, Isabella De Blasi, Rafik Rhouma, Christopher McMahon, Jean-Paul Lam, et al. 2022. A dataset of simulated patient-physician medical interviews with a focus on respiratory cases. *Scientific Data*, 9(1):1–7.

Kallirroi Georgila, Anton Leuski, Volodymyr Yanov, and David Traum. 2020. Evaluation of off-the-shelf speech recognizers across diverse dialogue domains. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6469–6476, Marseille, France. European Language Resources Association.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alexander I Rudnicky. 2006. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

Casey Kennington, Daniele Moro, Lucas Marchand, Jake Carns, and David McNeill. 2020. rrSDS: Towards a robot-ready spoken dialogue system. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 132–135, 1st virtual meeting. Association for Computational Linguistics.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefevre. 2014. An easy method to make dialogue systems incremental. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 98–107, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *arXiv*.

Ramesh Manuvinakurike, Trung Bui, Walter Chang, and Kallirroi Georgila. 2018. Conversational image editing: Incremental intent identification in a new dialogue task. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*,

pages 284–295, Melbourne, Australia. Association for Computational Linguistics.

Thilo Michael. 2020. Retico: An incremental framework for spoken dialogue systems. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 49–52.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Fabrizio Morbini, Kartik Audhkhasi, Kenji Sagae, and Ron Artstein. 2013. Which ASR should I choose for my dialogue system? In *Proceedings of the SIGDIAL 2013 Conference*, pages 394–403, Metz, France. Association for Computational Linguistics.

Andrew Cameron Morris, Viktoria Maier, and Phil Green. 2004. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *Eighth International Conference on Spoken Language Processing*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Andrew Rafla and Casey Kennington. 2019. Incrementalizing RASA's Open-Source natural language understanding pipeline. *arXiv*.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 710–718, Athens, Greece. Association for Computational Linguistics.

Michael K Tanenhaus and Michael J Spivey-Knowlton. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632.

Dong Yu and Li Deng. 2016. *Automatic speech recognition*, volume 1. Springer.