Real-time ASR Customization via Hypotheses Re-ordering: A Comparative Study of Different Scoring Functions

Anonymous ACL submission

Abstract

General purpose automatic speech recognizers (ASRs) require customization to the domain and context, to achieve practically acceptable accuracy levels when used as part of voice digital assistants. Further, such general purpose ASRs typically output multiple alternative hypotheses for the same input utterance. In this paper, we consider the hypothesis re-ordering framework and evaluate the impact of three different scoring functions for re-ordering the hypotheses: phoneme-based, character-based and word-based, and determine their strengths and weaknesses. Based on our intuitions and experimental validation, we determine that phoneme-based scoring is the best for closed domain contexts, while character-based and word-based scoring do better in case of more open-domain contexts. Our results show that character-based scoring gives the best performance improvement in terms of word error rate over general purpose ASRs for voice assistants used in a classroom context. Our analysis also reveals that character-based scoring is preferred for shorter utterances while word-based scoring is preferred for longer utterances.

1 Introduction

The recent success of voice as an interaction modality, ushered in by voice digital assistants (Amazon; Google; Apple; Tulshan and Dhage, 2018) promises to revolutionize both consumer and enterprise space. The use of voice (particularly while being hands-free or walking in far-field scenarios) as a substitute for typing text for diverse use cases, ranging from fixed-command-control to information retrieval of more open-ended concepts in a variety of domains (Brill et al., 2019), can transform the user experience and bring efficiency to human-machine interactions. At the same time, transcribing voice commands using general purpose automatic speech recognition (ASR) (Haeb-Umbach et al., 2020; Jefferson, 2019) as part of a digital assistant to accurately represent the intent of the user is critical to ensure that user experience is friction-free; errors in recognized transcriptions could lead to an unsuccessful task, resulting in a disappointed user reverting to typing.

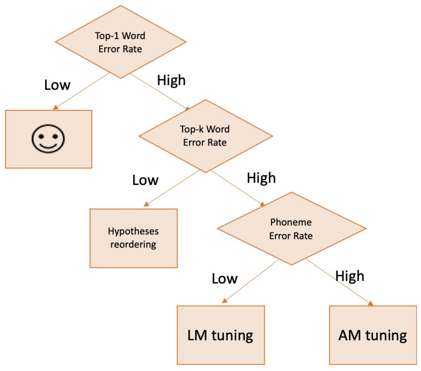

Figure 1: Solution Design Space for ASR Accuracy

Achieving high accuracy of ASR by voice assistants in different domains is challenging due to several reasons (Howe and Yampolskiy). Few reasons include ambiguity introduced by homophones within a language, speech decoding biased by training data used in language modeling, different accents relative to acoustic training data, and closeness of certain concepts to other languages. Consequently, contextual inference of users' intents and domain-specific entities in the utterances is critical, and often better handled as a post-ASR step. While certain use cases such as in-car and in-home device control can be more easily handled by fixed vocabulary language model training and force-fitting to the well-known command set, use cases involving domain specific entities and broad vocabulary need more sophisticated post-processing, the latter of which is the focus of this paper.

Problem Description: A typical ASR takes speech data as input, uses one or more pre-trained acoustic (AM) and language (LM) models, and outputs one or more alternatives (hypotheses) of text as the possible utterances. The accuracy of the recognition process depends on the fidelity of the input, the sophistication of the AM as well as the LM models, and/or the decision process involved in selecting the final text among the hypotheses. When tuning the speech recognition process for any domain, all these steps play an important role towards achieving optimal performance. Fig. 1 illustrates the importance of each step for different types of speech utterances transcribed by a given ASR. This figure shows that the ideal scenario has the top-1 (i.e. the top hypothesis identified by the ASR) word error rate (WER) (Ali and Renals, 2018) to be low, which happens when the ASR is properly tuned for the domain language and use case. If top-1 WER is high, we consider looking at the top-k WER (minimum WER among WERs between ground-truth transcript and each of the top-k hypotheses). If this top-k WER is low, we can employ a hypotheses re-ordering algorithm (Variani et al., 2020) to rearrange the top-1 hypothesis in order to get a low top-1 WER. On the other hand, if top-k WER is also high, it implies that ASR needs tuning as none of its hypothesis can capture the ground truth. In such a case, we consider phoneme error rate (PER) (Kessler, 2005) instead of WER. If top-k PER is low, it implies that the AM of the speech recognizer is decoding phonemes correctly but the LM needs further tuning. And the final case to consider is when the top-k PER is also high, implying that the AM of the ASR system itself needs tuning to improve ASR performance. With increasing sophistication of off-the-shelf ASR systems due to the large amount of usage data collected and manually annotated, we assume that both AM and LM training already do their best to provide the alternative hypotheses. The major objective in this paper is to propose different scoring functions for re-ordering these alternative hypotheses considering the low top-k WER and high top-1 WER regime.

Contributions: Our major contributions are: (a) Given a speech utterance, one or more hypotheses of alternative text, and the usage context (defined in detail later), we define three ways of computing the relative distance between the hypotheses and context; a heuristic is then derived for selection and re-ordering of hypotheses in real-time for every new utterance. (b) Experiments on a real dataset of utterances in the educational setting show that character-based edit distance can lead to the best performance by minimizing WER compared to other scoring functions. (c) We also show that WER is minimized if character-based edit distance is used when the *average word length* of the hypotheses set for an utterance is <2, and word-based edit distance is used when average length ≥ 2.

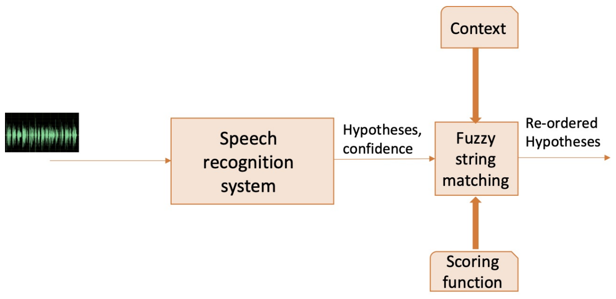

Figure 2: Generic hypotheses re-ordering framework

The rest of the paper is organized as follows. Sec. 2 discusses related work. Sec. 3 discusses the high level hypotheses re-ordering approach. Sec. 4 describes the scoring functions and presents our experimental results. Finally, we conclude in Sec. 5.

2 Related Work

The ability of humans to improve the accuracy of speech recognizers by re-ordering the outcome of these systems (Lippmann, 1997) has inspired speech recognition communities for decades to work on improving ASR's performance by hypotheses re-ordering. Early attempts employed linear regression (Chotimongkol and Rudnicky, 2001), discriminative language models (Roark et al., 2004, 2007), support vector machines (Stuhlsatz et al., 2006) and memory-based learning (Jonson, 2006) to address the re-ordering problem through post-processing. Later, (Lojka and Juhár, 2014; Soto et al., 2016) combined multiple ASR systems by using confidence scores of words or employing word embedding based machine learning approaches to improve speech recognition accuracy. With the rise of deep neural networks in the past decade, recurrent neural network based language models (Mikolov et al., 2010; Chetupalli and Ganapathy, 2020; Erdogan et al., 2016) have become more popular for hypotheses re-ordering. Further, an encoder-classifier model was developed in (Ogawa et al., 2018) that compares the pairs in top-k lists to do re-ordering while (Apte et al., 2021) proposed a re-ordering solution by retrieving the most likely candidate correction to abandoned utterances in the voice search query domain. Besides improving accuracy in terms of WER only, there has been research on other aspects such as intent ranking based on domain-specific contextual models (Anantha et al., 2020; Corona et al., 2017). Nevertheless, reducing WER remains of great interest in the field of speech recognition. In this paper, we aim at comparing different scoring functions that can customize ASR outputs via hypotheses re-ordering and find applicability in real-time scenarios which is a major limitation of existing literature.

3 Hypotheses re-ordering

Typical ASRs take speech data as input and provide multiple alternative transcriptions (hypotheses) along with their associated confidence values, based on a generic language model of the ASR. These may lack the specific context in which the user has uttered the command. Dialogue systems of assistants that have context would want to utilize that information to choose the appropriate hypothesis from the list of hypotheses provided as output by the ASR. To this end, we describe a generic algorithm that re-orders the hypotheses based on the domain context. Consider, for instance, a digital assistant that enables retrieval of information for questions of the form: “Who is <Abraham Lincoln>" or “Where is the <Eiffel Tower>". In such a system, the context is provided as a list of expected utterances in a regular expression format, such as “Who is _entity_" (“_entity_" is a placeholder and can be any open-ended term). The generic algorithm for hypothesis re-ordering in such a system is presented as a block diagram in Fig. 2.

For every utterance, the ASR outputs multiple hypotheses with their respective confidence values as shown in Fig. 2. Each hypothesis is then compared with every command in the command list provided as context, and the *closest* match as defined by the scoring function, is determined. This scoring function also quantifies how *close* this closest command is to the particular hypothesis. This triplet information (hypothesis, closest command, closeness score) is stored for each of the hypotheses output by the ASR. The score is used to re-order the hypotheses, which is the output of the algorithm.

4 Experiments and results

In this section, we first describe the dataset we use for our experiments. Next, we define three different scoring functions (detailed discussions are provided in Appendix A) and discuss our results for hypotheses re-ordering using the scoring functions. We also show the quantitative and qualitative impact of the three scoring functions.

4.1 Dataset

For our study, we focus on a use-case of a voice assistant being used in an educational setting. This is a specific domain where generic ASRs do not give the best performance and fail for some simple domain-specific utterances. We collected data from multiple users using 144 different voice-enabled personal assistant devices (and speakers). These users (who are teachers) have uttered commands that are pertinent to educational activities such as “go to the link on wikipedia", “next slide", “show me videos of the ocean" etc. over a period of 4 months as part of their daily classroom activities. Note that this data was collected while the device was used in a production scenario as part of a pilot program. We have annotated 13754 utterances from this collected data whose ground truth was determined by listening to the audio files. These details are given in Table 1.

These utterances are generally not a part of the default language model of a generic ASR, and hence would not be the top hypothesis output by the ASR. As mentioned earlier, this is the scenario where hypothesis re-ordering is expected to give a performance improvement. For hypothesis re-ordering, the context is extremely important. For this purpose, we used a context consisting of expected commands in the regular expression format (refer to Sec. 3), which cover a wide range of phrases, and not specific to only our dataset. The collected utterances were sent through a cloud speech to text engine (Manaswi, 2018) along with a request to receive up to 10 hypotheses as output wherever available. We also used an appropriate speech adaptation[1] to improve the default performance. This resulted in a top-1 WER of 13.11%, which is reasonable for a generic ASR (see Table 2). However, for practical purposes, an improved performance can be achieved using our proposed hypotheses re-ordering algorithm (refer to Fig. 2). Some examples of when the ASR's default top-1 transcription failed are provided below.

Ground truth	ASR top-1 transcript
make teams	make teens
find dog images	blind dog images
pair my laptop	clear my laptop

4.2 Scoring functions

In this section, we focus on defining three different scoring functions and discuss their impact on improving the hypothesis re-ordering algorithm. These scoring functions help to capture the differences between two phrases in terms of syntactic/phonetic features, which can aid in more intelligently using context for hypothesis re-ordering.

[1]https://cloud.google.com/speech-to-text/docs/speech-adaptation

Devices	Utterances	Average length of utterances	Unique commands
144	13754	3.327	3346

Table 1: Dataset statistics

Dataset size	Default	Word-based	Char-based	Phoneme-based
13754	13.11	10.57	**9.86**	11.05

Table 2: WERs for different scoring functions

4.2.1 Phoneme-based edit distance

Here we define the phoneme-based edit distance that compares two phrases on how they *sound* i.e. edit distance between their phonemic transcriptions. As seen in Table 2, the WER drops to 11.05% when such a scoring function is used to re-order the hypotheses. Some examples where such a scoring function helped are shown below, along with examples where it did not help. As can be seen, the failures are in cases where the ground truth and the re-ordered top-1 hypothesis are homonyms.

Ground truth	ASR top-1	Re-ordered transcript
pair my laptop	send my laptop	pair my laptop
open globe	open gloves	open globe
draw a line	draw alignment	draw a lion
set transcriber to two	set transcriber 2:2	set transcriber to

4.2.2 Word-based edit distance

Next we consider the word-based edit distance which is defined as the number of edit operations (add, delete, replace) on words needed to *transform* one phrase into the other. As observed in Table 2, the WER drops to 10.57% when such a scoring function is used. Some examples where such a scoring function helped is shown below along with examples where it did not help. As seen, the major errors are on the mistranscription of the entities which are not addressed by the hypothesis re-ordering algorithm.

Ground truth	ASR top-1	Re-ordered Transcript
close this tab	what was this tab	close this tab
zoom in	zumen	zoom in
show videos of oceans	show videos of portions	show videos of portions

4.2.3 Character-based edit distance

Finally, we define the character-based edit distance as a measure of the similarity between two strings (phrases) at the character level. As seen in Table 2, the WER is 9.86% when such a scoring function is used for hypothesis re-ordering, providing the best performance improvement among the three scoring functions. This is because the character-based score helps to capture the partial overlap between similar words in the two phrases which the word-based score fails to capture. The following are some examples where such a scoring function helped along with some examples where it did not. As seen in the examples, this scoring function works really well for short utterances of length <2.

Ground truth	ASR top-1	Re-ordered Transcript
next	make	next
reduce	radius	reduce
go home	Bill home	go home
pick a class	he got class	he took a class

4.3 Discussion

Experiments on our dataset helped in revealing the strengths and weaknesses of each of the three scoring functions. For our use-case, Table 2 showed that the word-based and character-based scoring functions are more appropriate with lower WER than the phoneme-based scoring function. This is because phoneme-based scoring works best when the context and commands are exact commands with no expected variations (eg. presence of additional words in the utterance such as articles), like the ones used in command-control use-cases. Further, phoneme-based scoring also fails when there are homonyms (eg. “set transcriber to two" vs. “set transcriber to to"). We also observed that the character-based scoring function performs better when the utterance is shorter (<2 words) while the word-based one works better for longer utterances.

5 Conclusion

We studied the general framework of using distance based scoring for ASR hypotheses re-ordering. Such a framework is extremely helpful in cases where a generic ASR is to be used in a constrained context/use-case. Our study analysed the role of three different distance functions in improving hypothesis re-ordering. We determined the strengths and weaknesses of all distance functions and determined when each of them is most suitable for use. We observed that character-based scoring function performed the best achieving a minimum WER of 9.86%. Our analysis also showed that character-based scoring function is most appropriate for short utterances and word-based scoring function is best suited for longer utterances. We discussed the ethical impact of our work in Sec. A.3 of Appendix A.

References

Ahmed Ali and Steve Renals. 2018. Word error rate estimation for speech recognition: e-wer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 20–24.

Amazon. Alexa. `https://developer.amazon.com/en-US/alexa`.

Raviteja Anantha, Srinivas Chappidi, and William Dawoodi. 2020. Learning to rank intents in voice assistants. *arXiv preprint arXiv:2005.00119*.

Apple. Siri. `https://www.apple.com/siri/`.

Ajit Apte, Allen Wu, Ambarish Jash, Amol H Wankhede, Ankit Kumar, Ayooluwakunmi Jeje, Dima Kuzmin, Ellie Ka In Chio, Harry Fung, Heng-Tze Cheng, et al. 2021. Mondegreen: A post-processing solution to speech recognition error correction for voice search queries.

Thomas M Brill, Laura Munoz, and Richard J Miller. 2019. Siri, alexa, and other digital assistants: a study of customer satisfaction with artificial intelligence applications. *Journal of Marketing Management*, 35(15-16):1401–1436.

Srikanth Raj Chetupalli and Sriram Ganapathy. 2020. Context dependent rnnlm for automatic transcription of conversations. *arXiv preprint arXiv:2008.03517*.

Ananlada Chotimongkol and Alexander I Rudnicky. 2001. N-best speech hypotheses reordering using linear regression. In *Proc. Seventh European Conf. Speech Commun. Techn.*, pages 1829–1832.

Rodolfo Corona, Jesse Thomason, and Raymond Mooney. 2017. Improving black-box speech recognition using semantic parsing. In *Proc. Eighth Int. Joint Conf. Natural Language Process. (Volume 2: Short Papers)*, pages 122–127.

Hakan Erdogan, Tomoki Hayashi, John R Hershey, Takaaki Hori, Chiori Hori, Wei-Ning Hsu, Suyoun Kim, Jonathan Le Roux, Zhong Meng, and Shinji Watanabe. 2016. Multi-channel speech recognition: LSTMs all the way through. In *CHiME-4 workshop*, pages 1–4.

Google. Google assistant. `https://assistant.google.com`.

Reinhold Haeb-Umbach, Jahn Heymann, Lukas Drude, Shinji Watanabe, Marc Delcroix, and Tomohiro Nakatani. 2020. Far-field automatic speech recognition. *Proceedings of the IEEE*, 109(2):124–148.

William J Howe and Roman V Yampolskiy. Impossibility of unambiguous communication as a source of failure in ai systems.

Madeline Jefferson. 2019. Usability of automatic speech recognition systems for individuals with speech disorders: Past, present, future, and a proposed model.

Rebecca Jonson. 2006. Dialogue context-based re-ranking of ASR hypotheses. In *Proc. 2006 IEEE Spoken Language Technology Workshop*, pages 174–177.

Brett Kessler. 2005. Phonetic comparison algorithms. *Trans. Philological Society*, 103(2):243–260.

Richard P. Lippmann. 1997. Speech recognition by machines and humans. *Speech Communication*, 22(1):1 – 15.

Martin Lojka and Jozef Juhár. 2014. Hypothesis combination for slovak dictation speech recognition. In *Proc. Elmar - Int. Symp. Electronics Marine*, pages 43–46.

Navin Kumar Manaswi. 2018. Speech to text and vice versa. In *Deep Learning with Applications Using Python*, pages 127–144. Springer.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Atsunori Ogawa, Marc Delcroix, Shigeki Karita, and Tomohiro Nakatani. 2018. Rescoring n-best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model. In *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*, pages 6099–6103.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language*, 7(12):39–43.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech & Language*, 21(2):373–392.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. 42nd Annual Meeting Association Comput. Linguistics (ACL-04)*, pages 47–54.

V. Soto, O. Siohan, M. Elfeky, and P. Moreno. 2016. Selection and combination of hypotheses for dialectal speech recognition. In *Proc. IEEE Int. Conf. Acoustics Speech Signal Process. (ICASSP)*, pages 5845–5849.

A Stuhlsatz, M Katz, SE Krüger, HG Meier, and A Wendemuth. 2006. Support vector machines for postprocessing of speech recognition hypotheses. In *Proc. Int. Conf. Telecommun. Multimedia TEMU*.

Amrita S Tulshan and Sudhir Namdeorao Dhage. 2018. Survey on virtual assistant: Google assistant, siri, cortana, alexa. In *International symposium on signal processing and intelligent recognition systems*, pages 190–201. Springer.

Ehsan Variani, Tongzhou Chen, James Apfel, Bhuvana Ramabhadran, Seungji Lee, and Pedro Moreno. 2020. Neural oracle search on n-best hypotheses. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7824–7828. IEEE.

A Appendix

A.1 Scoring functions

The goal of the scoring functions is to capture the differences between hypotheses in terms of syntactic features or phonetic features, which can aid in more intelligently reordering the hypotheses using the context. In what follows, we discuss three such functions and the intuition behind using them.

A.1.1 Word-based edit distance

The first scoring function is based on the word-based edit distance defined as the number of edit operations (add, delete, replace) on words needed to *transform* one phrase into the other. This distance function (also known as Levenshtein distance (Navarro, 2001)), is motivated by the fact that it is heavily used in ASR systems to evaluate the word error rate (WER) and also in natural language systems to quantify the difference between two sentences/phrases.

In our implementation, this score between two phrases p and q is given as:

$$\mathbf{score_w}(p, q) = \frac{100}{\epsilon + \mathbf{WED}(p, q)}$$

where $\mathbf{WED}(p, q)$ is the word-edit distance between the two phrases and ϵ is a small value to ensure the score function is bounded when the phrases p and q are same. As an example, the score between the phrases p =“make four teams" and q =“make four groups" is $\mathbf{score_w}(p, q) = 90.91$ when $\epsilon = 0.1$.

In our use-case, since some phrases can have “_entity_" within the sentence, we accommodate it in the word edit distance calculation by treating it as a dummy word whose edit operation contributes nothing towards the edit distance. In other words, the adding/deleting of “_entity_" or replacing it to any other word is not considered as an edit operation.

A.1.2 Character-based edit distance

The character-based edit distance based score is a measure at the character level. It is a measure of

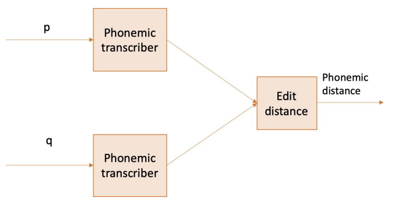

Figure 3: Block diagram of the phonemic distance calculation

the strings' similarity. For two strings p and q, in our implementation, this score is defined as:

$$\mathbf{score_c}(p, q) = \mathbf{round}\left(100 \times \frac{2\mathbf{M}(p, q)}{\mathbf{T}(p, q)}\right)$$

where $\mathbf{T}(p, q)$ is the total number of characters in both strings together, $\mathbf{M}(p, q)$ is the number of matches in the two strings, and $\mathbf{round}(\cdot)$ rounds to the nearest integer.

The character-based edit distance function more precisely quantifies the difference between phrases, especially when the phrases are small. For example, if p =“pause" and q =“cause", the word-based edit distance would determine only 1 word is different but cannot capture the fact that most of the word is similar. In this case, it is valuable to identify the partial overlap that is present in the two phrases. The character-based score allows us to capture this information. For the example of p =“pause" and q =“cause", $\mathbf{T}(p, q) = 10$ and $\mathbf{M}(p, q) = 4$, and hence the $\mathbf{score_c}(p, q) = 80$. To accommodate “_entity_" in our phrases, we remove it from our phrase before calculating the character-based score.

A.1.3 Phoneme-based edit distance

The third scoring function that we use is phoneme-based. Since we are working with audio data, it is natural to use phoneme-based comparisons to determine the strength of overlap between the phrases. While the previous two scoring functions compare the strings on how they are written, the phoneme-based edit distance compares the strings on how they *sound*. The phoneme-based edit distance between two phrases is defined as the edit distance between the phonemic transcriptions (Kessler, 2005) of the two phrases. Fig. 3 shows a block-diagram of the flow. We can use several phonemic transcribers (Kessler, 2005) in Fig. 3. In our implementation, we used the Metaphone algorithm for phonemic transcriptions (Philips, 1990).

The phonemic score is then given as follows:

$$\mathbf{score_p}(p, q) = \frac{100}{\epsilon + \mathbf{PED}(p, q)}$$

where $\mathbf{PED}(p, q)$ is the phonemic edit distance as defined above, and ϵ is a small number to bound the phonemic score. For an example of p =“pause" and q =“clause", the phonemic score is $\mathbf{score_p}(p, q) = 47.62$ when $\epsilon = 0.1$, as $\mathbf{PED}(p, q) = 2$. To accommodate “_entity_" in our phrases, we remove it from our phrase before passing it through the phonemic transcriber.

A.2 Ablation test

We have observed in Sec. 4.3 that the phoneme-based scoring function works best when the context and commands are exact commands with no expected variations while it fails when homonyms are present in the utterances. In addition, we observed that the character-based scoring function performs better when the utterance is shorter (<2 words) while the word-based one works better for longer utterances (≥2 words).

Smart-switching and truncation: Based on above observations, we implemented a dynamic choosing of scoring function, where the average length of the hypotheses set is used to determine the appropriate scoring function. For shorter average length (< 2), the character-based scoring function is used, while word-based is used if average length is ≥ 2. We also observed few cases where an extremely low confidence hypothesis was having a slightly better match with the context, and getting pushed up in the re-ordering. To avoid such cases, we employed truncation, where the hypotheses list was truncated to only include hypotheses whose confidence was above a (relative) threshold, unless there is an exact match (distance is 0). Our results show that such an intelligent switching along with truncation resulted in a WER of 10.17% which is better compared to when we use only the word-based scoring function or the phoneme-based scoring function.

A.3 Implementation details and Ethical impact

As mentioned in the paper, we have collected data from English language speaking teachers using our voice assistant device in a classroom. The assistant currently only supports English language. These teachers participating in the pilot are from different schools of the United States and have signed a special privacy policy that ensures no PII information is stored. We also have the policy to not share the audio files externally beyond internal analytic purposes. This data is also anonymized and no user information is available for a given audio file. The collected audio files were annotated by an internal employee who was hired for the annotation requirements.

The algorithm described in the paper and the corresponding scoring functions were implemented in python for offline analysis[2]. We have implemented this algorithm on our voice assistant during the pilot program itself and thereby ensured (and observed) that users (school teachers) had an improved experience because of using it. The major packages used for this are fuzzywuzzy for distance measures and metaphone for phonemic transcriptions. This code was run on a simple laptop using CPU power as the algorithms do not require much computational power.

[2]We have made all our codes publicly available at: https://rb.gy/nc5xju