Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Improving Domain-Independent Cloud-Based Speech Recognition with Domain-Dependent Phonetic Post-Processing

Johannes Twiefel, Timo Baumann, Stefan Heinrich, and **Stefan Wermter**
University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D - 22527 Hamburg, Germany

Abstract

Automatic speech recognition (ASR) technology has been developed to such a level that off-the-shelf distributed speech recognition services are available (free of cost), which allow researchers to integrate speech into their applications with little development effort or expert knowledge leading to better results compared with previously used open-source tools.

Often, however, such services do not accept language models or grammars but process free speech from any domain. While results are very good given the enormous size of the search space, results frequently contain out-of-domain words or constructs that cannot be understood by subsequent domain-dependent natural language understanding (NLU) components. We present a versatile post-processing technique based on phonetic distance that integrates domain knowledge with open-domain ASR results, leading to improved ASR performance. Notably, our technique is able to make use of domain restrictions using various degrees of domain knowledge, ranging from pure vocabulary restrictions via grammars or N-Grams to restrictions of the acceptable utterances. We present results for a variety of corpora (mainly from human-robot interaction) where our combined approach significantly outperforms Google ASR as well as a plain open-source ASR solution.

1 Introduction

Google, Apple, Bing, and similar services offer very good and easily retrievable cloud-based automatic speech recognition (ASR) for many languages free of charge, which can also take advantage from constant improvements on the server side (Schalkwyk et al. 2010).

However, results that are received from such a service cannot be limited to a domain (e.g. by restricting the recognizer to a fixed vocabulary, grammar, or statistical language model) which may result in poor application performance (Morbini et al. 2013). For example, out-of-vocabulary errors for custom-made natural language understanding components may be higher with such generic services than with simpler and less reliable open-source speech recognition solutions, e.g. based on Sphinx (especially for languages where good open-source acoustic models are unavailable).

To address this gap, we propose a simple post-processing technique based on phonetic similarity that aligns the output from Google ASR (or any other similar service with 'unrestrictable' recognition output) to a specified sub-language that is more suitable for natural language understanding (NLU) in a given domain. The result string from the speech recognizer is transformed back to a sequence of phonemes which is then rescored to a language model based on domain knowledge. We consider phonemes to be the appropriate level of detail that (a) can still be recovered from ASR result strings, and (b) remains relatively stable to different ASR errors that are caused by inadequate language modelling. Our method is based on Sphinx-4 (Walker et al. 2004) and hence works with various kinds of language specifications such as grammars, statistical language models (SLMs), or blends of both kinds.

In the remainder of the paper, we discuss related previous work in the following section and detail our approach to post-processing in Section 3 and considerations on the implementation in Section 4. We present an experiment in which we use our post-processing technique in Section 5 and discuss the results in Section 6.

2 Related Work

Search by Voice developed by Google Inc. described by Schalkwyk et al. (2010) is a distributed speech recognition system. While voice activity detection (VAD) and feature extraction may be performed on the client (depending on the client's capabilities), the computationally expensive decoding step of speech recognition is performed on Google's servers, which then returns a list of hypotheses back to the client. The system employs acoustic models derived from GOOG-411 (Van Heerden, Schalkwyk, and Strope 2009), a telephony service that has been operated by Google. GOOG-411 enabled its users to search telephone numbers of businesses by using speech recognition and web search, and has collected large amounts of acoustic training data (about 5,000 hours of training data until 2010). Given this origin, a disadvantage of Search by Voice is the language model which is optimized for web searches. In addition, there is no public interface to change Google's ASR to a custom domain-dependent language model. Therefore, the benefit of good acoustic models cannot be exploited well in domain-specific projects.

2.1 Distributed vs. Local Speech Recognition

In the analysis of Morbini et al. (2013), several speech recognition systems (including cloud-based recognizers) are compared in terms of their applicability for developing domain-restricted dialogue systems. The results show that distributed speech recognition systems can offer better recognition accuracy than local customizable systems even if they are not adjusted to the domain; however, there is no clear 'best' system in their evaluation. It appears plausible that a combination of cloud-based recognizers with domain-restricted local system leads to an overall better system, which we aim at in this paper. In addition, Morbini et al. show that better ASR performance correlates with better NLU performance.

2.2 Sphinx-4

Sphinx-4 is an open-source speech recognition software developed at CMU (Walker et al. 2004) that is modular, easily embeddable and extensible. The frontend includes voice activity detection, and the decoder is based on time-synchronous Viterbi search using the token-pass algorithm (Young, Russell, and Thornton 1989). A main shortcoming are the weak open-source acoustic models which, for English, are often based on the HUB-4 data set (Garofolo, Fiscus, and Fisher 1997) and use MFCC features (Mermelstein 1976). However, the Sphinx' versatility can be exploited to completely change the feature representation (and meaning) as well as the search space representation (called 'linguist' in Sphinx terminology), which we aim at as explained in Section 4.3 below.

2.3 Post-processing Google's Search by Voice

Milette and Stroud (2012, ch. 17) present an approach of post-processing results from Google ASR on Android. The list of available commands and the results from Google's speech recognition are transformed using a phonetic indexing algorithm such as Soundex or Metaphone. However, phonetic indexing algorithms are meant to de-duplicate different spellings of the same names (Jurafsky and Martin 2009, p. 115). Thus, they (a) are optimized for names but not necessarily for other words of a language, and (b) may result in multiple target commands being reduced to the same representation. In addition, their matching is limited to individual words such that mis-segmentation in the ASR results cannot be recovered from. We extend the approach by Milette and Stroud (2012) using a more flexible rescoring method based on Levenshtein distance between recognition result and accepted language and enabling rescoring across word boundaries.

2.4 Levenshtein Distance Optimization

Ziółko et al. (2010) modify Levenshtein distance (Levenshtein 1966) using different insertion, deletion and substitution costs based on held-out data to generate word hypotheses from a DTW-based phoneme recognizer. Our work is orthogonal in that optimized Levenshtein costs would certainly further improve our results.

Zgank and Kacic (2012) propose to estimate acoustic confusability of words. Actual and recognized word tokens are transformed to their phoneme representations and the confusability score is computed based on normalized Levenshtein distances between the phoneme representations. Word confusabilities are used to estimate expected ASR performance given some currently available user commands. In contrast, we aim to improve ASR results by comparing a word's phonemes using Levenshtein distance either for individual words, for whole sentences, or for more expressive language models, by including Levenshtein computations into the Viterbi decoder.

2.5 Reranking the N-best List

Morbini et al. (2012) compare the performance of speech recognition systems including Google Search by Voice and taking N-best results into account. As had been previously found (Rayner et al. 1994), they state that often not the best matching hypothesis is at the top of the N-best list of results, which is why we will make use of N-best information below.

3 General Approach

As described in Section 2, distributed speech recognition systems like Google Search by Voice can provide better performance than local speech recognition systems because they often provide better trained acoustic models and possibly more advanced frontend processing, e.g. based on deep belief networks (Jaitly et al. 2012). Their disadvantage is that they often cannot be adjusted to specific domains by way of a task-specific language model, as domain knowledge cannot be externally provided.

Providing domain knowledge to the speech recognition service is impossible under this *black box* paradigm, and hence, we instead resort to post-processing the results of the system using the available domain knowledge. It turns out that there is hidden knowledge contained in the 'raw' results delivered by Google that our post-processing is able to recover: In preliminary test runs, we compared the word error rate (WER) and phoneme error rates (PER) of Google's speech recognition for the same hypotheses. PER was computed based on transforming words to phonemes for both reference text and recognition hypothesis and aligning them with standard Levenshtein distance. Those preliminary tests indicated that the PER is much lower than WER for the hypothesis returned from Google, which supports our initial assumption that there is more information contained in the results received from Google than is visible at the word level: e.g., in our tests, the word "learn" was often mis-recognized as "Lauren", which, despite being quite different *graphemically*, differs only marginally *phonemically*. We explain the ingredients to our post-processing technique in this section and detail the individual implementation strategies in the following Section 4.

3.1 Pronunciation Extraction

Under the assumption that the speech recognizer's acoustic models are adequate to the task, the first step to finding alternative word hypotheses is to recover the phoneme sequence

that is underlying the ASR hypothesis. While this task is often solved by simple pronunciation dictionaries that map every word's graphemic to its phonemic representation, this would be insufficient given the very large (and unknown) vocabulary of the Google speech recognizer. We thus trained the grapheme to phoneme converter SequiturG2P, (Bisani and Ney 2008) on CMUdict 0.7a[1], which enables our system to generate plausible phonemisations for any incoming word.

3.2 Phoneme Matching with Levenshtein Distance

We compare the phoneme sequences of the ASR output and the hypotheses based on domain knowledge using the Levenshtein distance (Levenshtein 1966), which is also frequently used to compare hypotheses on the word level (e.g. to computer WER).

Plain Levenshtein distance works well to find the most likely full sentence from a given set of sentences, or to replace every recognized word by the most similar word in a given vocabulary. However, it is insufficient to be used in a time-synchronous Viterbi search, as the binary costs (0 for matches, 1 for insertion, deletion or substitution) are too hard. We therefore use a cost of 0.1 for matches and 0.9 for all other edit operations in the implementation described in Subsection 4.3 below.[2] We also experiment with variable costs for phoneme substitution as detailed next.

3.3 Levenshtein Cost Optimization

We experiment with variable substitution costs to account for the intuitive notion that some phonemes are more easily interchanged than others. Specifically, we tried two approaches, one based on linguistic insight, the other based on actual data from the Google speech recognizer.

Costs Based on the IPA Table The international phonetic association classifies phones[3] according to an elaborate, yet regular scheme (IPA 1999): Consonants are categorized by place and manner of articulation (MOA), and phonation, with the place being (in order): *bilabial*, *labiodental*, *dental*, *alveolar*, *palatoalveolar*, *alveopalatal*, *palatal*, *velar*, *glottal* and *unknown*, manner being (in order): *plosive*, *fricative*, *nasal approximant*, and phonation being either *voiced* or *voiceless*. Inspired by the categorization for consonants, vowels are differentiated by height (*close*, *near-close*, *close-mid*, *mid*, *open-mid*, *near-open* and *open*) and backness (*front*, *near-front*, *central*, *near-back* and *back*). Furthermore, vowels may be *monophthongs* (i.e. static), or *diphthongs* (sliding between two phonemic positions).

Given this categorization, we compute the linguistic distance based on the distance between two phonemes in each of the given dimensions (e.g. plosives and fricatives are assigned a distance of 1, plosives and nasals a distance of 2, etc.). To be able to calculate a distance between vowels and consonants, vowels are added to the MOA category after *approximants* as *monophthongs* and *diphthongs*. To account for the different importance of the dimensions, we sum the errors but scale MOA error with a factor of 5.

Again, to remain in the range from 0.1 to 0.9 for substitution cost, we compute the cost as

$$c(x) = \frac{1}{(x \times max + 250) \times 0.1} + 0.1$$

where max is the maximum possible distance and x is the distance to be normalized.

Google Reverse In order to compute a scoring matrix that is tuned to the specifics of the Google speech recognizer, we let it recognize a set of development sentences. Results were then transformed to phonemes and aligned with the expected phoneme sequences. The count of substitutions between expected and actual phonemes were then calculated from the alignments (and costs again normalized to the range between 0.1 and 0.9). The outcomes for optimized Levenshtein costs are discussed in Section 5.

3.4 Domain Knowledge

To be able to adjust Google's speech recognition to a domain, the available domain knowledge needs to be modelled, ideally in terms of the estimated linguistic constructs that will be used. Our implementations currently provide for three kinds of linguistic restrictions. The weakest restriction can be implemented by allowing only a certain vocabulary (but no ordering restrictions for words); a very strong restriction can be set by only allowing a limited number of sentences (or commands) to be spoken at any point in time. While potentially least powerful (list of vocabulary) or least flexible (list of allowed sentences), these approaches have the advantage of being easy to implement and easy to administer as a non-speech expert.

More flexibility can be gained by using a 'full-blown' language model as used in current speech recognition engines: either (weighted) grammars, or N-Gram statistical language models, which are both more expressive and compact than the naïve approaches mentioned above. However, writing grammars or estimating N-Gram models requires a certain degree of expertise and their implementation is more complex, as will be shown in the following section.

4 Implemented Strategies

Using our general approach, we implemented three different post-processors, as depicted in Figure 1. In all approaches, the word-level hypothesis from the Google speech service is transformed to the phoneme level, and then matched using the respective domain model. Post-processors differ in the degree of the domain knowledge that is used and in the complexity of their implementation.

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Bisani and Ney (2008) report a PER of 5.8 % for SequiturG2P on CMUdict 0.6.

[2] In effect, we try to avoid 'dangerous' scoring regions in which the search manager otherwise could prune away hypotheses that would later turn out to be useful.

[3] Phones are equated with phonemes in the context of this work, even though real phoneticians or phonologists may disagree.

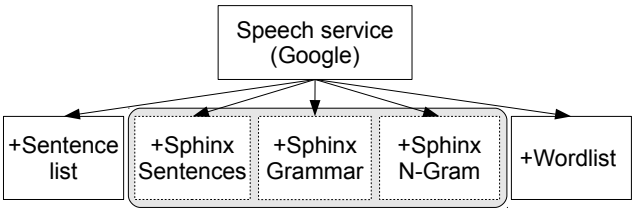

Figure 1: The overall system: speech recognition results can be passed on to post-processing based on full-sentence matching, word-based matching, or to one of several Sphinx-based post-processing schemes. (Implementations are ordered by degree of domain knowledge/language restriction, with the most restricted scheme at the very left.)

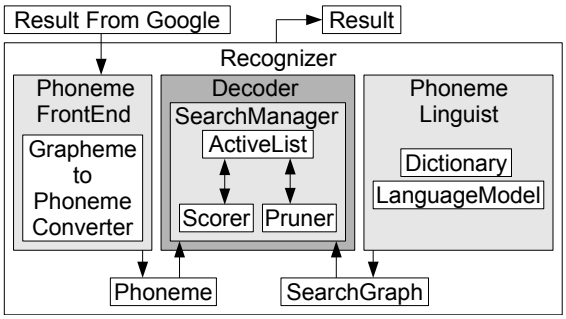

Figure 2: Components of the Google+Sphinx implementation, based on Figure 1 in (Walker et al. 2004).

4.1 Google+Sentencelist Approach

The simplest approach to a post-processor finds the sentence that best matches the hypothesis from a list of given in-domain sentences. Our implementation takes up to 10-best results that are returned by the speech service and computes the Levenshtein distances between these results and all its target sentences. The sentence that receives the minimum Levenshtein distance to any of the top-ranking hypotheses from the speech service is the winning sentence.

An advantage of this approach are the hard constraints of the results, as each possible result can be mapped to an expected sentence and then be handled even by simplistic NLU components. However, the hard constraints are also a disadvantage because the user cannot speak freely to the system. Even if the spoken input does not make sense at all, it is mapped to a best matching sentence which may not be desirable. To counteract this, a threshold (in terms of PER) can be defined, which, if surpassed by all sentences in the list, results in a special 'no-match' being returned. While the algorithm works very quick in general, its complexity grows linearly with the number of sentences accepted and the length of the allowed sentences.

4.2 Google+Wordlist Approach

For less constrained speech, we follow the approach by Milette and Stroud (2012) and implement a word-by-word post-processing scheme in which every word in the incoming (best) hypothesis is compared to the vocabulary, and the best-matching word is taken to be the target word.

An advantage of this approach is that the speaker can speak more freely to the system as the words order is not predetermined. One problem is that sometimes Google's speech service recognizes two spoken words as one or one spoken word as two words. In most of these cases, the hypothesized words are not similar to either of the intended vocabulary words and are transformed wrongly.

4.3 Google+Sphinx Approach

A more flexible post-processing approach that supports a wide range of language models can be built by making use of the Sphinx' versatility and extensibility. Sphinx' object encapsulation ensures that the content of frontend feature frames is ignored in all processing except the scoring stage, in which the emission probability of this feature frame given a Hidden Markov Model (HMM) triphone sub-state is computed. These computations are completely encapsulated in the search state representation (called 'linguist' in Sphinx terminology). In Sphinx-4 we replace the original frontend with our own phoneme frontend, which converts an incoming result string from Google ASR to its phonemic representation (see Figure 2). On the other end, we implemented a linguist that scores the phonemes against the search nodes' assigned phonemes using one of the costs as defined in Subsections 3.2 and 3.3.

Sphinx-4 uses Bakis HMM networks without skip-transitions (which are hard to implement). As there are usually many frames per phone, states that should be skipped can be allowed to emit one frame without accumulating much error. Thus, while phoneme insertion cost can be handled simply by state-repetition costs, phoneme deletion cannot be handled as easily. To enable deletions nonetheless, we duplicate all phoneme occurrences, which allows the Viterbi search to spread out deletion costs, as is conventionally done.

All other processing remains untouched, meaning that all kinds of language models provided by Sphinx-4 can be used. Furthermore, we are working to extend the integration with the Google speech service to make it possible to transparently integrate it into any Sphinx-4 recognition setup.

As this approach combines Google ASR and Sphinx-4, we call it Google+Sphinx (G+S) below.

5 Experiment and Results

We use several corpora relevant to HRI in the evaluation of our approach, which differ in domain size (and linguistic restriction) and recording conditions. We will first describe the corpora used and then present the results.

Scripted HRI data set (SCRIPTED): To test ASR under natural conditions in an HRI scenario, we use a corpus that has previously been recorded by Heinrich and Wermter (2011), which contains English speech from two non-native speakers (female and male) using headset microphones. The speakers read out (in an acted voice) sentences that were produced from a predefined grammar, for a total of 592 utterances. The corpus is especially useful as a grammar for parsing the utterances is available.

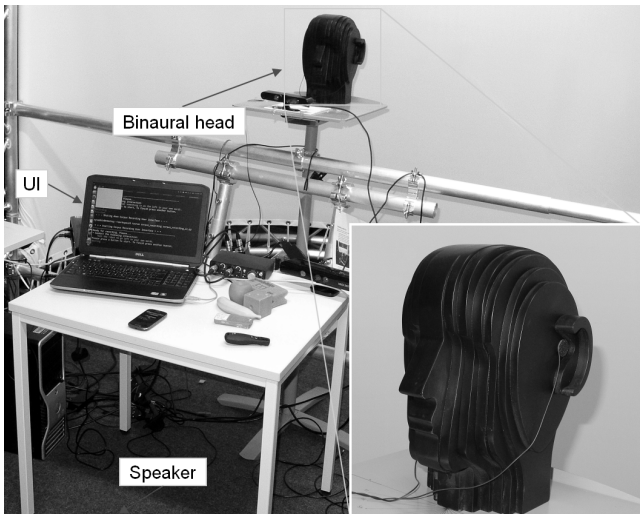

Figure 3: Setup of the SPONT corpus recording.

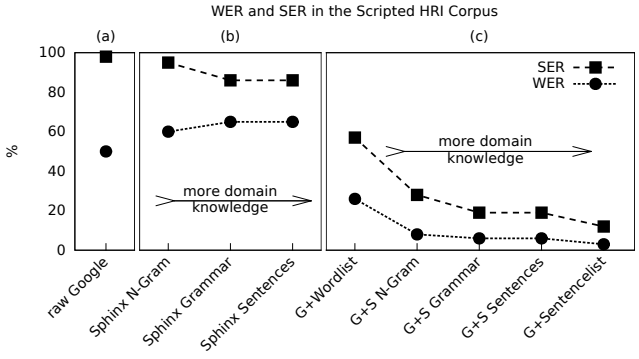

Figure 4: Performance comparison on the SCRIPTED corpus: (a) raw Google, (b) plain Sphinx with various setups, and (c) G+S with various setups. Levenshtein-based post-processing improves performance and improvements are higher the more domain-knowledge restricts the search.

TIMIT Core Test Set (TIMIT): To provide an easily comparable ASR test, we evaluate with the TIMIT corpus (Garofolo et al. 1993) of which we choose the Core Test Set which consists of 192 different sentences by 24 speakers using close-talking microphones. As the sentences spoken are very diverse, there is no speech recognition grammar for TIMIT.

Spontaneous HRI data set (SPONT): To also test ASR for robot-directed spontaneous speech, we collected an additional data set. The speakers were instructed to vocalize robot commands presented in keywords as free speech. The audio data was recorded by a binaural head that has acoustic characteristics similar to the characteristics of the human head, to test in natural HRI conditions, with a distance of 1.5 m to the speakers (see Figure 3). We collected 97 audio files from 15 different native and non-native English speakers from various countries. Speech was not restricted and too varied to be captured by a grammar.

We performed all experiments with 'raw' Google speech recognition, with a local Sphinx-4 installation using HUB-4 acoustic models, and with our post-processing techniques: G+Sentencelist, G+Wordlist, and our G+S combined post-processor with $0.1/0.9$ Levenshtein costs using as language models N-Grams, an (unweighted) grammar (if applicable), and a grammar that includes all possible sentences (thus similar to the G+Sentencelist technique, but without making use of N-best information). The results for sentence and word error rates are presented in Table 1; the columns for the SCRIPTED corpus are in addition plotted in Figure 4. In the sub-figures (b) and (c), system settings are ordered by the degree of language restriction / domain knowledge used.

As can be seen in Figure 4 (b) (compare also Table 1) the speech recognition performance for Sphinx is similar to Google, regardless of whether a grammar or N-Grams are used, or even the list of possible sentences (some improvement in SER comes at the cost of WER). The results indicate that Google's better acoustic models compensate for the Sphinx' domain knowledge. Figure 4 (c) shows the results for our post-processing strategies, again ordered by the amount of domain knowledge that is used. The results show that the combined systems greatly and significantly (Twiefel 2014) benefits from more domain knowledge, in which the superior acoustic model (Google) and tighter domain language restrictions (Sphinx) play together. For example, the G+S N-Grams condition (SCRIPTED corpus) results in a WER of $8.0\,\%$, which is a relative improvement of about $85\,\%$ to the raw Google ($50.2\,\%$) and Sphinx N-Gram ($60.5\,\%$) conditions. Finally, the slight improvement between G+S Sentences and G+Sentencelist may show the advantage of using N-best results, which is currently not implemented in the combined system versions.

Across the corpora shown in Table 1, the tendency of better results with more domain knowledge is repeated. Specifically, using word N-Grams in combination with phonetic post-processing radically cuts down error rates from using Google ASR or Sphinx-4 N-Grams alone. However, error rates especially for the SPONT corpus remain high, which points to an inability of either recognizer's acoustic model to cope with spontaneous speech collected with a far-distance microphone.

Finally, to our surprise and in contrast to the findings by (Ziółko et al. 2010), the choice of confusion cost estimation between different phonemes was negligible for G+S post-processing: Table 2 shows the results of a small-scale optimization experiment on parts of the SCRIPTED corpus. Results differ only marginally. Given that the many free parameters in the IPA table approach would have to be estimated from data or require more linguistic knowledge (as, apparently, our ad-hoc heuristics deteriorates results), and given that the 'Google Reverse' approach of estimating phoneme confusion from data would be prone to AM improvements on the server side (a main incentive of using cloud-based services), we conclude that the alignment flexibility of our simplistic $0.1/0.9$ heuristics is sufficient, given that the Google ASR already has a relatively low phoneme error rate.

Table 1: Evaluation results (WER and SER) for all corpora: raw speech recognition results for Google ASR and various Sphinx settings, simple word- or sentence-optimizing Levenshtein post-processing of Google ASR, and the combined Google+Sphinx-based Levenshtein post-processing using different language models (always using 0.1/0.9 costs for post-processing).

	WER in %			SER in %		
System	SCRIPTED	TIMIT	SPONT	SCRIPTED	TIMIT	SPONT
Raw Google	50.230	33.356	74.717	97.804	80.208	91.667
Sphinx N-Gram	60.462	23.949	69.057	95.101	64.063	93.750
Sphinx Grammar	65.346	*	*	85.980	*	*
Sphinx Sentences	65.346	52.675	85.283	85.980	54.167	86.458
Google+Sentencelist	3.077	0.382	71.698	11.993	0.521	77.083
Google+Wordlist	23.231	30.510	71.509	57.432	79.688	90.625
Google+Sphinx N-Gram	7.962	18.000	67.547	27.703	38.020	86.458
Google+Sphinx Grammar	6.038	*	*	19.257	*	*
Google+Sphinx Sentences	5.846	1.401	65.094	18.581	10.417	71.875

*No grammar available

Table 2: Results for the three confusion cost estimations for alignment (WER in %, G+S N-Gram condition).

Confusion cost est.	SCRIPTED	TIMIT	SPONT
Derived IPA Table	9.692	17.197	68.679
Google Reverse	8.192	17.006	66.038
0.1 / 0.9	7.962	18.000	67.547

6 Conclusions and Future Work

We have presented our work on mitigating the gap between the superior acoustic modelling performance of free but unrestrictable cloud-based speech recognition services and relatively weak, open-source acoustic models, and the high performance impact of language restrictions provided by domain knowledge. We have implemented several approaches that combine the freely available cloud-based Google speech recognizer with phonetic post-processing that integrates domain knowledge and allows varying degrees of restriction as can be expressed by different kinds (and configurations) of standard language models and grammars in the Sphinx-4 framework.

Our results (compare Figure 4) show very clearly the advantage of supplying domain knowledge to speech recognition (the more the better). Our combined Sphinx-based solution handles Levenshtein-based phoneme alignment in an identical way as standard speech recognition using the Viterbi algorithm and has the clear advantage of being able to operate with varying degrees and representations of domain knowledge, over the simpler list-based (vocabulary or allowed sentences) post-processing methods, which, however, do not require expert knowledge to set up.

Our usage of the Viterbi decoder does, however, come with some drawbacks, e.g. regarding the inconsistent handling of deletions (as discussed above) which may also hinder learning the optimal edit costs from data. Furthermore, as the cloud-based recognizer's speech is used as input to the decoding step, and input to decoding is assumed to be non-ambiguous, using N-best results as input to the Sphinx-based post-processor will be difficult. Figure 4 (c) shows the advantage of including N-best input into the matching process (compare the two right-most columns), which, however, might only be possible with an approach that is similar to lattice-rescoring (Jurafsky and Martin 2009, p. 375). Using multiple alternative phonemisations (as provided by SequiturG2P) in the alignment process faces similar difficulties. Our N-Gram implementation of Sphinx-based post-processing is currently limited to bigrams as the Sphinx-4 linguist with arbitrary N-Gram support has a much higher code-complexity. However, this is not a principled limitation and extending the implementation to cover higher-order N-Grams will be conceptually straightforward.[4]

Google speech recognition can run in an incremental (i.e. online) mode (McGraw and Gruenstein 2012) in which results are produced while speech is still being produced. Using a freely available incremental processing toolkit based on Sphinx-4 (Baumann, Atterer, and Schlangen 2009), we expect that our approach is easy to port for our use-case as well, which will enable more advanced interaction capabilities in human-robot interaction (HRI) settings.

In summary, powerful cloud-based but domain-independent speech recognition systems can be improved by using domain knowledge in phonetic distance-based post-processing. These systems offer the opportunity to extend interaction capabilities of intelligent systems, e.g. in known HRI scenarios. To foster such research, our implemented framework DOCKS (DOmain- and Cloud-based Knowledge for Speech recognition) is available as open-source software at www.informatik.uni-hamburg.de/WTM/software/. We plan to work towards making it a full drop-in replacement of the Sphinx-4 frontend and acoustic models.

[4]This will also enable code optimizations in Sphinx' arbitrary N-Gram code which will further reduce post-processing runtime to be negligible compared to the speech service (which, in general, is also very fast).

Acknowledgments. The authors would like to thank Sven Magg, Erik Strahl and the students of the international Master Intelligent Adaptive Systems for their help in collecting the SPONT corpus. This research has been partly supported by the EU project RobotDoC under 235065 ROBOT-DOC from the 7th Framework Programme (FP7), by the DFG German Research Foundation (grant #1247) – International Research Training Group CINACS, and by the Daimler and Benz foundation (PostDoc grant #32-02/13).

References

Baumann, T.; Atterer, M.; and Schlangen, D. 2009. Assessing and improving the performance of speech recognition for incremental systems. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 380–388.

Bisani, M., and Ney, H. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication* 50(5):434–451.

Garofolo, J. S.; Lamel, L. F.; Fisher, W. M.; Fiscus, J. G.; Pallett, D. S.; and Dahlgren, N. L. 1993. DARPA TIMIT acoustic phonetic continuous speech corpus CDROM.

Garofolo, J.; Fiscus, J. G.; and Fisher, W. M. 1997. Design and preparation of the 1996 HUB-4 broadcast news benchmark test corpora. In *Proceedings of the 1997 DARPA Speech Recognition Workshop*, 15–21.

Heinrich, S., and Wermter, S. 2011. Towards robust speech recognition for human-robot interaction. In Narioka, K.; Nagai, Y.; Asada, M.; and Ishiguro, H., eds., *Proceedings of the IROS2011 Workshop on Cognitive Neuroscience Robotics (CNR)*, 29–34.

IPA, I. 1999. *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press.

Jaitly, N.; Nguyen, P.; Senior, A. W.; and Vanhoucke, V. 2012. Application of pretrained deep neural networks to large vocabulary speech recognition. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA.

Jurafsky, D., and Martin, J. H. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson International, 2 edition.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics – Doklady* 10(8):707–710.

McGraw, I., and Gruenstein, A. 2012. Estimating word-stability during incremental speech recognition. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA.

Mermelstein, P. 1976. Distance measures for speech recognition – psychological and instrumental. In *Proceedings of the Joint Workshop on Pattern Recognition and Artificial Intelligence*, 374–388.

Milette, G., and Stroud, A. 2012. *Professional Android Sensor Programming*. ITPro collection. Wiley.

Morbini, F.; Audhkhasi, K.; Artstein, R.; Van Segbroeck, M.; Sagae, K.; Georgiou, P.; Traum, D. R.; and Narayanan, S. 2012. A reranking approach for recognition and classification of speech input in conversational dialogue systems. In *Proceedings of the 2012 IEEE Spoken Language Technology Workshop (SLT)*, 49–54. IEEE.

Morbini, F.; Audhkhasi, K.; Sagae, K.; Artstein, R.; Can, D.; Georgiou, P.; Narayanan, S.; Leuski, A.; and Traum, D. 2013. Which ASR should I choose for my dialogue system? In *Proceedings of the 14th annual SIGdial Meeting on Discourse and Dialogue*, 394–403.

Rayner, M.; Carter, D.; Digalais, V.; and Price, P. 1994. Combining knowledge sources to reorder n-best speech hypothesis lists. In *Proceedings of the ARPA Human Language Technology Workshop*, 217–221.

Schalkwyk, J.; Beeferman, D.; Beaufays, F.; Byrne, B.; Chelba, C.; Cohen, M.; Kamvar, M.; and Strope, B. 2010. Your word is my command: Google search by voice: A case study. In *Advances in Speech Recognition*. Springer. 61–90.

Twiefel, J. 2014. Development and evaluation of semantically constained speech recognition architectures. Master's thesis, Universität Hamburg, Dept. of Informatics, Hamburg, DE.

Van Heerden, C.; Schalkwyk, J.; and Strope, B. 2009. Language modeling for what-with-where on GOOG-411. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA.

Walker, W.; Lamere, P.; Kwok, P.; Raj, B.; Singh, R.; Gouvea, E.; Wolf, P.; and Woelfel, J. 2004. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Technical Report SMLI TR2004-0811, Sun Microsystems Inc.

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

Zgank, A., and Kacic, Z. 2012. Predicting the acoustic confusability between words for a speech recognition system using Levenshtein distance. *Electronics and Electrical Engineering* 18(8):81–84.

Ziółko, B.; Gałka, J.; Jadczyk, T.; and Skurzok, D. 2010. Modified weighted Levenshtein distance in automatic speech recognition. In *Proceedings of the 16th Conference on Applications of Mathematics in Biology and Medicine*,