Mitigating Noisy Inputs for Question Answering

Denis Peskov,[1] *Joe Barrow,*[1] *Pedro Rodriguez,*[1] *Graham Neubig,*[2] *Jordan Boyd-Graber*[3]

[1]University of Maryland Department of Computer Science and UMIACS
[2]Carnegie Mellon University Language Technology Institute
[3]University of Maryland Department of Computer Science, iSchool, UMIACS, and LSC

{dpeskov, jdbarrow, pedro}@cs.umd.edu, gneubig@cs.cmu.edu, jbg@umiacs.umd.edu

Abstract

Natural language processing systems are often downstream of unreliable inputs: machine translation, optical character recognition, or speech recognition. For instance, virtual assistants can only answer your questions after understanding your speech. We investigate and mitigate the effects of noise from Automatic Speech Recognition systems on two factoid Question Answering (QA) tasks. Integrating confidences into the model and forced decoding are empirically shown to improve the accuracy of downstream neural QA systems. We create and train models on a novel synthetic corpus of over 500,000 noisy sentences and evaluate on two human corpora from Quizbowl and Jeopardy! competitions.

1. Introduction

Progress on question answering (QA) has claimed human-level accuracy. However, most factoid QA models are trained and evaluated on clean text input, which becomes noisy when questions are spoken due to Automatic Speech Recognition (ASR) errors. This consideration is disregarded in trivia match-ups between machines and humans: IBM Watson [1] on Jeopardy! and Quizbowl matches between machines and trivia masters [2] provide text data for machines while humans listen. A fair test would subject both humans and machines to speech input.

Unfortunately, there are no large *spoken* corpora of factoid questions with which to train models; text-to-speech software can be used as a method for generating training data at scale for question answering models (Section 2). Although synthetic data is less realistic than true human-spoken questions it easier and cheaper to collect at scale, which is important for training. These synthetic data are still useful; in Section 4.1, models trained on synthetic data are applied to human spoken data from Quizbowl tournaments and Jeopardy!

Noisy ASR is particularly challenging for QA systems (Figure 1). While humans and computers might know the title of a "revenge novel centering on Edmund Dantes by Alexandre Dumas", transcription errors may mean deciphering "novel centering on edmond dance by alexander <*unk*>" instead. Dantes and Dumas are low-frequency words in the English language and hence likely to be misinterpreted by a generic ASR model; however, they are particularly important for answering the question.

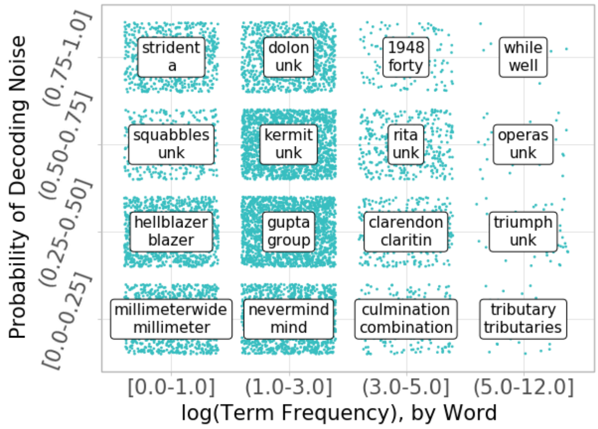

Figure 1: ASR *errors on* QA *data: original spoken words (top of box) are garbled (bottom). While many words become into "noise"—frequent words or the unknown token—consistent errors (e.g., "clarendon" to "clarintin") can help downstream systems. Additionally, words reduced to <unk> (e.g., "kermit") can be useful through forced decoding into the closest incorrect word (e.g., "hermit" or even "car").*

Additionally, the introduction of distracting words (e.g., "dance") causes QA models to make errors [3]. Section 2.1 characterizes the signal in this noise: key terms such as named entities are often missing or corrupted, which is detrimental for QA.

Previous approaches to mitigate ASR noise for answering mobile queries [4] or building bots [5]—typically use unsupervised methods such as term-based information retrieval. Our datasets for training and evaluation can produced *supervised* systems that directly answer spoken questions. Machine translation [6] also uses ASR confidences; we evaluate similar methods on QA.

Specifically, some accuracy loss from noisy inputs can be mitigated through a combination of forcing unknown words to be decoded as the closest option (Section 3.2), and incorporating the uncertainties of the ASR model directly in neural models (Section 3.3).

The forced decoding method reconstructs missing terms by using patterns in errors as well as terms related to the transcribed input. Word-level confidence scores incorporate uncertainty from the ASR system into neural models. Section 4 compares these methods against baseline methods on our synthetic and human speech datasets for Jeopardy! and Quizbowl.

2. Spoken Question Answering Datasets

Neural networks require a large training corpus, but recording hundreds of thousands of questions is not feasible. Crowd-sourcing with the required quality control (speakers who say "cyclohexane" correctly) is expensive. As an alternative, we generate a data-set with Google Text-to-Speech on 96,000 factoid questions from a trivia game called Quizbowl [2], each with 4–6 sentences for a total of over 500,000 sentences.[1] We then decode these utterances using the Kaldi chain model [7], trained on the Fischer-English dataset [8] for consistency with past results on mitigating ASR errors in MT [6]. This model has a Word Error Rate (WER) of 15.60% on the eval2000 test set. The WER increases to 51.76% on our Quizbowl data, which contains out of domain vocabulary. The most BLEU improvement in machine translation under noisy conditions could be found in this middle WER range, rather than in values below 20% or above 80% [6]. Retraining the model on the Quizbowl domain would mitigate this noise; however, in practice one is often at the mercy of a pre-trained recognition model and our methods address this scenario. Machine translation has also been added to machine translation data [9, 10]. Alternate methods for collecting large scale audio data include Generative Adversarial Networks [11] and manual recording [12].

The task of QA requires the system to provide a correct answer out of many candidates based on the question's wording. We test on two varieties of different length and framing. Quizbowl questions, which are generally four to six sentences, tests a user's depth of knowledge; early clues are challenging and obscure but they progressively become easy and well-known. Competitors can answer these types of questions at any point. Computer QA is competitive with the top players [13]. Jeopardy! questions are single sentences and can only be answered after the question ends. To test this alternate syntax, we use the same method of data generation on a dataset of over 200,000 Jeopardy questions [14].

2.1. Why QA is challenging for ASR

ASR changes the features of the recognized text in several important ways: the overall vocabulary is quite different and important words are corrupted. First, it reduces the overall vocabulary. In our dataset, the vocab drops from 263,271 in the original data to a mere 33,333. This is expected, as ASR only has 42,000 words in its vocab, so the long tail of the Zipf's curve is lost. Second, unique words—which may be central to answering the question—are lost or misinterpreted; over 100,000 of the words in the original data occur only once. Finally, ASR systems tend to deletes words which makes the sentences shorter; in ourcase, the average length decreases from 21.62 to 18.85 words per sentence.

The decoding system is able to express uncertainty by predicting <*unk*>. These account for slightly less than 10% of all our word tokens, but is a top-2 prediction for 30% of the 260,000 original words. For QA, words with a high TF-IDF measure are valuable. While some words are lost, others can likely be recovered: "hellblazer' becomes "blazer", "clarendon" becoming "claritin". We evaluate this by fitting a TF-IDF model on the Wikipedia dataset and then comparing the average TF-IDF per sentence between the original and the ASR data. The average TF-IDF score drops from 3.52 to 2.77 per sentence.

[1] http://cloud.google.com/text-to-speech

3. Mitigating Noise

This section discusses two approaches to mitigating the effects of missing and corrupted information caused by ASR systems. The first approach—forced decoding—exploits systematic errors to arrive at the correct answer. The second uses confidence information from the ASR system to down-weight the influence of low-confidence terms. Both approaches improve accuracy over a baseline DAN model and show promise for short single-sentence questions. An IR approach is more effective on long questions.

3.1. IR Baseline

The IR baseline reframes Jeopardy! and Quizbowl QA tasks as document retrieval ones with an inverted search index. We create one document per distinct answer; each document has a text field formed by concatenating all questions with that answer together. At test time questions are treated as queries, and documents are scored using BM25 [15, 16]. We implement this baseline with Elastic Search and Apache Lucene.

3.2. Forced Decoding

We have systematically lost information. We could predict the answer if we had access to certain words in the original question and further postulate that wrong guesses are better than knowing that a word is unknown.

We explore commerical solutions—Bing, Google, IBM, Wit—with low transcription errors. However, their APIs ensure that an end-user often cannot extract anything more than one-best transcriptions, along with an aggregate confidence for the sentence. Additionally, the proprietary systems are moving targets, harming reproducibility.

We use Kaldi [17] for all experiments. Kaldi is a commonly-used, open-source tool for ASR; its maximal transparency enables approaches that incorporate uncertainty into downstream models. Kaldi provides not only top-1 predictions, but also confidences of words, entire lattices, and phones (Table 1). Confidences are the same length as the text, range from 0.0 to 1.0 in value, and correspond to the respective word or phone in the sequence. The mean one-best confidence is 91%.

The typical end-use of an ASR system wants to know when when a word is not recognized. By default, a graph will have a token that represents an unknown; in Kaldi, this becomes <*unk*>. At a human-level, one would want to know that an out of context word happened.

However, when the end-user is a downstream model, a systematically wrong prediction may be better than a

Table 1: *As original data are translated through* ASR, *it degrades in quality. One-best output captures per-word confidence. Full lattices provide additional words and phone data captures the raw* ASR *sounds. Confidence models and forced decoding could be used for such data.*

Clean	For 10 points, name this revenge novel centering on Edmond Dantes, written by Alexandre Dumas ...
1-Best	for$^{0.935}$ ten$^{0.935}$ points$^{0.871}$ same$^{0.617}$ this^{1} ... revenge novel centering on <*unk*> written by alexander <*unk*> ...
"Lattice"	for$^{0.935}$ [eps]$^{0.064}$ pretend$^{0.001}$ ten$^{0.935}$ pretend point points point name same named name names this revenge novel ...
Phones	f_B$^{0.935}$ er_E$^{0.935}$ t_B$^{0.935}$ eh_I^{1} n_E$^{0.935}$ ... p_B oy_I n_I t_I s_E sil s_B ey_I m_E dh_B ih_I s_E r_B iy_I v_I eh_I n_I jh_E n_B aa_I v_I ah_I l_I ...

generic statement of uncertainty. So by removing all reference to <*unk*> in the model's Finite State Transducer, we force the system to decode "Louis Vampas" as "Lousiana" rather than <*unk*>. The risk we run with this method is introducing words not present in the original data. For example, "count" and "mount" are similar in sound but not in context embeddings. Hence, we need a method to downweight incorrect decoding.

3.3. Confidence augmented DAN

We build on Deep Averaging Networks [18, DAN], assuming that deep bag-of-words models can improve predictions and be robust to corrupted phrases. The errors introduced by ASR can hinder sequence neural models as key phrases are potentially corrupted and syntactic information is lost.

The original Deep Averaging Network, or DAN, classifier has three sections: a "neural-bag-of-words" (NBOW) encoder, which composes all the words in the document into a single vector by averaging the word vectors; a series of hidden transformations, which give the network depth and allow it to amplify small distinctions between composed documents; and a softmax predictor.

The encoded representation **r** is the averaged embeddings of input words. The word vectors exist in an embedding matrix **E**, from which we can look up a specific word w with $\mathbf{E}[w]$. The length of the document is N. To compute the composed representation r, the DAN averages all of the word embeddings:

$$\mathbf{r} = \frac{\sum_i^N \mathbf{E}[w_{\mathrm{i}}]}{N} \tag{1}$$

The network weights **W**, consist of a weight-bias pair for each layer of transformations $(\mathbf{W^{(h_i)}}, \mathbf{b^{(h_i)}})$ for each layer i in the list of layers L. To compute the hidden representations for each layer, the DAN linearly transforms the input and then applies a nonlinearity: $\mathbf{h_0} = \sigma(\mathbf{W^{(h_0)}}\mathbf{r} + \mathbf{b^{(h_0)}})$. Successive hidden representations h_{i} are: $\mathbf{h_i} = \sigma(\mathbf{W^{(h_i)}}\mathbf{h_{i\text{-}1}} + \mathbf{b^{(h_i)}})$. The final layer in the DAN is a softmax output: $\mathbf{o} = \mathrm{softmax}(\mathbf{W^{(o)}}\mathbf{h_L} + \mathbf{b^{(o)}})$. We modify the original DAN models to use word-level confidences from the ASR system as a feature.

In increasing order of complexity, the variations are: a Confidence Informed Softmax DAN, a Confidence Weighted Average DAN, and a Word-Level Confidence DAN. We represent the confidences as a vector **c**, where each cell c_{i} contains the ASR confidence of word w_{i}.

The simplest model averages the confidence across the whole sentence and adds it as a feature to the final output classifier. For example in Table 1, "for ten points" averages to 0.914. We introduce an additional weight in the output $\mathbf{W^c}$, which adjusts our prediction based on the average confidence of each word in the question.

However, most words have high confidence, and thus the average confidence of a sentence or question level is high. To focus on *which* words are uncertain we weight the word embeddings by their confidence attenuating uncertain words before calculating the DAN average.

Weighting by the confidence directly removes uncertain words, but this is too blunt an instrument, and could end up erasing useful information contained in low-confidence words, so we instead learn a function based on the raw confidence from our ASR system. Thus, we recalibrate the confidence through a learned function f:

$$f(\mathbf{c}) = \mathbf{W^{(c)}}\mathbf{c} + \mathbf{b^{(c)}} \tag{2}$$

and then use that scalar in the weighted mean of the DAN representation layer:

$$\mathbf{r}^{**} = \frac{\sum_i^N \mathbf{E}[w_{\mathrm{i}}] * f(c_{\mathrm{i}})}{N}. \tag{3}$$

In this model, we replace the original encoder **r** with the new version $\mathbf{r}^{**}$ to learn a transformation of the ASR confidence that down-weights uncertain words and up-weights certain words. This final model is referred to in the results as "Confidence Model".

Architectural decisions were determined by hyperparameters sweeps and are consistent across experiments and include: having a single hidden layer of 1000 dimensionality for the DAN, multiple drop-out, and batch-norm layers, and a scheduled ADAM optimizer. Our DAN models train until convergence, as determined by early-stopping. Code is implemented in PyTorch [19], with TorchText for batching.[2]

4. Results

Achieving 100% accuracy on this dataset is not a realistic goal, as not all test questions are answerable (specifically, some answers do not occur in the training data and hence cannot be learned by a machine learning system). Baselines for the DAN (Table 2) establish realistic goals: a DAN trained and evaluated on the *same train and dev set*, only

[2]Code, data, and additional analysis available at `https://github.com/DenisPeskov/QBASR`

Table 2: *Both forced decoding (FD) and the best confidence model improve accuracy. Jeopardy only has an At-End-of-Sentence metric, as questions are one sentence in length. Combining the two methods leads to a further joint improvement. The IR and DAN with clean data accuracies are provided as reference.*

	Quizbowl				Jeopardy!	
	Synth		Human		Synth	Human
Model	Start	End	Start	End		
Methods Tested on Clean Data						
IR	0.064	0.544	0.400	1.000	0.190	0.050
DAN	0.080	0.540	0.200	1.000	0.236	0.033
Methods Tested on Corrupted Data						
IR base	0.021	0.442	0.180	0.560	0.079	0.050
DAN	0.035	0.335	0.120	0.440	0.097	0.017
FD	0.032	0.354	0.120	0.440	0.102	0.033
Confidence	0.036	0.374	0.120	0.460	0.095	0.033
FD+Conf	0.041	0.371	0.160	0.440	0.109	0.033

in the original non-ASR form, correctly predicts 54% of the answers. Noise drops this to 44% with the best IR model and down to $\approx 30\%$ with neural approaches.

Since the noisy data quality makes full recovery unlikely, we view any improvement over the neural model baselines as recovering valuable information. At the question-level, strong IR outperforms the DAN by around 10%.

Since IR can avoid all the noise while benefiting from additional independent data points, it scales as the length of data increases. There is additional motivation to investigate this task at the sentence-level. Computers can beat humans at the game by knowing certain questions immediately; the first sentence of the Quizbowl question serves as a proxy for this threshold. Our proposed combination of forced decoding with a neural model led to the highest test accuracy results and outperforms the IR one at the sentence level.

A strong TF-IDF IR model can top the best neural model at the multi-sentence question level in Quizbowl; multiple sentences are important because they progressively become easier to answer in competitions. However, our models improve accuracy on the shorter first-sentence level of the question. This behavior is expected since textscir methods are explicitly designed to disregard noise and can pinpoint the handful of unique words in a long paragraph; conversely they are less accurate when they extract words from a single sentence.

4.1. Qualitative Analysis & Human Data

The synthetic dataset facilitates large-scale machine learning, but ultimately we care about performance on human data. For Quizbowl we record questions read by domain experts at a competition. To account for variation in speech, we record five questions across ten different speakers, varying in gender and age; this set of fifty questions is used as the human test data. Figure 3 has examples of variations. For Jeopardy! we manually parsed a complete episode by question.

Table 3: *Variation in different speakers causes different transcriptions of a question on Oxford.*

Speaker	Text
Base	John Deydras, an insane man who claimed to be Edward II, stirred up trouble when he seized this city's Beaumont Palace.
S1	unk an insane man who claimed to be the second unk trouble when he sees unk beaumont → Richard_I_of_England
S2	john dangerous insane man who claims to be the second stirring up trouble when he sees the city's beaumont → London
S3	unk dangerous insane man who claim to be unk second third of trouble when he sees the city's unk palace → Baghdad

The predictions of the regular DAN and the confidence version can differ. For input about The House on Mango Street, which contains words like "novel", "character", and "childhood" alongside a corrupted name of the author, the regular DAN predicts The Prime of Miss Jean Brodie, while our version predicts the correct answer.

4.2. Discussion & Future Work

Confidences are a readily human-interpretable concept that may help build trust in the output of a system. Transparency in the quality of up-stream content can lead to downstream improvements in a plethora of NLP tasks.

Exploring sequence models or alternate data representations may lead to further improvement.

Including full lattices may mirror past results for machine translation [6] for the task of question answering. Phone-level approaches work in Chinese [12], but our phone models had lower accuracies than the baseline, perhaps due to a lack of contextual representation. Using unsupervised approaches for ASR [20, 21] and training ASR models for decoding Quizbowl or Jeopardy! words are avenues for further exploration.

5. Conclusion

Question answering, like many NLP tasks are impaired by noisy inputs. Introducing ASR into a QA pipeline corrupts the data. A neural model that uses the ASR system's confidence outputs and systematic forced decoding of words rather than unknowns improves QA accuracy on Quizbowl and Jeopardy! questions. Our methods are task agnostic and can be applied to other supervised NLP tasks. Larger human-recorded question datasets and alternate model approaches would ensure audio questions are answered accurately, allowing human and computer trivia players to compete on an equal playing field.

6. Acknowledgments

This work was supported by NSF Grants IIS-1748663 and IIS-1748642. The views expressed in this paper are our own. We thank the reviewers and the Quizbowl and Kaldi communities for their help.

7. References

[1] D. A. Ferrucci, “Build Watson: an overview of DeepQA for the Jeopardy! challenge,” in *19th International Conference on Parallel Architecture and Compilation Techniques*, 2010, pp. 1–2.

[2] J. Boyd-Graber, S. Feng, and P. Rodriguez, *Human-Computer Question Answering: The Case for Quizbowl*. Springer Verlag, 2018.

[3] R. Jia and P. Liang, “Adversarial examples for evaluating reading comprehension systems,” in *Proceedings of Empirical Methods in Natural Language Processing*, 2017, pp. 2021–2031.

[4] T. Mishra and S. Bangalore, “Qme!: A speech-based question-answering system on mobile devices,” in *Human Language Technologies:*, 2010, pp. 55–63.

[5] A. Leuski, R. Patel, D. Traum, and B. Kennedy, “Building effective question answering characters,” in *Proceedings of the Annual SIGDIAL Meeting on Discourse and Dialogue*, 2009, pp. 18–27.

[6] M. Sperber, G. Neubig, J. Niehues, and A. Waibel, “Neural lattice-to-sequence models for uncertain inputs,” in *Proceedings of the Association for Computational Linguistics*, 2017.

[7] V. Peddinti, G. Chen, V. Manohar, T. Ko, D. Povey, and S. Khudanpur, “Jhu aspire system: Robust lvcsr with tdnns, ivector adaptation and rnn-lms,” in *Automatic Speech Recognition and Understanding (ASRU), IEEE Workshop on*, 2015, pp. 539–546.

[8] C. Cieri, D. Miller, and K. Walker, “The fisher corpus: a resource for the next generations of speech-to-text,” in *Proceedings of the Language Resources and Evaluation Conference*, 2004.

[9] P. Michel and G. Neubig, “Mtnt: A testbed for machine translation of noisy text,” in *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

[10] Y. Belinkov and Y. Bisk, “Synthetic and natural noise both break neural machine translation,” in *Proceedings of the International Conference on Learning Representations*, 2017.

[11] C. Donahue, B. Li, and R. Prabhavalkar, “Exploring speech enhancement with generative adversarial networks for robust speech recognition,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5024–5028.

[12] C.-H. Lee, S.-M. Wang, H.-C. Chang, and H.-Y. Lee, “Odsqa: Open-domain spoken question answering dataset,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 949–956.

[13] I. Yamada, R. Tamaki, H. Shindo, and Y. Takefuji, “Studio Ousia’s quiz bowl question answering system,” in *NIPS Competition: Building Intelligent Systems*, 2018, pp. 181–194.

[14] M. Dunn, L. Sagun, M. Higgins, V. U. Güney, V. Cirik, and K. Cho, “Searchqa: A new Q&A dataset augmented with context from a search engine,” *CoRR*, vol. abs/1704.05179, 2017.

[15] J. Ramos, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the International Conference of Machine Learning*, 2003.

[16] S. Robertson, H. Zaragoza *et al.*, “The probabilistic relevance framework: Bm25 and beyond,” *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[17] D. Povey, A. Ghoshal, G. Boulianne, N. Goel, M. Hannemann, Y. Qian, P. Schwarz, and G. Stemmer, “The Kaldi speech recognition toolkit,” in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[18] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, “Deep unordered composition rivals syntactic methods for text classification,” in *Proceedings of the Association for Computational Linguistics*, 2015.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *Conference on Neural Information Processing Systems: Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 2017.

[20] F. Wessel and H. Ney, “Unsupervised training of acoustic models for large vocabulary continuous speech recognition,” *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 23–31, 2004.

[21] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Proceedings of Advances in Neural Information Processing Systems*, 2009, pp. 1096–1104.

[22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *Proceedings of the Association for Computational Linguistics*, 2002, pp. 311–318.

[23] C. E. Shannon, “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

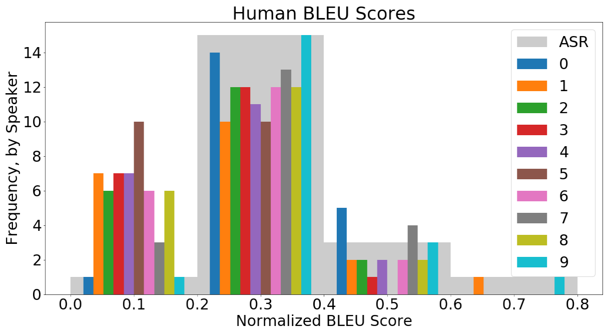

Figure 2: *A comparison of* BLEU *score distributions across human speakers (color-coded) to our artificial method, visualized by the step line. The distributions of* BLEU *scores are similar, with human data being slightly lower, justifying our weak supervision training approach.*

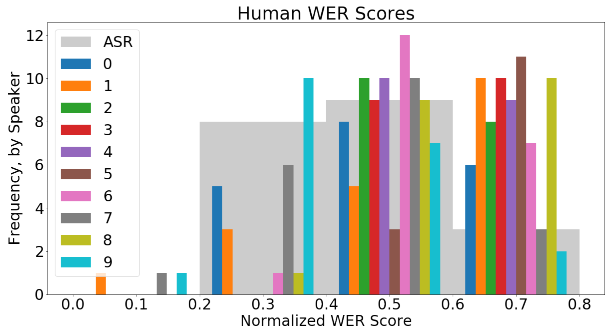

Figure 3: *Similarly a comparison of* WER *score distributions across human speakers (color-coded) to our artificial method, visualized by the step line. The distributions of* WER *scores are similar as well. Speakers are color-coded. The background step line is the* WER *of the automatic TTS approach.*

A. Further Data Analysis

One potential concern with the synthetically-generated dataset is that ASR systems might be either better or worse at recognizing text-to-speech(TTS) speech. If the ASR system is trained on human data, then it might be an out-of-domain sample, or there might be systematic pronunciation issues that lower ASR accuracy. Alternatively, TTS-generated speech might prove more regular or cleaner than human speech, so an ASR system may produce a higher transcription accuracy on this data. Thus, we determine the distributional overlap between the ASR output on both the synthetic and natural data.

We compare BLEU scores [22] between the gold standard data and the decoded data for between the human and synthetic data variations. By using BLEU scores, which capture n-gram overlap between the target and source text, we can compare the variance in ASR between the two datasets. Figure 2 illustrates this variance. Additionally, Figure 3 shows the comparison of Word Error Rate (WER). Human data has more instances of higher WER and lower BLEU scores than the auto-generated data on the same questions; however, the two sources of speech data generally follow a similar distribution and our results are comparable in accuracy to our synthetic data. Therefore, we conclude that our method serves as a good approximation for the task, which allows weak supervision to work.

B. Negative Results

Alternative methods were applied to mitigate ASR-induced noise in the course of experimentation, including noisy channel techniques typically used in Information Retrieval and lattice-structured Recurrent Neural Networks. For completeness, we discuss the results of these two experiments in this section. While neither method provided an improvement on the question answering task, their discussion might prove useful for future research.

B.1. Noisy Channel Expansion

In both Information Retrieval and NLP it is often useful to model processes that induce noise using Shannon's noisy channel model [23]. We know the answer would be predictable if we had access to certain words in the original question. The noisy channel model allows us to reconstruct the original data as cleanly as possible by modeling the process by which noise was induced, in this case the trip from text to speech and back to text. We propose two forms of query expansion based on this model, both of which are typically used in Cross Language Information Retrieval.

The first model uses IBM Model 3 to generate an alignment table between the corrupted ASR data and the original text data. The alignment table serves as the underlying corruption model which we are aiming to reverse. We use our training data a second time and generate possible word candidates that were missed during decoding.

The second model uses a more robust version of the same Information Retrieval technique looks at two-way translations between ASR and original data based on (Xu, 2008). Whereas the first model included many junk translations—stop-words such as “unk” or “the” would be mapped to a long tail of meaningful words—this version does not suffer from this problem: even if “the” maps to “Monte”, “Monte” does not map back to “the”.

In both cases, the reconstructed data was used to train the DAN model. That neither was able to improve over the confidence modeling DAN indicates that the errors made by the ASR system were likely not recoverable with the translation models we used. This is unsurprising, as many low-frequency important words were mapped to a handful of high-frequency terms, collapsing the space and preventing simple recoverability.

B.2. Lattice-Structured RNN

The confidence models are not calculate on a full lattice, and hence cannot not reconstruct alternate paths in situations with low confidences. A more complex model can ingest the entire lattice, and not the top word prediction. The lattice can update multiple words needed, as their relationships are preserved. "Leo Patrick" can now be reinterpreted as "Cleopatra", as the lattice relationship allows alternate paths to be explored. The confidence values provide additional value about what path to follow within a lattice.

We produce three variations:

1. A "lattice" LSTM that consumes the full lattice by linearizing the graphs with a topological sort and feeding it through a normal LSTM.
2. A lattice LSTM without confidences. This network only sees the word vectors when consuming the lattice structure.
3. A lattice LSTM with confidences integrated as features. The confidences are concatenated to the word vector inputs.

This sequence demonstrates the gain from each part of the model. The first tests the benefit of additional data. The second tests the benefit of the structure of this data. The third tests the importance of the confidence of each item in the data.

Unfortunately, none of these experiments outperformed the confidence augmented DAN. These may be due to instability or training issues, however.