Chapter 1: See Other File

Chapter 2: See Other File

Chapter 3: Automatic Data Generation from a Found Source

The fastest method of creating large neural-scale datasets is through automatic generation of synthetic data. This chapter discusses a large audio dataset created with Text-To-Speech technology, and the limitations thereof (Section 3.1).[1] We reach similar conclusions from another automatically-created dataset, intended for evaluating co-reference (Chapter **??**). Both datasets, while large, are not real-

[1] Denis Peskov, Joe Barrow, Pedro Rodriguez, Graham Neubig, and Jordan Boyd-Graber. 2019. Mitigating noisy inputs for question answering. In Conference of the International Speech Communication Association. Peskov is responsible for the data creation, the gathering of recordings from users, running the neural models, figure and table design, and paper writing.

istic and motivate using humans for **generation** (Chapter **??**). Furthermore, both datasets ultimately depend on **experts** for validation (Chapter **??**).

3.1 Automated Data Creation for Question Answering

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 (Ferrucci, 2010) on Jeopardy! and QB matches between machines and trivia masters (Boyd-Graber et al., 2018) provide text data for machines while humans listen. A fair assessment of an Artificial Intelligence's ability to answer human trivia questions would subject the machine to speech input, akin to how typical human would process sound.[2]

Hence, computers should be provided with the same audio input that a human would hear. The computer needs a model to decode the audio into textual format to process the audio and answer the question. 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 3.2). 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;, models trained on synthetic data are

[2]An audibly impaired person would be delivered questions in a non-audio medium, but would still experience a cognitive delay, unlike a machine.

applied to human spoken data from QB tournaments and Jeopardy! (Section 3.4.1).

Noisy ASR is particularly challenging for QA systems (Figure 3.1). 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. Additionally, the introduction of distracting words (e.g., "dance") causes QA models to make errors (Jia and Liang, 2017). Key terms like named entities are often missing, which is detrimental for QA (Section 3.2.1).

Previous approaches to mitigate ASR noise for answering mobile queries (Mishra and Bangalore, 2010) or building bots (Leuski et al., 2009) typically use unsupervised methods, such as term-based information retrieval. Our datasets for training and evaluation can produce *supervised* systems that directly answer spoken questions. Machine translation (Sperber et al., 2017) 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 3.3.2), and incorporating the uncertainties of the ASR model directly in neural models (Section 3.3.3). The forced decoding method reconstructs missing terms by using terms audibly similar to the transcribed input. Word-level confidence scores incorporate uncertainty from the ASR system into a Deep Averaging Network, introduced earlier (Background Section **??**). These methods are compared against

baseline methods on our synthetic and human speech datasets for Jeopardy! and QB (Section 3.4).

3.2 Automatically Generating a Speech Dataset

Neural networks require a large training corpus, but recording hundreds of thousands of questions is not feasible. Methods for collecting large scale audio data include Generative Adversarial Networks (Donahue et al., 2018) and manual recording (Lee et al., 2018). For manual recording, crowd-sourcing with the required quality control (speakers who say "cyclohexane" correctly) is prohibitively 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 (Boyd-Graber et al., 2018), each with 4–6 sentences for a total of over 500,000 sentences.[3] We then decode these utterances using the Kaldi chain model (Peddinti et al., 2015), trained on the Fischer-English dataset (Cieri et al., 2004) for consistency with past results on mitigating ASR errors in MT (Sperber et al., 2017). This model decodes enough noise into our data to test mitigation strategies.[4]

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

[4] This model has a Word Error Rate (WER) of 15.60% on the eval2000 test set. The WER increases to 51.76% on our QB data, which contains out of domain vocabulary. Since there is no past work in question answering, we use machine translation as proxy for determining an appropriate Word Error Rate, as intentional noise has been added to this subdomain (Michel and Neubig, 2018; Belinkov and Bisk, 2018). 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% (Sperber et al., 2017). Retraining the model on the QB domain would mitigate this

3.2.1 Why Question Answering is challenging for ASR

Question Answering (QA) requires the system to provide a correct answer out of many candidates based on the question's wording. 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 our ASR system 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 delete unintentionally delete words, which makes the sentences shorter. In our QB data, the average number of words decreases from 21.62 to 18.85 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' becoming "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, the most popular metric for evaluating how important a word is for a document,

noise; however, in practice one is often at the mercy of a pre-trained recognition model due to changes in vocabularies or speakers.

drops from 3.52 to 2.77 per sentence. Examples of this change can be seen in Figure 3.1.

For generalization, we test the effect of noise on two types of distinct questions. QB questions, which are generally four to six sentences long, test 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 (Yamada et al., 2018). 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 (Dunn et al., 2017).

3.3 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. However, a IR approach is more effective on long questions since noisy words are completely avoided during the answer selection process.

3.3.1 IR Baseline

The IR baseline reframes Jeopardy! and QB QA tasks as document retrieval tasks 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 new, unseen questions are treated as queries, and documents are scored using BM25 (Ramos, 2003; Robertson et al., 2009). We implement this baseline with Elastic Search and Apache Lucene.

3.3.2 Forced Decoding

We have systematically lost information due to ASR decoding. 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.

As a first step, we explored commercial 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.

Therefore, we use Kaldi (Povey et al., 2011) 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 3.1). Each item in the sequence represents a word and has a confidence in

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 ...

Table 3.1: 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. Our confidence model and forced decoding approach could be used for such data in future work.

range [0, 1] correspond to the respective.

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 generic statement of uncertainty. So by removing all reference to <*unk*> in the model, we force the system to decode "Louis Vampas" as

"Louisiana" rather than <*unk*>.[5] 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.

3.3.3 Confidence Augmented DAN

The errors introduced by ASR can hinder sequence neural models as key phrases are potentially corrupted. We modify the original DAN model, introduced in Background Section **??**, to use word-level confidences from the ASR system as a feature and be robust to corrupted phrases. 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 $\mathbf{c}$, 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 3.1, "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

[5]More specifically, <*unk*> is removed from the Finite State Transducer, which sets the input/output for the ASR system.

calculating the DAN average. In the previous example—"for ten points"—"for" and "ten" are frequently occuring words and have a confidence of .935, while "points" has a lower confidence of .871. The next word—"same"—should be "name" and hence the embedding referenced is incorrect. But, the lower confidence of .617 for this prediction decreases the overall weight of the embedding in the model.

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:

$$f(\mathbf{c}) = \mathbf{W^{(c)}}\mathbf{c} + \mathbf{b^{(c)}} \tag{3.1}$$

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

$$\mathbf{r^{**}} = \frac{\sum_i^N \mathbf{E}[w_\mathrm{i}] * f(c_\mathrm{i})}{N}. \tag{3.2}$$

In this model, we replace the original encoder $\mathbf{r}$ 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 called our "Confidence Model".

Architectural decisions are determined by hyperparameter sweeps. They include: having a single hidden layer of 1000 dimensionality for the DAN, multiple drop-out, batch-norm layers, and a scheduled ADAM optimizer. Our DAN models train until convergence, as determined by early-stopping. Code is implemented in

PyTorch (Paszke et al., 2017), with TorchText for batching.[6]

3.4 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 3.2) establish realistic goals: a DAN trained and evaluated on the *same train and dev set*, only 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 QB 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

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

question level in QB; 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 IR 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.

3.4.1 Qualitative Analysis & Human Data

The synthetic dataset facilitates large-scale machine learning, but ultimately we care about performance on human data. For QB 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. Table 3.3 provides examples of variations. For Jeopardy! we manually parsed a complete episode by question.

The predictions of the regular DAN and the confidence version can differ. As one example, 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. As another example the model in Table 3.3 predicts "London" if "beaumont" and "john" are preserved, but "Baghdad" if the proper nouns, but not "palace" and "city", are lost.

3.5 Implications of Automation

The advantages of this method are cost and scalability, which is demanded by the current paradigm of neural models. This however comes at the expense of quality. A limitation of our past work in **automation** is generalization: text-to-speech only has female voices and is consistently decoded, while the voices of real humans are decoded with large variations. Unseen data points are likely to confound a model trained on unnatural data. Additionally, automated data creation still depends on having quality source data, that often has to come from expert users. In this project, we record **found** questions that were already written by Quizbowl **experts**. Writing hundreds of thousands of our questions would not have been tractable. Hence, expert design is necessary for automation, as implemented in our other project (Chapter **??**).

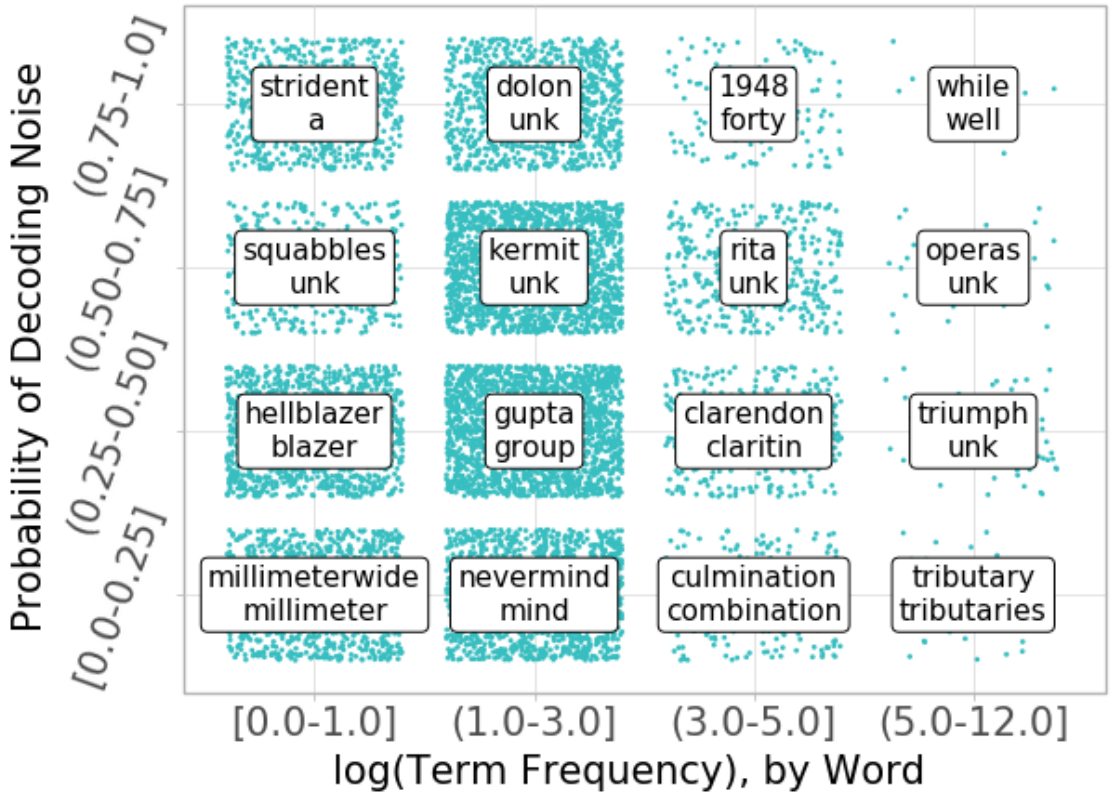

Figure 3.1: ASR errors on QA data: original spoken words (top of box) are garbled (bottom). While many words become into "noise"—frequent words or the unknown token—consistent errors (e.g., "clarendon" to "clarintin") can help downstream systems. Additionally, words reduced to $<unk>$ (e.g., "kermit") can be useful through forced decoding into the closest incorrect word (e.g., "hermit" or even "car").

	QB				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

Table 3.2: 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 in certain cases. IR and DAN models trained and evaluated on clean data are provided as a reference point for the ASR data.

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

Table 3.3: Variation in different speakers causes different transcriptions of a question on Oxford. The omission or corruption of certain named entities leads to different answer predictions, which are indicated with an arrow.

Bibliography

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of the International Conference on Learning Representations*.

Jordan Boyd-Graber, Shi Feng, and Pedro Rodriguez. 2018. *Human-Computer Question Answering: The Case for Quizbowl*. Springer Verlag.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The fisher corpus: a resource for the next generations of speech-to-text. In *Proceedings of the Language Resources and Evaluation Conference*.

Chris Donahue, Bo Li, and Rohit Prabhavalkar. 2018. Exploring speech enhancement with generative adversarial networks for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new Q&A dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

David A. Ferrucci. 2010. Build Watson: an overview of DeepQA for the Jeopardy! challenge. In *19th International Conference on Parallel Architecture and Compilation Techniques*, pages 1–2.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of Empirical Methods in Natural Language Processing*.

Chia-Hsuan Lee, Shang-Ming Wang, Huan-Cheng Chang, and Hung-Yi Lee. 2018. Odsqa: Open-domain spoken question answering dataset. In *2018 IEEE Spoken Language Technology Workshop (SLT)*.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2009. Building effective question answering characters. In *Proceedings of the Annual SIGDIAL Meeting on Discourse and Dialogue*.

Paul Michel and Graham Neubig. 2018. Mtnt: A testbed for machine translation of noisy text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Taniya Mishra and Srinivas Bangalore. 2010. Qme!: A speech-based question-answering system on mobile devices. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *Conference on Neural Information Processing Systems: Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*.

Vijayaditya Peddinti, Guoguo Chen, Vimal Manohar, Tom Ko, Daniel Povey, and Sanjeev Khudanpur. 2015. Jhu aspire system: Robust lvcsr with tdnns, ivector adaptation and rnn-lms. In *Automatic Speech Recognition and Understanding (ASRU), IEEE Workshop on*, pages 539–546.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Nagendra Goel, Mirko Hannemann, Yanmin Qian, Petr Schwarz, and Georg Stemmer. 2011. The Kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the International Conference of Machine Learning*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the Association for Computational Linguistics*.

Ikuya Yamada, Ryuji Tamaki, Hiroyuki Shindo, and Yoshiyasu Takefuji. 2018. Studio Ousia's quiz bowl question answering system. In *NIPS Competition: Building Intelligent Systems*, pages 181–194.