INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Symbol sequence search from telephone conversation

Masayuki Suzuki[1], *Gakuto Kurata*[1], *Abhinav Sethy*[2], *Bhuvana Ramabhadran*[2], *Kenneth W Church*[2], *Mark Drake*[3]

[1] IBM Watson Multimodal, IBM Research - Tokyo, Chuo-ku, Tokyo, 103-8510, JP
[2] IBM Watson Multimodal, IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598, US
[3] IBM Systems, Lenexa, KS, 11200, US

szuk@jp.ibm.com, gakuto@jp.ibm.com, asethy@us.ibm.com, bhuvana@us.ibm.com, kwchurch@us.ibm.com, mdrake@us.ibm.com

Abstract

We propose a method for searching for symbol sequences in conversations. Symbol sequences can include phone numbers, credit card numbers, and any kind of ticket (identification) numbers and are often communicated in call center conversations. Automatic extraction of these from speech is a key to many automatic speech recognition (ASR) applications such as question answering and summarization. Compared with spoken term detection (STD), symbol sequence searches have two additional problems. First, the entire symbol sequence is typically not observed continuously but in sub sequences, where customers or agents speak these sequences in fragments, while the recipient repeats them to ensure they have the correct sequence. Second, we have to distinguish between different symbol sequences, for example, phone numbers versus ticket numbers or customer identification numbers. To deal with these problems, we propose to apply STD to symbol-sequence fragments and subsequently use confidence scoring to obtain the entire symbol sequence. For the confidence scoring, We propose a long short-term memory (LSTM) based approach that inputs word before and after fragments. We also propose to detect repetitions of fragments and use it for confidence scoring. Our proposed method achieves a 0.87 F-measure, in an eight-digit customer identification number search task, when operating at 20.3% WER.

1. Introduction

As the performance of automatic speech recognition (ASR) steadily improves [1, 2], the scope of ASR applications is expanding. There are already many successful applications of spoken interaction, such as IoT applications, dictation, voice search, and spoken dialog systems. In addition, there are several applications of ASR operating on the spontaneous speech between humans, such as spoken term detection, speech to speech translation, call center interaction and meeting summarization. However, these applications operate at significantly higher Word Error Rate (WER) levels and recent work has focussed on improving these applications. One such example is telephone agent assists that use question answering and summarization of telephone conversations [3, 4] are promising applications of ASR to spontaneous speech.

In this paper, we tackle the task of searching for symbol sequences in spontaneous conversation, which is one of the key components for many ASR applications to spontaneous speech. Symbol sequences such as phone numbers, credit card numbers, ticket numbers and customer identification numbers are often communicated in business telephone conversations, and related applications often need to extract such information both off-line and in real-time. Examples of these applications include, compliance applications and automatic population of key information such as, customer identification and problem ticket numbers into a database during a transaction.

Table 1: *Example of transcribed conversation where customer identification number 12345-678 is communicated. A is a customer and B is a telephone agent.*

A: my number is uh one two three four five
B: one two three four five
A: one two ... okay right the next code is uh six seven eight
B: okay six um six seven eight

Very often, grammar based search, are used in spoken term detection (STD), for symbol sequence search task [5]. However, we wish to highlight two key problems that have to be addressed, given the spontaneous nature of human conversations. Table 1 shows an example of a conversation where an eight digit customer identification number is communicated. The following problems arise when we try to apply an STD with an eight continuous digit grammar.

The first problem is that, often, the entire symbol sequence may not be spoken continuously. This happens often, especially when the number of symbols is longer than one can hold in their memory or when separators are inserted in the sequence, such as, "12345-678." In the example in Table 1, "one two ... okay right the next code is uh" occurs between two fragments. The words inserted between fragments depend on the context of the conversation and are not easily captured by a mumble model. The problem is further complicated when we have to use monaural recordings of telephone conversations [6]. We would have to find the sequence *12345* from "my number is uh one two three four five one two three four five one two ... okay" in the example. While technologies such as speaker diarization are useful, they are not good enough to assist with the problem at-hand today.

The second problem is that a grammar for symbol sequence cannot distinguish two different symbol sequences such as that of a credit number from that of a customer identification number. The number of consecutive symbols can be a clue to distinguish them, but it is not enough as there may be many hesitations and repetitions. Furthermore, this gets complicated when there are many ASR errors, which is typical in call-center conversations based on the location from which the call is made and the compression (codec) used to record/process the audio.

To deal with these two issues, we propose an algorithm to identify whole symbol sequences from scattered fragments of symbol sequences in conversation by leveraging the context

http://dx.doi.org/10.21437/Interspeech.2017-904

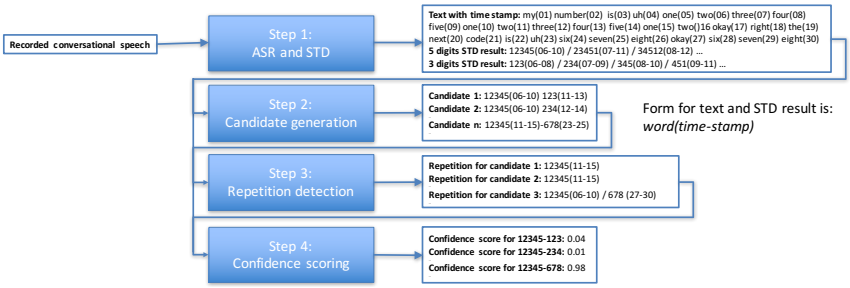

Figure 1: *Flowchart of our symbol-sequence search method*

words around the fragments. It consists of four steps:

1. Application of a grammar-based STD to fragments of symbol sequences. For example, five- and three-consecutive digits grammar can be used to extract fragments for the example case (12345-678).
2. Combination of extracted fragments to generate candidates of symbol sequences.
3. (Optional) Detection of repetition of the fragments for each candidate from the transcribed words preceding and succeeding the fragments.
4. Estimation of confidence scores for candidates and selection of the best candidate(s) based on a threshold.

Using fragments as units for STD enables us to find entire symbol sequences even if arbitrary words are inserted between fragments. For confidence scoring, we propose a long short-term memory (LSTM) [7] based approach that takes advantage of the words in the context preceding and succeeding candidate fragments. This approach is capable of distinguishing different types of symbol sequences. Detecting repetitions is optional, but it improves confidence scoring because repetitions are good clues [8]. One has to keep in mind that call-center agents are often trained to repeat symbol sequence to avoid communication errors.

We applied our proposed method to a customer identification number search task. Our proposed method achieved a 90.0% precision rate and 85.6% recall rate in an eight-digit customer identification number search task in which the word error rate of the ASR system was 20.3%.

The rest of this paper is organized as follows. We introduce some related work in section 2. We briefly introduce the STD method used in this paper in section 3. We explain our proposed method in section 5 and evaluate it in section 5. Section 6 concludes this paper.

2. Related work

There are many studies on STD [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], which can be applied for searching fragments of symbol sequence in our method. As far as we know, there is no other research on symbol sequence search task using STD.

There are many studies for named entity (NE) extraction, which is a task to extract NE such as persons, products, amounts, and so on from text [20, 5, 21, 22, 23, 24, 25, 26, 27]. This task is related to the symbol sequence search task because some of studies work on a NE task from speech and some of studies deal with telephone numbers as NE [5]. State-of-the-art system formularize the NE task as tagging of each word in sequences using recurrent neural network [25, 26]. These methods works well for the NE task from speech such as voice mail, where symbol sequences are observed continuously [5]. However, if symbol sequences are observed in sub sequences, we need another model to select the best entire symbol sequences as we proposed in this paper. In addition, it cannot leverage repetition information.

A main contribution of this paper is to propose an LSTM-based confidence scoring method leveraging repetition for candidates of entire symbol sequences. Extraction of fragments of symbol sequences can be solved by applying STD or NE extraction methods. Though we used grammar-based STD for fragment extraction in this paper, we can switch it to NE extraction methods from speech.

[28] proposes a LSTM-based target-dependent sentiment classification method, where the task is to estimate the sentiment of target phrases such as “battery life” with the help of context words in a sentence. This model and ours calculate the score of a target phrase from context words around it. Our method uses a similar LSTM architecture, although its aims is quite different. [28] showed that their LSTM-based approach is better than SVM with bag-of-features. We confirmed that the LSTM-based approach outperform the SVM-based approach for our task in our preliminary experiments.

3. Spoken term detection

Research on spoken term detection (STD), which is also known as keyword search (KWS), has been substantially advanced by competitive evaluations such as the NIST STD evaluation [9], DARPA RATS (Robust Automatic Transcription of speech) program [10], and IARPA BABEL program [11, 29]. In this paper, we use word-based STD component proposed in these evaluations.

First, we generate word lattices with the ASR system. We use a Weighted Finite Transducer based lattice indexing scheme that includes start-time, end-time and lattice posterior probability for each word [30, 29]. Given a keyword or grammar, we create a simple finite state acceptor of the keyword or grammar, and compose it with the lattice-based index to obtain all occurrences of the keyword or grammar in the lattice, along with the start-time, end-time, and posterior probability of each occurrence. We can adjust precision and recall by performing YES/NO decisions based on the posterior probabilities.

4. Proposed method

Fig. 1 shows the flowchart of our proposed method. Although our method supports any type of symbol sequence search, the following explanation focuses on a search for customer identity numbers consisting of five and three consecutive digits like 12345-678 for the sake of simplicity.

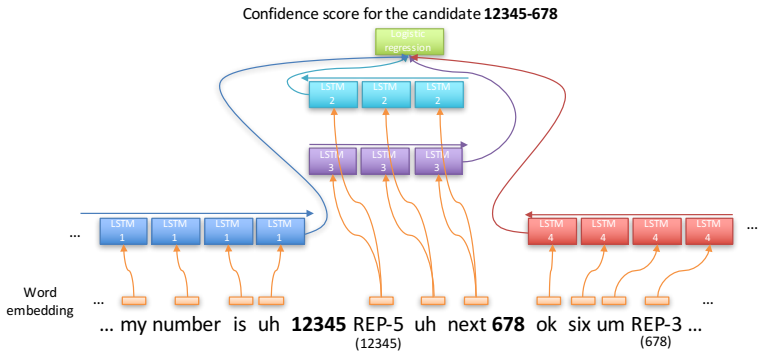

Figure 2: *LSTM-based confidence scoring for candidates in a transcribed text*

4.1. ASR and STD

In the first step, we use an ASR system operating at WER of approximately 20% trained on call-center conversations. A lattice-based, word/time index of the call-center conversations is generated as described in Section 3. In the provided example, we query five-digit and three-consecutive digit grammars. Therefore, we query the index for all five and three digit sequences represented by our query grammars. The results of this STD operation serve as input to the next step. However, if the task is identication of telephone numbers, then the symbol sequence query can comprise of a grammar that includes valid area codes, followed by a seven-digit sequence. It can be seen that the proposed search that converts these grammars to WFSAs is general enough and can be used for many types of symbol sequences.

4.2. Candidate generation

The next step is the generation of *valid* candidates by combining the five- and three-digit fragment results detected in Section 4.1. A superset of all possible andidates includes any combination of the results obtained by the separate searches on the fragment queries done in the first step in Section 4.1. During this step, invalid sequences from that superset are eliminated. In thsi example, we choose to keep results for the five-digit and three-digit searches which do not overlap in time. We also enforce the rule that the detected candidate for the five-digit search should occur before the three-digit candidate it is combined with. We also delete candidate combinations where the time difference between the end time of five-digit candidate and the start time of three-digit candidate is greater than a threshold (say, 30 seconds). This process generates a *valid* candidate set from the superset produced by the first step.

4.3. Repetition detection

In this step, we detect repetitions of fragments for each candidate. Call-center agents are often trained to repeat symbol sequences stated by the customer, to avoid communication errors. Thus, repetitions of fragments are seen frequently and are often used to correct the symbol sequence errors, both, by the customer and the agent. Although this step is optional, using information from repetitions improves confidence scoring in the next step.

We use the superset of candidates from the STD system in Section 4.1 to detect repetitions. Fragments which have the same symbol sequence with candidates are regarded as repetitions. To avoid the adverse effects of ASR errors, it is possible to allow mismatches to some extent such as, deletion, substiton and insertions of one or two symbols in a sequence. Increasing the mismatches beyond that renders the problem intractable. Note that we use not only repetition after fragments but also repetition before fragments.

If a repetition is found, we replace it with a special word, such as, “REP-5” and “REP-3” to distinguish it from the other symbol sequences. If these special words (REP-5 and REP-3) are observed near the fragments, the candidate more likely to be a correct detection.

4.4. Confidence scoring

The final step is to calculate confidence scores for all the valid candidates. The candidate that has the largest confidence score or a set of candidates whose confidence scores are larger than a threshold are selected as the final output of the system. We devised a LSTM-based model to calculate confidence scores. Fig. 2 illustrates this process. Word embeddings [31] corresponding to words preceding the detected fragments are used as input to the forward-direction LSTM, while words succeeding the detected fragments are used as input for the backward-direction LSTM. Logistic regression is used to calculate the posterior probability of the candidate being correct and this serves as the confidence score. The number of LSTMs is twice the number of fragment sequences being deteced as we use both forward and backward word contexts for each sequence. In the example used in this paper, we use four LSTMs: forward-direction LSTM for words preceding five-digit fragments, backward-direction LSTM for words succeeding five-digit fragments, forward-directionLSTM for words preceding three-digit fragments, and backward-directionLSTM for words succeding three-digit fragments. The final states of these LSTMs are concatenated and serve as input to the logistic regression layer. The output of this layer is the posterior probability of the candidate being correct. If a repetition is detected, a word embedding of the special word (REP-5 or REP-3) is used as input to the LSTM.

The LSTMs are trained on ASR transcriptions of call-center conversations. Valid symbol sequences are manually annotated in this data to serve as the ground truth. We generate positive and negative lables for these symbol sequences by extracting a superset of candidates using the previously described steps and assigning a label of ‘1’ or ‘0’ to these candidates.

5. Experiments

5.1. Data

We used an in-house call-center data set for these experiments. To the best of our knowledge, there is no publicly available

database that is suitable for the evaluation of this task. The data consists of monaural recordings of 2,117 calls recorded in a call center. All the calls included a customer identification number consisting of five and three digits, such as, 12345-678. We manually annotated the correct customer identification number for each call to serve as ground truth.

5.2. Model

For ASR and STD, we used a speaker-independent convolutional neural network (CNN) based acoustic model. The CNN was trained with 1,975 hours of telephony speech data by using the cross entropy and the sMBR criterion [32], followed by supervised adaptation with 25 hours of in-domain data [33]. The CNNs operated on blocks of 11 consecutive 40-dimensional logmel frames augmented with first and second derivatives over 9×9 convolution windows. The logmels were globally variance normalized and mean normalized per utterance. The configurations of the convolution and pooling layers were] taken from [34]. The size of the parameters for the CNN was $[243 \times 128, 1536 \times 256, 2048 \times 2048, 2048 \times 2048, 2048 \times 2048, 2048 \times 2048, 2048 \times 2048, 2048 \times 512, 512 \times 9300]$. The vocabulary comprised of 250K words, and the language model (LM) was a 4-gram LM with 200M n-grams. The LM was also adapted using manual transcriptions of the 25 hours of in-domain data by interpolation. The word error rate with the test data for ASR was 20.3%.

For training the LSTM-based confidence scorer, we performed a five-fold cross validation using 1,694 calls as training data. The time consuming nature of the manual annotations of symbol sequences forced this data set to be small. We selected 400 words from these transcripts based on the number of occurences. These included digits and other common words such as, *branch code, repeat, right, okay* and other function words. Word embeddings (8-dimensional word vector) were generated for these top 400 words in the following manner. The remaining words in the ASR's vocabulary were mapped to unknown words with specific embeddings associated with it. Each word is represented by a one-shot 1×400 dimensional vector that is fed into a 400×8 linear layer. The 8-dimensional embeddings are subsequently fed into an LSTM layer with 8 units. The output targets are the labels generated by the manual annotations (1 or 0). Four such LSTMs, two separate unidirectional LSTMs for each direction for both, the five-digit and three-digit sequences are built. The embeddings are therefore trained jointly with the confidence scorer. Special words representing repetitions (REP-5 and REP-3) are also represented by separate embeddings. We used 32 words preceding and succeeding fragments as unput for the LSTMs. Thus, the number of dimensions of the input vector for the logistic regression layer was $8 \times 4 = 32$. All the parameters, including the word embeddings, were randomly initialized and trained using back propagation through time.

For applications that need only the 1-best candidate, we selected the best candidate that had the highest confidence score given by the LSTM-based model. We also selectedadditional candidates that had confidence scores higher than a threshold to adjust the precision and recall rate depending on what the application's needs were.

5.3. Results

The blue curve in Fig. 3 show the precision-recall curve for various thresholds. If the application requires 90.0% precision, we can use 0.11 as a threshold, where precision rate = $1812/2014 = 90.0\%$ and recall rate = $1812/2117 = 85.6\%$.

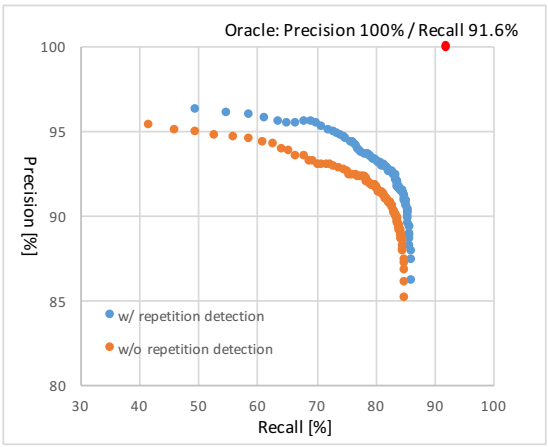

Figure 3: *Precision recall curve of our method*

F-measure of this point is 0.87. Similarly, if the application requires 95.0% precision, we can use 0.86 as the threshold, where precision rate = $1528/1608 = 95.0\%$ and recall rate = $1528/2117 = 72.2\%$.

The red dot in Fig. 3 shows the performance of *oracle* candidate selection. This implies that for 91.6% of the test data, the correct customer identification number is already included in the candidates. We need to improve the ASR and STD systems further to correctly retrieve the remaining 8.4% of the queries which do not include the correct answer in the list of candidates.

The orange curve in Fig. 3 show the precision recall curve when we skipped the repetition detection step. The other conditions are exactly the same as for the blue curve. By comparing the two plots, we can see the positive effect of using repetition detection and subsequently using the detected repetition as an additional input to the LSTM-based confidence scorer.

For comparison, we evaluated a simple rule-based symbol sequence extractor. We developed a grammar which accepts sequences of five consecutive digits, words often observed in the middle, followed by three consecutive digits. We performed STD with this grammar, and selected the candidate with the highest posterior probability. Results of the rule-based extractor were not good: precision is 428/2213 = 19.3%, recall is 428/2771 = 15.4%, and F-measure is 0.17. One main reason for the poor performance is the high variability of words between digits that a grammar cannot handle. Though we may be able to improve performance further by finetuning the grammar, we can say that the LSTM-based confidence scorer will beat the grammar-based system significantly, based on the large difference in performance.

6. Conclusion

We have proposed a method for searching symbol sequences in conversations. This is a critical component of call-center analytics. Our proposed method applies ASR and STD to fragments of symbol sequences, generates candidates, detects repetitions, and calculates confidence measures for the candidates. It achieves a 0.87 F-measure on identifying customer identification numbers, operating with an ASR system at 20.3%. The proposed work is easily extendible to other symbol sequences needed for not just call-center anyaltics but any similar application.

7. References

[1] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. C. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim, B. R. Roomi, and P. Hall, "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[2] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[3] S. Furui, T. Kikuchi, Y. Shinnaka, and C. Hori, "Speech-to-text and speech-to-speech summarization of spontaneous speech," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 4, pp. 401–408, 2004.

[4] S. Chopra, M. Auli, A. M. Rush, and S. Harvard, "Abstractive sentence summarization with attentive recurrent neural networks," *Proceedings of NAACL-HLT16*, pp. 93–98, 2016.

[5] G. Zweig, J. Huang, and M. Padmanabhan, "Extracting caller information from voicemail." in *INTERSPEECH*, 2001, pp. 291–294.

[6] M. Suzuki, G. Kurata, T. Nagano, and R. Tachibana, "Speech recognition robust against speech overlapping in monaural recordings of telephone conversations," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5685–5689.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper, "Enriching speech recognition with automatic detection of sentence boundaries and disfluencies," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 5, pp. 1526–1540, 2006.

[9] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *Proc. sigir*, vol. 7, 2007, pp. 51–57.

[10] L. Mangu, H. Soltau, H.-K. Kuo, B. Kingsbury, and G. Saon, "Exploiting diversity for spoken term detection," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8282–8286.

[11] J. Cui, X. Cui, B. Ramabhadran, J. Kim, B. Kingsbury, J. Mamou, L. Mangu, M. Picheny, T. N. Sainath, and A. Sethy, "Developing speech recognition systems for corpus indexing under the iarpa babel program," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6753–6757.

[12] D. Vergyri, I. Shafran, A. Stolcke, V. R. R. Gadde, M. Akbacak, B. Roark, and W. Wang, "The sri/ogi 2006 spoken term detection system." in *Interspeech*. Citeseer, 2007, pp. 2393–2396.

[13] I. Szöke, M. Fapšo, M. Karafiát, L. Burget, F. Grézl, P. Schwarz, O. Glembek, P. Matějka, J. Kopecký *et al.*, "Spoken term detection system based on combination of lvcsr and phonetic search," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2007, pp. 237–247.

[14] H.-y. Lee, T.-w. Tu, C.-P. Chen, C.-Y. Huang, and L.-S. Lee, "Improved spoken term detection using support vector machines based on lattice context consistency," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5648–5651.

[15] M. S. Seigel, P. C. Woodland, and M. Gales, "A confidence-based approach for improving keyword hypothesis scores," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8565–8569.

[16] G. Chen, S. Khudanpur, D. Povey, J. Trmal, D. Yarowsky, and O. Yilmaz, "Quantifying the value of pronunciation lexicons for keyword search in lowresource languages," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8560–8564.

[17] L. Haiyang, T. Zheng, G. Zheng, and H. Jiqing, "Confidence measure based on context consistency using word occurrence probability and topic adaptation for spoken term detection," *IEICE TRANSACTIONS on Information and Systems*, vol. 97, no. 3, pp. 554–561, 2014.

[18] J. Trmal, G. Chen, D. Povey, S. Khudanpur, P. Ghahremani, X. Zhang, V. Manohar, C. Liu, A. Jansen, D. Klakow *et al.*, "A keyword search system using open source software," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 530–535.

[19] L.-s. Lee, J. Glass, H.-y. Lee, and C.-a. Chan, "Spoken content retrieval—beyond cascading speech recognition with text retrieval," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1389–1420, 2015.

[20] D. Miller, S. Boisen, R. Schwartz, R. Stone, and R. Weischedel, "Named entity extraction from noisy input: speech and ocr," in *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics, 2000, pp. 316–324.

[21] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003, pp. 142–147.

[22] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[23] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *arXiv preprint arXiv:1511.08308*, 2015.

[24] Y. Ma, J.-j. Kim, B. Bigot, and T. M. Khan, "Feature-enriched word embeddings for named entity recognition in open-domain conversations," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 6055–6059.

[25] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder lstm for semantic slot filling," *arXiv preprint arXiv:1601.01530*, 2016.

[26] S. Zhu and K. Yu, "Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding," *arXiv preprint arXiv:1608.02097*, 2016.

[27] K. Church, W. Zhu, and J. Pelecanos, "C2d2e2: Using call centers to motivate the use of dialog and diarization in entity extraction," *EMNLP 2016*, p. 35, 2016.

[28] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective lstms for target-dependent sentiment classification," *arXiv preprint arXiv:1512.01100*, 2015.

[29] J. Cui, B. Kingsbury, B. Ramabhadran, A. Sethy, K. Audhkhasi, X. Cui, E. Kislal, L. Mangu, M. Nussbaum-Thom, M. Picheny *et al.*, "Multilingual representations for low resource speech recognition and keyword search," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 259–266.

[30] D. Can and M. Saraclar, "Lattice indexing for spoken term detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[31] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[32] G. Saon, H.-K. J. Kuo, S. Rennie, and M. Picheny, "The ibm 2015 english conversational telephone speech recognition system," *arXiv preprint arXiv:1505.05899*, 2015.

[33] M. Suzuki, R. Tachibana, S. Thomas, B. Ramabhadran, and G. Saon, "Domain adaptation of cnn based acoustic models under limited resource settings," *Interspeech 2016*, pp. 1588–1592, 2016.

[34] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.