DOMAIN ADAPTATION FOR PARSING IN AUTOMATIC SPEECH RECOGNITION

Alex Marin, Mari Ostendorf

University of Washington, Seattle, WA 98195

ABSTRACT

This paper addresses the problem of adapting a parser trained on out-of-domain data for use in automatic speech recognition (ASR) rescoring and error detection tasks. Using a self-training approach and adaptation with weakly-supervised data, we obtain improvements in ASR rescoring of confusion networks. Features extracted from the parser output are also used to improve detection of general ASR errors and out-of-vocabulary word regions in conjunction with a maximum entropy classifier.

Index Terms— Parsing, speech recognition, error detection, OOV detection

1. INTRODUCTION

Several studies have investigated the use of parsing for improving speech recognition. Language models containing syntactic information have been used successfully for decoding [1, 2, 3], discriminative n-best rescoring [4], and joint parsing and speech recognition [5]. Improvements in word error rates have been obtained in several studies, for both read news [6] and conversational speech [5, 7, 8, 9]. (Although conversational speech lacks the complexity of structure of formal language, it has greater acoustic confusability that even simple syntactic structure can help resolve.) These good results are mostly obtained in conditions where treebanked data is available that is reasonably well matched to the target domain. As for many language processing tasks, parsing performance degrades when there is a domain mismatch, which is the case for many scenarios.

In our prior work, we investigated the use of parsing for out-of-vocabulary (OOV) word detection in speech recognition [10]. In that study, we found that a parser trained on a mismatched domain was still useful for identifying OOVs, even though the same parser led to degraded speech recognition transcripts if used in a rescoring framework. Motivated by work on parsing that shows a benefit from semi-supervised training [11, 12, 13], this work explores its use both for improving speech recognizer accuracy as well as OOV (and error) detection. In addition, we look at weakly supervised methods that leverage known labels in the OOV/error detection task.

Because of our interest in both error detection and correction with the parser, our framework involves rescoring a word confusion network [14], rather than the single hypotheses or lattices used in other work when separately addressing recognition or OOV detection. The use of confusion networks introduces a different type of mismatch with respect to standard treebanked data, so we also explore semi-supervised learning to address this issue.

In the next sections, we outline the overall system architecture and describe the domain differences, followed by a more detailed description of the system and adaptation strategies. Experimental results are reported for the English side of recognition in a speech translation task, showing a significant benefit to semi-supervised training for both transcription and OOV/error detection.

This material is based upon work supported by DARPA under Contract No. HR0011-12-C-0016 (subcontract 27-001389). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

2. TASK AND SYSTEM OVERVIEW

For both error and OOV detection we employ a three-stage process, modeled after the approach used in [10]. The system takes in ASR output represented as a word confusion network (WCN), i.e. a sequence of slots, with each slot containing a list of words (arcs) and their confidences. An initial slot-level classification stage augments the WCN with error or OOV arcs (henceforth referred to as "error" arcs in general for simplicity). A parser stage identifies the optimal path through the WCN according to its parser model, and a second classification stage refines the error predictions. Unlike in [10], the last stage here performs a second round of slot-level classification, with additional features obtained from the parser output (described in more detail in section 3.2). This last stage classifier is similar to the use of a conditional random field with syntactic language model features [15] for OOV detection. The output of the system is a WCN annotated with errors or OOVs, respectively, and rescored to obtain a new best path through the structure.

Since one long word in the lattice representation of ASR output may correspond to a sequence of short words in another path through the lattice, any WCN slot may also include a null arc, corresponding to a null path for that slot. These null paths are terminal nodes from the perspective of the parser, which necessitates some changes to the parser grammar and creates a mismatch with respect to the treebanked data.

Probably a more important mismatch is between the domain of interest (the "target") and the domain (or domains) used to train the parser (the "source"). This mismatch manifests itself both in terms of word usage (with words in the target domain which are not present in the source treebank acting as OOVs from the parser's perspective) and sentence structure (influencing the quality of the higher-level syntactic structures generated by the parser).

The target domain represents a speech-to-speech translation task involving a variety of scenarios from medical aid to reconnaissance, represented by the TRANSTAC corpus, with our experiments focused on American English. The speech consists of relatively short utterances with a low formality level, drawn from two sources: TRANSTAC conversations and SRI recordings of read speech with topics selected to match the TRANSTAC data. TRANSTAC sentences tend to be longer and contain few OOV errors. The SRI utterances contain more OOVs but no disfluencies and are (relatively) shorter. A subset of speech data is used for training WCN aspects of the parse, and additional TRANSTAC language model training data is used for adapting the parser in semi-supervised learning. We train the parser using a treebank drawn from Switchboard [16] and Fisher

[17] sentences, released by the Linguistic Data Consortium.[1] The parse trees were edited to remove punctuation and lower-case all the words for a better match to the ASR output.

While there are substantial differences in speaking style between the target domain corpora and Switchboard, perhaps the more significant difference for the parser is in terms of the vocabulary mismatch, as shown in table 1. The vocabulary mismatch is with respect to the WCN training set. The filled pause and repetition rates in the table indicate differences in disfluencies: the TRANSTAC data is more like human-computer speech, which tends to be less disfluent.

Table 1. Corpus statistics

Statistic	Treebank	LM Train	WCN Train
# Utterances	16519	95916	951
Ave. utt. len.	4.9	18.1	10.3
Voc. size	4233	17728	2062
Voc. overlap	977	1551	2062
% filled pauses	2.7	0.41	2.3
% repetitions	1.3	0.26	0.8

3. INTEGRATION OF PARSING AND SPEECH

In our previous work [10], the parser received as input the WCN representation of ASR output, with the goal of detecting regions in the ASR output which correspond to OOV words. We will follow this approach here, though unlike [10], we generalize the setup to include all error types as an additional task; we also perform rescoring of the WCN for error correction.

We use a modified version of a probabilistic context-free grammar (PCFG) parser with dependency-based rescoring. The PCFG rules learned from treebank corpora are augmented with specific rules required to parse WCN structures. In particular, for each constituent X we add a pair of binary grammar rules, X $\rightarrow$ X NULL and X $\rightarrow$ NULL X, plus a single rule NULL $\rightarrow$ null to generate the terminal corresponding to the null path arcs. The error arcs are generated through two new rules, ERROR $\rightarrow$ ERROR ERROR to grow the error region and ERROR $\rightarrow$ error to generate the terminals, plus a unary rule X $\rightarrow$ ERROR for each constituent X.

The weights on the word and null arcs are obtained directly from the ASR system. The weights on the error arcs are obtained from the first stage error detection system, and will act as a prior on the probability that the original WCN slot does not contain the correct arc. We use a prior from the first stage OOV detection system in conjunction with the full confusion network for training a parser that can detect OOVs. To detect errors, we use a prior from the first stage error detection system and parse a pruned WCN which contains only the 1-best arc and the error arc in each slot.

3.1. Modeling Approach

To score each possible parse, we augment the base PCFG model with a discriminative log-linear model for the additional rules used to handle null arcs and error arcs. The PCFG is learned from counts of rules in the treebank. As in [10], we assign the unary rules NULL $\rightarrow$ null and ERROR $\rightarrow$ error the default probability 1 and all other added rules probability 0.1, then renormalize all the weights.

The log-linear model used for the null arc and error arc rules is structured so that each rule $a \rightarrow b$ used in a parse y corresponds to a feature $\phi(a \rightarrow b)$, with weights for the rules from the original PCFG being fixed to 0. Thus, the probability of a full parse y of the WCN w given model parameters θ is:

$$p(y|w,\theta) = p(y|w,\theta_{\text{PCFG}})p(y|\theta_{\text{LL}}) \quad (1)$$

$$= \prod_{a \rightarrow b \in y} p(b|a,\theta_{\text{PCFG}}) \frac{\exp(\phi(y)\cdot\theta_{\text{LL}})}{\sum_{y'}\exp(\phi(y')\cdot\theta_{\text{LL}})} \quad (2)$$

where y and y' are both parse trees with terminals forming a path through the WCN w.

The log-linear model feature weights θ_{LL} are computed using an averaged online perceptron learning algorithm [18]. The objective of the log-linear model training is task-specific: minimum word error rate or maximum OOV or error detection rate . The parse probability $p(y|\text{WCN}, G)$ is computed using the inside-outside algorithm. Since all the log-linear model features $\phi(a \rightarrow b)$ are local, the log-linear model evaluation can be done during the population of the chart.

3.2. Feature Extraction

In addition to using the parser output to label each slot in the confusion network as containing an error or not, we also extract features from the best tree produced by the parser. In particular, we focus on extracting features which capture differences between a tree produced by a parser which models ASR errors and one that does not, under the assumption that the local structure in error-free regions of the utterance should be similar in the two cases, but the structure in error regions should be different. ASR null arcs will be modeled explicitly in both the error-aware parser and the error-free parser.

3.2.1. Dependency Tuples

We use dependency information to compare the local structure of the two trees. Given a constituent parse tree for an utterance, we obtain a dependency parse tree by running a standard head-finding algorithm [19]. The dependency information allows us to construct a tuple for each slot in the confusion network, containing:

- the word selected by the parser for that slot
- the word selected by the parser for the slot acting as its head
- the syntactic category for each of the two words
- the root of the largest subtree spanning the slot but not its head
- the root of the smallest subtree containing both the slot and its head

For each slot, we compare the two dependency tuples extracted from the two parser models by counting the number of common elements.

3.2.2. Inside Score Pairs

We use inside score information to capture the confidence of the parser regarding the structure surrounding a given WCN slot. For each slot, we are interested in measuring the performance of the non-error parser both in the vicinity of that slot (the local structure) and on the periphery of the error region it corresponds to, if any (the structure overlapping error region boundaries), with the intuition that a difference in confidence between the two parsers in corresponding regions will provide information about whether to trust the error predictions made by the error-aware parser. We compute two inside score-based features:

- local score: the inside score of the smallest non-trivial subtree (i.e. with root higher than pre-terminals) in the non-error tree, with a word in the current slot as one terminal in the tree;

[1] http://catalog.ldc.upenn.edu/LDC2009T01

- boundary score: the inside score of the smallest subtree in the non-error tree that contains the current slot and an error region boundary slot, minimizing over boundary slots.

4. DOMAIN ADAPTATION

We focus on two methods for addressing the domain differences: self-training of the PCFG and weak task supervision for training the log-linear model.

4.1. PCFG Self-Training

We perform self-training to improve the parser vocabulary and sentence structure coverage of the target domain. For each sentence in the target domain, we obtain the highest-scored parse tree, and add it to the treebank if its score is above a certain threshold. We retrain the model after traversing the entire unlabeled corpus, then repeat the procedure. The threshold for each iteration is tuned using a held-out set. The procedure terminates when either no additional trees are added to the treebank or the improvement on the held-out set drops below a pre-set threshold.

A variation on the method allows us to parse entire confusion networks instead of single hypotheses. Since the highest-scoring parse tree may not correspond to the optimal path through the confusion network, we instead select the highest-scoring parse tree with the restriction that the path is optimal relative to the reference string. Preliminary experiments showed that this approach did not outperform the standard method.

4.2. Weak Task Supervision

As we do not have hand-annotated parse trees for the target domain, we instead apply a form of weak, task-driven supervision for training the log-linear component of the parsing model. In all cases we use the word error rate (WER) of the path corresponding to the highest-scoring parse tree for each confusion network as the objective, with variations as follows:

- when the parser is used as a rescoring language model, we restrict the path chosen by the parser to contain only regular words or null arcs;
- when the parser is used for detecting errors, the path may contain error arcs in addition to regular words and null.

The parser model update statistics are accumulated over the course of each iteration through the target domain training data, with a new model generated at the end of the procedure. In practice, only 5-10 iterations are needed to maximize the objective on the development set.

As described previously, we use two parser models, one which can produce trees containing error terminals and one which does not. The two models may be trained independently, starting from the raw PCFG model trained only on the treebank data, or using the non-error model as a starting point for training the additional error-related rule weights.

5. EXPERIMENTS

5.1. Experiment Paradigm

We use confusion networks produced by an extension of the SRI Dynaspeak [20] speaker-independent speech recognition system. Two separate systems are run - a Gaussian mixture model (GMM)-based system and a deep neural network (DNN)-based system. Lattices from the DNN system are converted to confusion networks using the SRILM toolkit [21] and augmented with arcs from the best GMM hypothesis whenever they do not appear in the corresponding DNN-based WCN slot. Slot-level ASR confidences are produced with a DNN system using features related to the confusion network structure and features from a recurrent neural network (RNN) language model [22].

Both slot-level classification stages use a maximum entropy (MaxEnt) classifier built with the MALLET [23] toolkit. The first MaxEnt stage uses as features a set of confusion network structure statistics, including the slot-level posterior mean and variance, the position of the slot in the sentence, the presence of null arcs, and word class features. The set of features used is described in more detail in [10]. We augment this feature set with the DNN-RNN confidence scores and features indicating the presence of arcs from the GMM system in the current, preceding, and following slot. The second MaxEnt stage augments the first stage features with the dependency and inside score features obtained from the parser.

We split the ASR output data 60-20-20 into training, development, and evaluation partitions, with the training portion used for parser self-training and for training the MaxEnt models. We tune hyperparameters for all systems and report all intermediate results on the development set. The eval set is reserved for reporting the final results using the best system configurations.

5.2. Rescoring to Reduce WER

We evaluate the parser's ability to improve ASR output by rescoring the confusion networks, using the path through the confusion network corresponding to the best parse tree as the new best ASR hypothesis. We compare the 1-best obtained from the raw confusion networks (our baseline) with the best paths produced by a parser trained on only the CTS treebank (similar to the configuration in [10], but not directly comparable due to ASR system improvements), as well as parser models adapted via self-training with either or both of the LM training and ASR output ("WCN training") corpora. We also investigate whether explicitly adapting the null rules helps. We use as evaluation measure word error rate (WER), computed using the SCLITE toolkit [24].

The results are summarized in table 2. We see that self-training using the WCN training set outperforms the baseline, whereas using the LM training corpus hurts, though the combination of the two outperforms the other configurations, both when we perform an additional round of log-linear model tuning and with the default null rule weights. Performing the null rule weight adjustments in fact hurts over just modeling their weights using self-training (when available), likely due to overtraining, though the performance drop is lower when both corpora are used in self-training, as a smaller percentage of self-training data will contain null arcs. The best configuration outperforms the no-parsing scenario, demonstrating that adapting the parser model to the target domain allows us to resolve some of the domain differences which made the initial parser model not useful for improving recognition accuracy.

Table 2. Parser rescoring results, dev set WER (baseline: 13.8)

Configuration	PCFG-only	PCFG+DEL rules
No Self-Train	15.1	16.4
LM Train	15.6	18.0
WCN Train	13.5	16.2
LM+WCN Train	**13.4**	**14.0**

5.3. OOV and Error Detection

We evaluate the contribution of parsing to both OOV and error detection, both in using the parser itself as an error (or OOV) predictor, and in using features derived from parse trees for the second round of MaxEnt classification. We report a modified word error rate (denoted as WER*) as well as F-score of the two classification tasks. For WER* results, we collapse all OOV words in the reference strings to a single "OOV" token and similarly mark all OOV (or error) regions detected by the classifiers with the same "OOV" token. By reporting both WER* and F-score, we can better account for the lengths of the errorful regions detected - in particular, whether error regions are too large and end up engulfing other correct words.

We compare self-training results using three configurations: LM training set, ASR WCN training set, or the two sets combined against not performing any self-training. In all cases, after the initial self-training of the PCFG model, additional training of the log-linear model was performed, to obtain a second parser model capable of parsing WCNs augmented with error arcs. The training of the log-linear model was done separately for OOV detection and for general error detection.

Table 3. Best self-training results, dev set OOV detection

SelfTrain	Configuration	WER*	F-score
None	1st ME	12.8	58.9
None	2nd ME	11.8	65.4
LM	2nd ME	11.7	63.0
WCN	2nd ME	11.8	61.7
LM+WCN	2nd ME	**10.7**	**66.9**

Results for OOV detection over the full WCN are presented in table 3. For each self-training data source, we present the best WER* and F-score results. All configurations which involve the parser and 2nd stage of ME classification outperform the initial ME stage alone. Comparing the various self-training configurations, we observe that, as in the rescoring case, each of the LM training set and the ASR WCN training set, in isolation, produce no significant improvement over the parser trained directly on the hand-annotated treebank; however, in combination, the two outperform the case of no self-training, both in WER* and in OOV detection F-score. The same parser model training configuration achieves the best WER* and OOV F-score results.

Table 4. Best self-training results, dev set error detection

SelfTrain	Configuration	WER*	Configuration	F-score
None	1st ME	12.4	1st ME	68.5
None	Parser	11.8	2nd ME	**72.4**
LM	Parser	12.0	2nd ME	**72.4**
WCN	2nd ME	**11.6**	2nd ME	71.9
LM+WCN	Parser	12.1	2nd ME	72.3

Results for error detection over the 1-best path through the confusion network are shown in table 4. As in the case of OOV detection, the second MaxEnt classification which uses parse features outperforms error detection using the initial feature set alone. However, the self-training results are mixed, with the LM-only training matching the configuration with no self-training, and the other two configurations underperforming slightly. On the other hand, when scoring using WER*, we find that the self-training using the ASR WCN set slightly outperforms the other configurations. Furthermore, whereas for OOV detection we found that the best WER was obtained from the 2nd MaxEnt classifier, for 1-best error detection we find that the parser outperforms the corresponding 2nd MaxEnt stage in most cases. We attribute this to the parser having an easier task of discriminating between only two arcs in each "slot" (the 1-best arc or the error arc), whereas in the OOV detection case, parsing with full confusion networks means the parser often has many more choices to consider, especially in regions corresponding to OOVs.

Table 5. Final system results, eval set

System	Resc .	OOV		Error	
	WER	WER*	F-score	WER*	F-score
baseline (1st ME)	14.6	12.6	64.7	12.1	67.1
base parser system	15.9	12.5	65.2	12.0	67.9
best parser system	**14.2**	**11.7**	**65.9**	**11.6**	**68.9**

The final results on the evaluation set are presented in table 5. The best configurations according to tuning on the dev set outperform both the first stage MaxEnt system and the baseline parser and second MaxEnt stage systems. In particular, the rescoring improvement carries to the eval set, whereas the rescored 1-best produced by the baseline parser performed significantly worse than the original WCN 1-best. The baseline parser system outperformed the first stage MaxEnt system in the OOV and error detection tasks, with the best self-training system providing additional improvements.

Table 6. OOV detection word F-scores, grouped by sentence length

Configuration	Sentence lengths (% of total)		
	0-4 (8.8%)	5-14 (73.3%)	15- (17.9%)
1st ME	47.1	55.0	14.0
base parser	53.3	54.4	11.6
base 2nd ME	50.0	59.9	**15.2**
best parser	44.4	29.3	4.5
best 2nd ME	**72.7**	**63.3**	11.7

To examine the contribution of the parser at different sentence complexity levels, we evaluate the performance of the parser-enhanced OOV detector systems, grouping the results by sentence length. The word-level F-score development set results for the baseline system and the best OOV detector using self-training are presented in table 6. In both systems, the parser performs worse than the MaxEnt systems (including the first stage) on the long sentences. On short and medium sentences, the first MaxEnt stage performs comparable to the parser and second MaxEnt stage. After self-training, the parser performs worse on its own, but the parse features help in second stage MaxEnt systems. The best final MaxEnt system far outperforms the baselines on short and medium sentences, though it performs slightly worse on long sentences.

6. CONCLUSIONS

We find that domain adaptation helps improve the final performance on both rescoring and error/OOV detection tasks. Self-training using a combination of corpora to improve vocabulary coverage and match the structure of the test data provides the biggest gains for rescoring and OOV detection. The self-training configurations help in particular with final-stage OOV detection in all but the longest sentences. Future work will explore new modeling approaches, such as using multiple error arcs to jointly detect and distinguish between different types of ASR errors, and the use of global features in the log-linear model used to augment the PCFG.

7. REFERENCES

[1] C. Chelba, "A structured language model," *Computer Speech and Language*, vol. 14, pp. 283–332, 1997.

[2] Ciprian Chelba and Peng Xu, "Richer syntactic dependencies for structured language modeling," in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*, 2001.

[3] W. Wang, A. Stolcke, and M. Harper, "The use of a linguistically motivated language model in conversational speech recognition," in *Proc. ICASSP*, 2004, vol. I, pp. 261–264.

[4] B. Roark, M. Saraclar, and M. Collins, "Corrective language modeling for large vocabulary ASR with the perceptron algorithm," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 2004, vol. 1, pp. I–749–52 vol.1.

[5] Jeremy G. Kahn and Mari Ostendorf, "Joint reranking of parsing and word recognition with automatic segmentation," *Computer Speech and Language*, vol. 26, no. 1, pp. 1–19, Jan. 2012.

[6] A. Rastrow, M. Dredze, and S. Khudanpur, "Efficient structured language modeling for speech recognition," in *Proceedings of Interspeech*, 2012.

[7] Michael Collins, "Discriminative syntactic language modeling for speech recognition," in *Proceedings of ACL*, 2005, pp. 507–514.

[8] M. Harper, B. Dorr, B. Roark, J. Hale, I. Shafran, Y. Liu, M. Lease, M. Snover, L. Young, R. Stewart, and A. Krasnyanskaya, "Parsing speech and structural event detection," Tech. Rep., JHU Summer Workshop, 2005.

[9] Dustin Hillard, *Automatic Sentence Structure Annotation for Spoken Language Processing*, Ph.D. thesis, Seattle, WA, USA, 2008.

[10] A. Marin, T. Kwiatkowski, M. Ostendorf, and L. Zettlemoyer, "Using syntactic and confusion network structure for out-of-vocabulary word detection," in *Proceedings of the Spoken Language Technologies Workshop*, 2012.

[11] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," in *Proceedings of HLT-NAACL*, 2006, pp. 152–159.

[12] Wen Wang, "Combining discriminative re-ranking and co-training for parsing Mandarin speech transcripts," in *Proceedings of ICASSP*, 2009, pp. 4705–4708.

[13] Zhongqiang Huang and Mary Harper, "Self-training PCFG grammars with latent annotations across languages," in *Proceedings of EMNLP*, 2009.

[14] Lidia Mangu, Eric Brill, and Andreas Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech & Language*, vol. 14, no. 4, pp. 373–400, 2000.

[15] C. Parada, M. Dredze, D. Filimonov, and F. Jelinek, "Contextual information improves OOV detection in speech," in *Proc. HLT-NAACL*, 2010, pp. 216–224.

[16] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proc. ACL*, 1992, vol. I, pp. 517–520.

[17] Christopher Cieri David, David Miller, and Kevin Walker, "The Fisher corpus: a resource for the next generations of speech-to-text," in *in Proceedings 4th International Conference on Language Resources and Evaluation*, 2004, pp. 69–71.

[18] Michael Collins, "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms," in *Proc. ACL/EMNLP*, 2002, pp. 1–8.

[19] Michael Collins, "Head-driven statistical models for natural language parsing.," *Computational Linguistics*, vol. 29, no. 4, pp. 589–637, 2003.

[20] Jing Zheng, A. Mandal, Xin Lei, M. Frandsen, N.F. Ayan, D. Vergyri, Wen Wang, M. Akbacak, and K. Precoda, "Implementing SRI's Pashto speech-to-speech translation system on a smart phone," in *Proc. SLT*, 2010, pp. 133 –138.

[21] Andreas Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of ICSLP*, 2002, pp. 901–904.

[22] Y.-C. Tam, Y. Lei, J. Zheng, and W. Wang, "ASR error detection using recurrent neural network language model and complementary ASR," in *Proceedings of ICASSP*, 2014.

[23] Andrew. K. McCallum, "MALLET: A machine learning for language toolkit," http://mallet.cs.umass.edu, 2002.

[24] "SCLITE Toolkit," ftp://jaguar.ncsl.nist.gov/pub/sctk-2.4.0-20091110-0958.tar.bz2.