Sequence-based Class Tagging for Robust Transcription in ASR

Lucy Vasserman, Vlad Schogol, Keith Hall

Google, Inc.

{lucyvasserman,vlads,kbhall}@google.com

Abstract

We present a method of modeling non-lexical vocabulary items such as numbers, times, dates, monetary amounts and address components that avoids the data sparsity and out-of-vocabulary problems of written-domain language models. Like previous approaches, we use a class-based language model and efficient finite-state class grammars during run-time decoding. We mitigate the problem of context-independent replacement of class items by employing a contextual sequence labeling model to identify which class instances should be replaced, leaving others to appear in their original form. Applied to the task of general voice-search audio transcription, our method achieves 10% relative error reduction (on the numeric error rate metric) compared to the previous system (based on a verbalizer transducer). On a numeric entity recognition task, our method achieves a 23% relative error reduction on the same metric. In both cases, word error rate remains the same or is reduced.

Index Terms: speech recognition, language modeling

1. Introduction

Recent improvements in Automatic Speech Recognition and Natural Language Understanding have led to an increased interest in developing voice-enabled personal assistants. To meet the needs of the user, these assistants should have excellent support for numbers, times, dates, monetary amounts, address components and other non-lexical vocabulary items.

The task of producing correct and well-formatted written text for non-lexical vocabulary items is a difficult one. The main challenge is mapping between the verbal domain and the written domain [1, 2]. First, there is ambiguity with respect to segmentation. For example, "forty two dollars and thirty cents" could be transcribed as "$42.30" or "40 $2.30" or "$42 and 30¢", etc. Second, there is ambiguity with respect to semantics. The words "eleven thirty" could be a time "11:30", a house number "1130" or they can be kept in the verbal domain as "eleven thirty". The correct decision depends on context.

Data sparsity and out-of-vocabulary (OOV) issues are also a major problem for numeric entities. There may not be enough training data to adequately estimate the n-gram statistics for every numeric entity in the vocabulary. For some entities, such as phone numbers, we cannot reasonably expect to have hundreds of millions of lexicon entries to represent them all.

In this work, we resolve the above issues by using a statistical sequence tagger to identify and replace class instances in raw text with their label, and then training a class-based language model on the output of this tagger. We use class instance statistics from the tagging step to train class acceptor grammars in the verbal domain, which are substituted into the language model at decode time as in [2].

There are a number of advantages of our approach:

- The mapping from verbal domain to written domain is context dependent. When a user speaks "set alarm for eleven thirty", the class-based model produces "set alarm for <time> eleven thirty </time>" as the most likely path. There is no ambiguity about the correct written form.
- The arc weights of the class grammars are adapted to the semantic classes they represent. We obtain maximum-likelihood probability estimates for the different verbal forms of the same written entity.
- The OOV problem is greatly reduced as we are able to generate written forms for any numeric entity up to some pre-determined maximum length. The data sparsity problem is also greatly reduced as a result of modeling in the verbal domain.

2. Related Work

There have been a number of approaches to deal with the above mentioned issues in representing classes in language models.

Verbal-domain language models attempt to first transform all training data to the verbal domain leaving the correct formatting of classes to text-normalization of the verbal transcription [3].

Language model verbalization (introduced in [1] and [2]) partially addresses the problem by introducing a finite-state verbalizer transducer into the decoding network. This allowed the authors of [2] to preserve the richness of a written language model while requiring only a verbal-domain lexicon for pronunciations. Two problems remain:

Problem 1: The verbalizer transducer is not context-dependent, thus introducing too many unnecessary verbalization paths into the network.

Problem 2: All verbalizations are given equal probability, which is suboptimal since adding additional rare variants could easily hurt performance in the common case.

Class-based language models [4] have been used previously to model non-lexical vocabulary items with regular languages. While our work is based on this approach, we attempt to avoid the problem of the context-independent replacement of class items. In this work, we use a contextual sequence labeling model to identify which instances of a class should be replaced, leaving other instances to appear in their original form. This is very similar to the motivation behind the Word-Phrase-Entity (WPE) LMs presented in [5]. In the WPE models, they show that context dependent models for class membership improve performance. Unlike the iterative approach used in their system, we directly transform the raw training material to the class-based domain in a single pass. Another previous solution, presented in [6] used a semantic-role labeling parser to identify date and time expressions which were then used in a

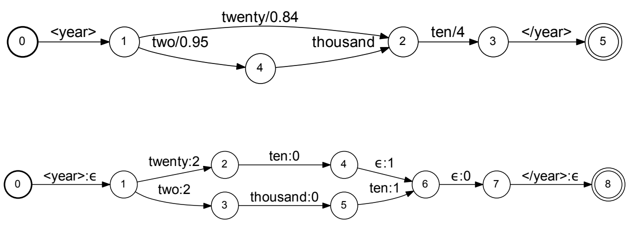

Figure 1: Class acceptor grammar (top) and denormalization grammar (bottom) example. These FSTs correspond to a small excerpt of the full grammar for *YEAR*.

vocabulary-based deterministic tagging of the components of these expressions.

3. Methodology

In this paper, we propose a context-aware method for building class-based language models. To build such a system, we proceed in four steps:

1. Construct written-to-verbal *verbalization grammars* for each class we wish to model.
2. Train a conditional random field (CRF) tagger to identify spans of tokens in language model training material that correspond to each of the classes [7].
3. Rewrite the language model training material into the class-based domain by replacing class text spans with a class placeholder symbol as dictated by the tagger. Save the count statistics for pairs of (class, replaced-text). Use these statistics to train the class acceptor grammar weights.
4. Train the language model from the classed text produced in the previous step and incorporate the class acceptors with either static expansion or dynamic expansion. The class denormalization grammars can then be applied as a post-processing step to convert recognition results into written form.

3.1. Constructing the *verbalization grammar*

We use OpenGRM[1] to write the grammars. Such grammars can be efficiently compiled into finite-state transducers [8, 9]. Suppose R is one such rewrite grammar. Then we also produce a pair of derived grammars:

- a verbal form *class acceptor grammar*:

  $(\epsilon : \text{<r>}) \quad \text{Project}_{OUTPUT}(R) \quad (\epsilon : \text{</r>})$

  This grammar must be word-based and relabeled to match the recognition engine's word list because it is used as part of the first-pass language model.

- a *class denormalization grammar*:

  $(\text{<r>} : \epsilon) \quad \text{Invert}(R) \quad (\text{</r>} : \epsilon)$

  This grammar must be character-based so that it can produce arbitrary output strings.

In Figure 1, we show the derived grammars for a simple verbalization grammar that maps the written symbol "2010" to two possible verbal forms.

[1]`http://www.opengrm.org`.

3.2. Training the class tagger

We obtained a data set of 24,246 manually labeled sentences (175K tokens), with a mean sentence length of 7.2 tokens. Each token is labeled with one of 17 possible tags. About 80% of the tokens are labeled with a 'none' tag, meaning that the token is not part of one of the pre-determined non-lexical classes. The remaining tokens are tagged with an actual class tag. The counts for the top 11 classes are given in Table 1. The data evaluators were instructed to use the 'none' tag when labeling a numeric token that appears as part of a named entity (e.g. "xbox 360").

We use the following set of baseline features:

isolated features:	w_i, d_i, c_i, n_i
neighboring words:	$w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}$
neighboring clusters:	$c_{i-2}, c_{i-1}, c_{i+1}, c_{i+2}, c_{i+3}$
word-digit pairs:	$(w_i, d_{i-1}), (w_i, d_{i+1})$
digit-digit pairs:	$(d_i, d_{i-1}), (d_i, d_{i+1})$
range features:	$R_i(B, E)$
divisibility:	$D_i(10)$
divisibility pairs:	$(w_{i-1}, D_i(10))$
transition:	(w_i, t_{i-1})

Here, i denotes a position in the input sentence; w_i and t_i are the word and output label (respectively) at that position, d_i are indicators of word types, n_i are indicators of number types, and c_i is the cluster assigned to the word. We also designed some domain-specific features: $D_i(x)$ indicates whether the word at position i is a representation of an integer number divisible by x and $R_i(B, E)$ indicates whether it is an integer in the range $[B, E]$. We also include class bias features, which capture the class distribution found in the training set.

Tag	Train Set	Held-out Set
Address Number	4568	495
Day	592	76
Money	6713	780
Month	617	78
Number	9032	980
Percent	1014	130
Phone Number	1187	125
Postal Code	1210	138
Street Number	991	133
Time	2171	234
Year	2287	262

Table 1: Distribution of the 11 most common output tags in the training set consisting of 21,817 sentences, and in the held-out set consisting of 2,429 sentences.

3.3. Phrase cluster features

In addition to the small, supervised dataset described above, we wish to take advantage of the large amounts of unlabeled data available to us. The inclusion of unsupervised cluster features is a common tactic to improve performance on rare examples in classification, named-entity recognition [10], and slot-filling [11]. Word clusters are already included in our baseline feature set. Lin and Wu [12] showed good results using phrase clusters, and we adapt their solution for our use case. We expect phrase clusters to solve several problems that word-based features do not handle well:

- Classes may be variable token length, especially given poorly formatted input data: e.g. "212 555 1234" and "212-555-1234" are both phone numbers.

- Phrase clusters provide some of the benefits of bigram or trigram features without the data sparsity issues.

Working off a large dataset of search queries, we cluster all phrases based on the context words around them. For each phrase, defined as any sequence of 1 to 4 words, we gather all of the context words that appear adjacent to the phrase. We calculate the pointwise mutual information (PMI) between the phrase and the word as follows, where phr is the phrase and w is a context word:

$$PMI(phr, w) = \log \left(\frac{P(phr, w)}{P(phr)P(w)} \right)$$

Each phrase is represented as a feature vector of PMI values for all context words. We then cluster these feature vectors using an SVD + Varimax soft clustering to generate 200 phrase clusters.

We add two new features to the CRF tagger: one for phrase clusters for all phrases that include the token to be classified and one for all context phrases, i.e. phrases that surround the active token.

Table 2 shows examples of CRF class tagging with and without phrase cluster features. We see that phrase clusters improve generalization.

Sentence (focus token in **bold**).	Without PC	With PC
007 Daniel Craig	*ADDRESS*	*NONE*
what was the Judiciary Act of **1789**	*NUMBER*	*YEAR*
65 Malibu SS	*ADDRESS*	*YEAR*
house for rent under **5000**	*MONEY*	*MONEY*
house for rent under **50**	*NUMBER*	*MONEY*

Table 2: CRF tags with and without phrase clusters (PC).

3.4. Augmented loss

Another method to incorporate unsupervised data is an augmented loss training set, which the CRF training algorithm uses as a secondary metric to optimize [13]. We incorporate negative training examples, i.e. sentences with numeric tokens that should not be in any class, via an augmented loss objective. Tagging an entity that should not be in a class hurts the resulting language model in two ways: (1) it allows all class members in a context where only one makes sense and (2) it biases the class grammar towards the specific entity (see training section below). The augmented loss dataset combats these errors with unsupervised negative examples.

One very common type of negative example is verbal form numbers. Our class-based system always produces numbers in written form (e.g. 3), but in some cases verbal form is more appropriate (e.g. three). This is especially common for single digit numbers. We created an augmented loss dataset of sentences with single digit numbers that should be transcribed in the verbal form.

To generate this dataset, we made the assumption that the verbal form is preferred for single digit numbers in all contexts where the verbal form is valid (i.e. "two items on my todo list" but not "2 * 47"). We mined human-transcribed utterances for examples of the numbers 1 through 9 written out in verbal form and duplicated those sentences to use the written form as well. Both the verbal form and written form become negative examples. Including the written form makes the model robust to poorly formatted user input data.

3.5. Training the class acceptors

To calculate optimal arc weights for the class acceptor grammars, we use a training corpus of the replaced text identified by the tagger. Since most of our training material is in the written domain, and the training examples need to be in the verbal domain, we are forced to keep only those examples that can be converted from written to verbal form by the verbalization grammar for that class. In practice, this limitation is negligible since we see enough examples that match the written side of this grammar that we can obtain a broad verbal domain sample.

Given a verbal domain training corpus, and the FST representing the acceptor grammar (as in Figure 1 (top)), we can efficiently estimate maximum likelihood arc weights by applying the forward-backward algorithm.

One problem with using the verbalization grammar is that the grammar is unweighted, thus giving uniform probabilities to all verbalization options for a written entity. We can do better if we decode untranscribed audio data with our initial system, and then use the verbal domain class instances hypothesized by the recognizer (before denormalization) to supplement the class acceptor training corpus. Furthermore, these unsupervised transcripts can be filtered by recognition confidence to reduce false positives.

We apply these grammars to the general language model either via static composition or via dynamic composition [14, 2]. In the case that there are cycles in the grammars, dynamic composition is the most feasible solution as it is constrained by the speech input being decoded.

4. Experimental Results

We use two manually transcribed test sets to evaluate the performance of our approach in the context of numeric transcription. The first test set `VOICE-SEARCH` (85,409 words, including 1,927 numeric entities) is a sample from general voice-search traffic, and tracks any regressions that appear as a result of biasing too heavily toward the selected classes. The other test set `NUMERIC` (22,298 words, including 3,646 numeric entities) contains utterances we expect to benefit from class-based modeling of numeric entities.

4.1. Numeric error rate

In addition to the standard word-error-rate (WER) metric, we also employ the numeric error rate (NER) metric introduced in [1]. This metric helps narrow in on the portion of the errors we are interested in fixing, as well as giving a more accurate picture than WER which is insensitive to some types of formatting mistakes.

4.2. Baseline system

Our speech recognition system is based on a long short-term memory neural network acoustic model [15, 16] with a vocabulary of approximately 4 million words. The baseline language model is a Katz [17] smoothed 5-gram model pruned to 100M n-grams, trained using Bayesian interpolation to balance multiple sources [18]. The OOV rate of this recognizer is 0.26%. Our second-pass rescoring LM is a distributed model trained on the same data fully concatenated using Katz backoff and pruned to 15 billion n-grams [19]. The system's lexicon transducer incorporates the pronunciations derived from the verbalizer transducer presented in [2].

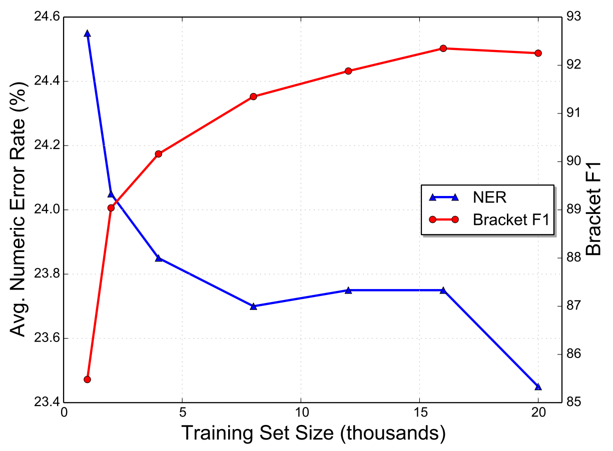

Figure 2: Training set size vs bracket F1 and average numeric error rate across both test sets for model C (Online + Phrase Clusters).

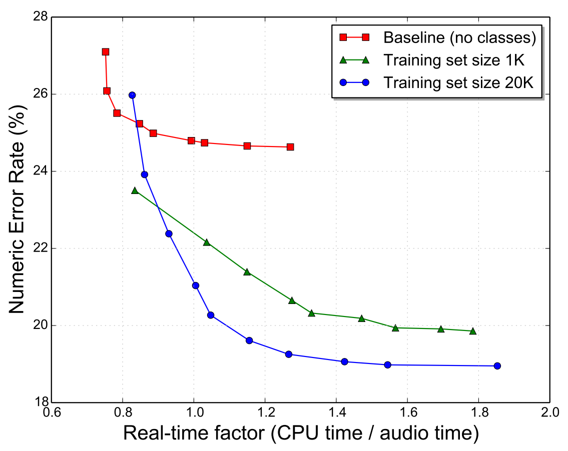

Figure 3: Numeric error rate on the NUMERIC test set vs. real-time factor for two tagger training set sizes: 1K and 20k (model C, Online + Phrase Clusters).

4.3. Results

Model	Brkt F1	NER Test set *NUM*	NER Test set *VS*	WER Test set *NUM*	WER Test set *VS*
Baseline	-	25.0	30.7	16.2	11.0
Batch (A)	92.5	19.7	28.9	14.7	10.9
Batch+Phr (B)	92.3	19.6	28.1	14.6	10.9
Onl (G)	92.5	19.5	29.0	14.5	10.9
Onl+Phr (C)	92.3	19.3	27.6	14.5	10.9
Onl+Neg (E)	90.6	19.3	27.8	14.5	10.9
Onl+Phr+Neg (F)	91.0	19.1	28.3	14.5	10.9

Table 3: NER and WER with varying tagger models (Phr - the Phrase Cluster models, Neg - the Negative examples introduced via augmented loss, Onl - Online training). Each tagger model was trained with 20,000 sentences. *NUM* refers to the NUMERIC entities test set and *VS* refers to the VOICE-SEARCH test set.

We evaluated the CRF tagger model and resulting ASR with different configurations of features and different training methods (batch performed using an LBFGS optimizer and online using a variant of the MIRA algorithm), and compared these with a non-class-based baseline system. All language models had the same vocabulary size and were pruned to the same number of n-grams.

In Figure 2 we see the effect of different CRF training set sizes on numeric error rate and Bracket F1. Bracket F1 performance consistently improves from 85-86% at 1K training examples, to 92% at 20K training examples. Most of the gain in NER is achieved by about 4K to 5K examples of training data, but there are improvements even out to 20K.

In Figure 3 we see the effect of CRF training set size on the real-time factor vs NER. Larger CRF training sets generally perform better, and the class-based system outperforms the baseline at most real-time factor values. In addition, the class-based system continues to improve in quality at higher real-time factors (up to 1.5), which provides more flexibility to select an operating point that meets the needs of the run-time environment.

Table 3 also shows that all class-based systems outperform the baseline in numeric error rate (NER) and in WER on the NUMERIC test set, with 23% relative improvement on NER for the best models (C, E, and F). All models produce the same WER on the VOICE-SEARCH test set, which indicates that NER reductions are not coming at the expense of non-numeric utterances.

As expected, phrase clusters (models B and C) improve NER performance over the basic feature set (models A and G) for both batch and online training. The augmented loss dataset (model E) also improves performance over the basic feature set (G, augmented loss is only available for online training). However, combining phrase clusters and augmented loss improves NER on the NUMERIC test set, but hurts the VOICE-SEARCH test set. Results like this usually imply an excessive bias towards the classes in the language model, which causes over-prediction of numeric entities. More investigation is needed to confirm and address this problem with model F.

5. Conclusions

We presented a method for training context-dependent class-based language models to improve recognition of non-lexical vocabulary items. Using a contextual sequence labeling model to identify class instances in training data mitigates the common problem of context-independent class labeling. We showed various features and methods for training the sequence labeling model and compared performance of different models. We conclude that our context-dependent class-based language models reduce numeric error rate by 23% over the baseline system.

6. Acknowledgements

Thanks to Cyril Allauzen, Pedro Moreno, Johan Schalkwyk, Francoise Beaufays, Hasim Sak, Richard Sproat and Mark Epstein.

7. References

[1] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer Speech & Language*, vol. 15, no. 3, pp. 287–333, 2001.

[2] H. Sak, F. Beaufays, K. Nakajima, and C. Allauzen, "Language model verbalization for automatic speech recognition." in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8262–8266.

[3] C. Chelba, J. Schalkwyk, T. Brants, V. Ha, B. Harb, W. Neveitt, C. Parada, and P. Xu, "Query language modeling for voice search," in *Proceedings of the 2010 IEEE Workshop on Spoken Language Technology*, 2010, pp. 127–132.

[4] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, pp. 467–479, 1992.

[5] M. Levit, S. Parthasarathy, S. Chang, A. Stolcke, and B. Dumoulin, "Word-phrase-entity language models: Getting more mileage out of n-grams," in *Proc. Interspeech*. ISCA - International Speech Communication Association, September 2014. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=219833

[6] X. Luo and M. Franz, "Semantic tokenization of verbalized numbers in language modeling," in *Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000*, 2000, pp. 158–161.

[7] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[8] M. Mohri and R. Sproat, "An efficient compiler for weighted rewrite rules," in *IN 34TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 1996, pp. 231–238.

[9] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The opengrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*, ser. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 61–66.

[10] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, ser. CoNLL '09, 2009, pp. 147–155.

[11] R. Sarikaya, A. C, A. Deoras, and M. Jeong, "Shrinkage based features for slot tagging with conditional random fields." ISCA - International Speech Communication Association, September 2014. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=219821

[12] D. Lin and X. Wu, "Phrase clustering for discriminative learning," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ser. ACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 1030–1038. [Online]. Available: http://dl.acm.org/citation.cfm?id=1690219.1690290

[13] K. Hall, R. McDonald, and S. Petrov, "Training structured prediction models with extrinsic loss functions," in *Domain Adaptation Workshop at NIPS*, October 2011.

[14] M. Mohri, "Weighted grammar tools: the grm library," in *Robustness in Language and Speech Technology*. Springer, 2001, pp. 165–186.

[15] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, 2014, pp. 338–342.

[16] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of Long Short-Term Memory recurrent neural networks," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, 2014, pp. 1209–1213.

[17] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," in *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1987, pp. 400–401.

[18] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," in *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association*, 2011, pp. 1429–1432.

[19] P. Jyothi, L. Johnson, C. Chelba, and B. Strope, "Distributed discriminative language models for google voice search," in *Proceedings of ICASSP 2012*, 2012, pp. 5017–5021.