

ERROR-DRIVEN PRUNING OF LANGUAGE MODELS FOR VIRTUAL ASSISTANTS

Sashank Gondala^{1,*} Lyan Verwimp^{2,*} Ernest Pusateri²
 Manos Tsagkias² Christophe Van Gysel²

¹Georgia Institute of Technology, ²Apple
 sgondala@gatech.edu,
 {lverwimp, epusateri, etsagkias, cvangysel}@apple.com

ABSTRACT

Language models (LMs) for virtual assistants (VAs) are typically trained on large amounts of data, resulting in prohibitively large models which require excessive memory and/or cannot be used to serve user requests in real-time. Entropy pruning results in smaller models but with significant degradation of effectiveness in the tail of the user request distribution. We customize entropy pruning by allowing for a keep list of infrequent n-grams that require a more relaxed pruning threshold, and propose three methods to construct the keep list. Each method has its own advantages and disadvantages with respect to LM size, ASR accuracy and cost of constructing the keep list. Our best LM gives 8% average Word Error Rate (WER) reduction on a targeted test set, but is 3 times larger than the baseline. We also propose discriminative methods to reduce the size of the LM while retaining the majority of the WER gains achieved by the largest LM.

Index Terms— ASR, LM Pruning, discriminative, data selection, error prediction

1. INTRODUCTION

VAs are popular services [1] that help users accomplish multiple tasks through voice queries. The Automatic Speech Recognition (ASR) engine, the VA component responsible for converting spoken queries into text, faces a challenge due to the many task domains VAs supports. Task domains include performing actions on the device where the VA runs (e.g. placing a call on a cell phone) or querying information about real-world events such as the outcome of a sports competition.

Accurately recognizing queries that concern contemporary events is a difficult problem due to the dynamic nature of the world. Hence, within VA systems, the language models (LMs) of the ASR system are typically trained on synthetic queries that are generated from knowledge bases [2], in addition to transcribed user queries. For example, when the artist Kanye West announced his album *Donda: With Child* in July 2020, the artificial queries “*play Donda With Child*”

and “*what is Donda With Child*” are included within LM training data. Likewise, artificial queries corresponding to entities that may occur infrequently within real usage data (i.e., tail entities), but should still be recognized accurately, are also included within LM training data.

As users expect low-latency responses from online services [3], n-gram backoff LMs [4] are frequently used, along with entropy-based pruning [5] to reduce speech recognition runtime and memory consumption. Entropy-based pruning removes n-grams from an LM that have the smallest impact on training set perplexity. This can be problematic if explicitly-observed n-grams from synthetic queries are removed, making them indistinguishable from unobserved n-grams in the training data. However, entropy pruning can be modified to apply different pruning thresholds to a subset of LM n-grams, and hence, we use a more relaxed pruning threshold for specific (tail) n-grams that enhance recognition.

In this paper we investigate how to determine the minimal set of n-grams that require a more relaxed pruning threshold from textual features. In particular, we are interested in improving recognition on a set of synthetic queries Q that we know we want our VA to recognize, but are absent or underrepresented in our live usage data. This problem is challenging due to the large number of synthetic queries, the lack of generalization in n-gram LMs, the dynamicity of live usage data, and VA runtime constraints.¹

Our research questions are: **(RQ1)** Are there text-based signals that are a good predictor for speech recognition difficulty? **(RQ2)** Can we determine an optimal subset of synthetic query n-grams that need a more relaxed pruning threshold without degrading speech recognition effectiveness? We contribute: (1) a formal framework for applying different entropy-based pruning thresholds on subsets of n-grams, (2) three methods for determining a subset of n-grams for which pruning needs to be relaxed to improve their recognition, (3) insight into which signals are useful to predict speech recognition difficulty directly from text.

¹We also tried a variety of different smoothing methods [6] to try to achieve the same goal, including Kneser-Ney smoothing [7, 8]; no significant improvements on our tail- and entity-rich test sets were observed.

¹Work done while the first author was an intern at Apple.
 *Equal contribution.

2. CUSTOMIZED PRUNING FOR QUERIES

We describe three methods for pruning backoff LMs, all of which build upon entropy-based pruning [5], with the goal of improving recognition on a set of synthetic queries Q . In entropy-based pruning (*EP*), n-grams that are estimated to increase training data perplexity by less than a threshold, θ , are greedily removed from a backoff n-gram LM. We extend this approach by introducing a *keep list* of n-grams. The keep list defines a set of n-grams, extracted from the queries in Q , for which a more relaxed threshold, θ_{keep} , will be applied. Notice that even though we apply this extension to entropy pruning and a set of synthetic queries, it can in principle be applied to any type of n-gram LM pruning (e.g. [9]) and any dataset for which you want to improve recognition. The three methods described below differ only in the way the keep list is generated. Each method provides a different space of trade-offs between keep list generation cost, LM size, and ASR accuracy.

2.1. Query-driven Entropy Pruning

In query-driven entropy pruning (*QEP*), we generate the keep list by extracting n-grams from every query $q \in Q$. Using the full set of synthetic queries does not require extra processing and gives the best ASR accuracy on tail- and entity-rich test sets, but can easily blow up the size of the LM. While effective, QEP is infeasible to apply when Q is large and contains many unique n-grams.

2.2. Error-driven Entropy Pruning

To reduce the number of n-grams in the keep list, we apply a discriminative approach (e.g., [10, 11, 12]), error-driven entropy pruning (*EEP*). Here we only want to exclude those n-grams from pruning for which our baseline ASR, using an entropy-pruned LM, fails. However, large-scale manual transcription of audio is expensive, and we want to optimize the ASR accuracy of data that is possibly heavily underrepresented in real user data. Therefore, for every phrase in Q , we generate audio with Text-To-Speech (TTS) and recognize it with our baseline ASR (see [13] for a similar approach). The decoding errors are then used as a source to extract n-grams for the keep list. This TTS-ASR loop is costly both in terms of time and computational resources and thus prohibitive for large datasets.

2.3. Approximate Error-driven Entropy Pruning

The TTS-ASR loop can be avoided if we can predict, based on textual features alone, whether a query q will be recognized incorrectly by the baseline ASR. Approximate error-driven entropy pruning (*AEEP*) – works as follows: we train a binary classifier (see §4.1) on a training set T_Q , which is sampled from the same distribution as Q , that predicts the outcome of the TTS-ASR loop, and more specifically, whether an n-gram will be recognized incorrectly. After training, we use that classifier to select a subset of Q as keep list.

For every unique query $q_i \in T_Q$ in our dataset, we apply the TTS-ASR loop to obtain \tilde{q}_i , the top-hypothesis after recognizing the synthesized audio of phrase q_i . Subsequently, we obtain an alignment (edit distance) between q_i and \tilde{q}_i and extract n-grams from q_i . The input to our model is an n-gram extracted from q_i and it is assigned a positive label if the target token in the n-gram (e.g. the target token in a 4-gram is the 4th token) differs from its aligned token in \tilde{q}_i . Every instance is represented as a real-valued feature vector. We consider five categories of features:

Word-level features. We compute n-gram statistics, including n-gram count and frequency in T_Q , and whether the n-gram context words and/or target word are out-of-vocabulary.

Language model features. We compute features derived from the baseline entropy-pruned LM, such as log probability, perplexity and entropy, of the full n-gram and of the target word given the context.

Phoneme-level features. Our VA lexicon is generated based on a list of pre-defined word-pronunciation mappings for frequent words and exceptions, and a grapheme-to-phoneme tool (G2P, see §3) that automatically generates pronunciations for words that are not in that list. For a given n-gram, we generate phonemes for each word individually with our G2P (even when the word occurs in the pre-defined list) and join them together to create the phoneme string and corresponding phoneme n-grams. From these, we extract the following features: the number of phonemes, whether the phoneme string contains infrequent phonemes, whether the word-pronunciation mapping occurs in the pre-defined list, and the edit distance between the phoneme string generated using G2P and the pre-defined pronunciation (if present).

Template features. Our dataset of synthetic queries consists of a set of templates (e.g. “Who is the `position-tag` of `team-tag`?”) where the tags are replaced with entities. We tag every word of the utterance to be either a template token or entity and create two features: whether the target word is an entity and whether any word in the n-gram is an entity.

Phonetic confusion features. For each n-gram, we take the target word, extract its phoneme string and compute the set of phoneme strings with an edit distance of 1. For each phoneme string in that set, we check if it maps to any of the words in the vocabulary and if it does, we compute the total log probability of the n-gram where the target word is replaced with the new mapped word. We store the highest of these log probabilities as a feature.

3. EXPERIMENTAL SETUP

Training LMs. We train our LMs on manually and automatically transcribed anonymized VA requests, and on the dataset of synthetic queries Q , which consists of a large set of domains, e.g. sports, music and home automation. The queries consist of a set of templates with slots and a list of entities that can fill those slots (§2.3). Both templates and entities are

typically derived from real user data and thus have prior probabilities that are used to sample the synthetic requests. In our experiments, query set Q consists of 2.8B queries (112M unique) spanning 27 domains. The total training data contains more than 10B utterances.²

Pruning methods. Query set Q is also used for our three proposed pruning methods to extract the following: all possible n-grams (QEP), a subset of n-grams based on the errors collected through the TTS-ASR loop (EEP), and a subset of n-grams based on the classifier trained on the output of the TTS-ASR loop (AEEP). In addition, we compare against regular entropy pruning (EP), where Q is not taken into account.

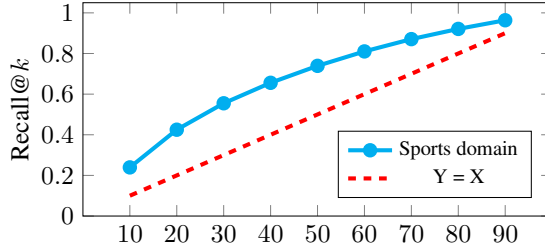
Evaluating the AEEP classifier. To assess the utility of AEEP, we train on all domains except one and validate and test on a held-out domain. We consider only one held out domain, that of sports, because it is the only domain for which we have domain-specific test sets using both manually transcribed user requests and synthesized requests.

ASR accuracy evaluation. After applying the pruning methods outlined above, we evaluate WER on three main test sets: (1) General VA contains VA requests sampled from the actual distribution, thus containing mostly frequent/head utterances. (2) Sports is a sample of less frequent VA requests for the sports domain. (3) TTS-All consists of synthesized requests, it is sampled from the same templates as Q but is a different instance. Thanks to the prior probabilities, we can make a distinction between subsets of TTS-All that contain utterances from the head (top 10%), torso (10-50%) and tail (50-100%) of the distribution. TTS-Sports is a subset of TTS-All containing all utterances related to sports. Our goal is to improve WER (with a minimum increase in LM size) on the Sports and TTS-All test sets, while not degrading recognition effectiveness on the General VA test set.

System description. Our ASR system consists of an acoustic model that is a deep convolutional neural network [14], a 4-gram LM with Good-Turing smoothing in the first pass (see [15, 16] for details), and the same LM interpolated with a Feed-Forward Neural Network (FFNN) LM [17] in the second pass. To build a scalable TTS-ASR loop, we use our previous generation speech synthesizer, a unit selection system described in [18]. We use scikit-learn [19] to train binary classifiers for AEEP. For combining entropy pruning with a keep list we modified SRILM [20]. Our G2P is an LSTM encoder-decoder architecture with attention (similar to [21]). For all experiments we set $\theta = 6e - 9$ and $\theta_{keep} = 0$, which means that all n-grams in the keep list are excluded from pruning. We experimented with different values for θ_{keep} as an alternative approach to reduce the size of the LM, but did not observe good improvements. All experiments reported here are for American English.³

²We are unable to provide the exact number due to confidentiality. The manually transcribed dataset is a small random sample.

³For QEP compared to EP, we observe similar WER reductions for several other languages/regional variants, e.g., German and Mandarin Chinese.



Percentage of n-grams assigned to positive class/keep list (k)

Fig. 1. Recall@ k for the sports domain.

4. RESULTS

4.1. AEEP Classifier

In this section, we report the results for training a binary classifier on the output of the TTS-ASR loop to predict whether an n-gram will be recognized incorrectly. Comparing 4 different model types, Random Forests (RF), AdaBoost, linear support-vector machines and FFNNs, we found that RFs achieved the best results on the validation data. Hence, all experiments reported below use RFs.

We parameterize our experiments based on the number of n-grams to consider: We take the top- k % of n-grams as ranked by the classifier confidence score and assign them to the positive class that will form the keep list. In Fig. 1, we show recall@ k for the sports domain: the relative number of n-grams correctly classified as positive if k % of the ranked data is assigned to the positive class/keep list. We observe good performance, with a recall of 0.65 at 40%. For the downstream task (§4.2), we select this 40% of n-grams as the keep list for AEEP (Sports). For our ablation study, we rank the features according to the importance assigned to them by the RF. We find that by using just the top-3 features to train a new RF, i.e. (1) log probability of target word given context, (2) log probability of the full n-gram and (3) largest log probability of the n-gram at an edit distance of 1, we still obtain a recall of 0.63 at 40%. We will use the data selected by this model as well in §4.2 (AEEP-Top-3).

These results provide an answer to **RQ1**: Yes, there are text-based signals that can predict ASR difficulty. LM and phonetic confusion features are the most important ones.

4.2. ASR Accuracy

We now evaluate our 3 customized pruning approaches in our ASR system. We set our success criterion to minimize the size of a LM while improving WER compared to that of a system using an entropy-pruned LM. In the last row of Table 1, we show the size of each LM in number of n-grams. QEP roughly triples the size of the entropy-pruned LM, while EEP doubles it. The unpruned LM contains 313M n-grams and is too large to use in ASR decoding. In our ASR system, using the larger LMs leads to increased, but still acceptable, memory usage, and we observe negligible impact on the decoding speed. Since all our proposed approaches increase

Table 1. WER results (best result per test set in bold) and size (last row) for LMs with various pruning strategies. EP base is the baseline, EP~ LMs are entropy-pruned LMs of the same size as the proposed models. AEEP uses all features, and AEEP-Top-3 uses the top-3 features after feature selection. ‘All domains’ and ‘Sports domain’ in the top row refer to the query set used to generate the n-gram set.

Test set	# utts.	EP base	All domains				Sports domain					
			EP~	QEP	EP~	EEP	EP~	QEP	EP~	EEP	AEEP	AEEP-Top-3
General VA	49k	4.13	4.08	4.13	4.09	4.13	4.13	4.13	4.14	4.13	4.13	4.13
Sports	4k	4.72	4.47	4.32	4.44	4.41	4.70	4.25	4.69	4.38	4.27	4.31
TTS-All	100k	15.07	14.42	13.89	14.64	14.00	14.95	14.22	15.00	14.27	14.26	14.27
Head	32k	8.54	7.85	7.38	8.05	7.49	8.40	7.73	8.46	7.76	7.78	7.78
Torso	34k	15.38	14.64	13.79	14.91	13.95	15.23	14.15	15.29	14.24	14.22	14.24
Tail	34k	20.80	20.25	19.99	20.47	20.06	20.71	20.28	20.76	20.31	20.29	20.29
TTS-Sports	28k	19.12	18.37	15.82	18.64	15.95	18.98	15.75	19.04	15.95	15.93	15.96
Number of n-grams		22M	71M	71M	45M	45M	27M	27M	25M	25M	25M	25M

the size of the LM, we created additional EP baselines of the same size for each pruning strategy for a fair comparison in terms of LM size – see the EP~ columns in Table 1.

The first row of Table 1 shows that none of the pruning approaches hurts the WER on our regular, head-heavy general VA test set. On the test sets representing the usage that we want to improve on (i.e., tail utterances with many named entities), we see consistent WER reductions for all settings.

QEP leads to the largest LMs and the best WER results on TTS-All, with 8% relative WER reduction compared to the baseline EP LM, while the EP LM of the same size (EP~ column to the left of QEP) has only a 4% reduction. On the sports test with user queries we also observe a WER reduction of 8%, compared to 5% for EP~. Looking at results qualitatively, we observe that the EP LMs do not contain many higher order n-grams that make a difference, e.g. *who is Phil Bengtson* (in TTS-All) is recognized correctly by the QEP LM because it contains all relevant higher-order n-grams while the EP LMs only contain the unigrams.

The EEP LM, that is significantly smaller, only leads to small WER degradations compared to the QEP LM. We observe 7% relative WER reduction w.r.t to the baseline on TTS-All, compared to only 3% for the corresponding EP~ LM.

Finally, approximating the TTS-ASR loop with AEEP gives us WER results that are close to the real decoding errors (EEP). The AEEP model using all features described in §2.3 gives a 17% relative WER reduction with respect to the baseline EP LM on the TTS-Sports test set, and 10% on the Sports test set extracted from real user data. AEEP-Top-3, the model using only the top-3 features (§4.1), is on par with the variant using all features. On the Sports test set with real user requests, AEEP does even better than EEP. We hypothesize that EEP is overfitting on the data that is used for the TTS-ASR loop, while AEEP counteracts this problem because it approximates the results of the TTS-ASR loop.

For both the TTS-based error-driven pruning (EEP) and its approximation (AEEP), we select about 40% of the full Sports dataset for the keep list. One could argue that any

addition of sports-related n-grams to the LM will improve the WER on sports test sets. Thus, as a sanity check we randomly select 40% of the sports n-grams and use it as the keep list. The resulting LM gives a WER of 16.27 on the TTS-Sports test set, which is still 0.3 absolute worse than AEEP. We can conclude that the 40% selected by EEP and AEEP is a more meaningful selection than a randomly selected 40%.

We can conclude that w.r.t. **RQ2** the optimal set of synthetic query n-grams that require a more relaxed pruning threshold can be the full set (§2.1) if there are no memory limits. However, we showed that it is possible to retain the majority of the WER gain with a much smaller LM by selecting n-grams with a model trained on textual features alone.

5. CONCLUSIONS

We explored three methods to customize LM pruning to improve ASR accuracy on infrequent and entity-rich utterances, by constructing a keep list of n-grams that require a more relaxed pruning threshold. QEP results in LMs that are three times larger than the baseline and give relative WER reductions of on average 8%, both on a targeted synthetic test set and a test set with user queries. EEP and the more efficient AEEP reduce the size of the keep list by selecting only decoding errors, resulting in LMs that are only twice as large as the baseline and still have good WER improvements of 17% on the domain-specific synthetic test set and 10% on the user query test set. We also showed that we can predict ASR difficulty based on textual signals.

In our future work, we would like to explore more text-only approaches to customize pruning, e.g. by modifying the pruning criterion itself or selecting data based on LM features and improved (FST-based) phonetic confusion features.

Acknowledgements. We thank Youssef Oualil, Amr Mousa, Russ Webb, Barry Theobald for their comments and feedback.

6. REFERENCES

- [1] Juniper Research, “Digital Voice Assistants in Use to Triple to 8 Billion by 2023, Driven by Smart Home Devices,” Press Release, Feb. 2019.
- [2] Ankur Gandhe, Ariya Rastrow, and Bjorn Hoffmeister, “Scalable language model adaptation for spoken dialogue systems,” in *IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 907–912.
- [3] Eric Schurman and Jake Brutlag, “Performance Related Changes and their User Impact,” Presentation at Velocity – Web Performance and Operations Conference, 2009.
- [4] Slava Katz, “Estimation of probabilities from sparse data for the language model component of a speech recognizer,” *Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.
- [5] Andreas Stolcke, “Entropy-based Pruning of Backoff Language Models,” *ArXiv*, vol. cs.CL/0006025, 1998.
- [6] Jianfeng Gao and Min Zhang, “Improving Language Model Size Reduction using Better Pruning Criteria,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002, pp. 176–182.
- [7] Stanley F. Chen and Joshua Goodman, “An empirical study of smoothing techniques for language modeling,” *Computer Speech and Language*, vol. 13, pp. 359–394, 1999.
- [8] Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu, “Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing,” in *Proceedings Interspeech*, 2010, pp. 2242–2245.
- [9] Kristie Seymore and Ronald Rosenfeld, “Scalable backoff language models,” in *International Conference on Spoken Language Processing (ICSLP)*, 1996, pp. 232–235.
- [10] Brian Roark, Murat Saraclar, and Michael Collins, “Corrective language modeling for large vocabulary ASR with the perceptron algorithm,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2004, pp. 749–752.
- [11] Takanobu Oba, Takaaki Hori, Atsushi Nakamura, and Akinori Ito, “Round-Robin Duel Discriminative Language Models,” *Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1244–1255, 2012.
- [12] Yuuki Tachioka and Shinji Watanabe, “Discriminative method for recurrent neural network language models,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5386–5390.
- [13] Jinxi Guo, Tara N. Sainath, and Ron J. Weiss, “A Spelling Correction Model for End-to-end Speech Recognition,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5651–5655.
- [14] Zhen Huang, Tim Ng, Leo Liu, Henry Mason, Xiaodan Zhuang, and Daben Liu, “SNDCNN: Self-Normalizing Deep CNNs with Scaled Exponential Linear Units for Speech Recognition,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6854–6858.
- [15] Ernest Pusateri, Christophe Van Gysel, Rami Botros, Sameer Badaskar, Mirko Hannemann, Youssef Oualil, and Ilya Oparin, “Connecting and Comparing Language Model Interpolation Techniques,” in *Proceedings Interspeech*, 2019, pp. 3500–3504.
- [16] Christophe Van Gysel, Manos Tsagkias, Ernest Pusateri, and Ilya Oparin, “Predicting Entity Popularity to Improve Spoken Entity Recognition by Virtual Assistants,” in *SIGIR*, 2020, pp. 1613–1616.
- [17] Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai, “The Fixed-Size Ordinally-Forgetting Encoding Method for Neural Network Language Models,” in *IJCNLP*, 2015, pp. 495–500.
- [18] Tim Capes, Paul Coles, Alistair Conkie, Ladan Golipour, Abie Hadjitarkhani, Qiong Hu, Nancy Huddleston, Melvyn Hunt, Jiangchuan Li, Matthias Neeracher, Kishore Prahallad, Tuomo Raitio, Ramya Rasipuram, Greg Townsend, Becci Williamson, David Winarsky, Zhizheng Wu, and Hepeng Zhang, “Siri On-Device Deep Learning-Guided Unit Selection Text-to-Speech System,” in *Proceedings Interspeech*, 2017, pp. 4011–4015.
- [19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., “Scikit-learn: Machine learning in Python,” *Journal of machine learning research*, vol. 12, pp. 2825–2830, 2011.
- [20] Andreas Stolcke, “SRILM – An extensible language modeling toolkit,” in *International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 901–904.
- [21] Shubham Toshniwal and Karen Livescu, “Jointly learning to align and convert graphemes to phonemes with neural attention models,” in *IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 76–82.