IMPROVED RECOGNITION OF CONTACT NAMES IN VOICE COMMANDS

Petar Aleksic, Cyril Allauzen, David Elson, Aleksandar Kracun, Diego Melendo Casado, Pedro J. Moreno

Google Inc.

{apetar,allauzen,elson,yak,mompe,pedro}@google.com

ABSTRACT

The recognition of contact names in mobile-device voice commands is a challenging problem. Some of the difficulties include potentially infinite vocabularies, low probability of contact tokens in the language model (LM), increased false triggering of contact voice commands when none are spoken, and very large and noisy contact name lists. In this paper we suggest solutions for each of these difficulties.

We address low prior probability and out-of-vocabulary contact name problems by using class-based language models, and creating on-the-fly user dependent small language models containing only relevant names. These models are compiled dynamically based on analysis of the mobile device state. Since these solutions can increase biasing towards contact names during recognition, it is crucial to monitor false triggering. To properly balance this bias we introduce the concept of a contacts insertion reward. This reward is tuned using both positive and negative test sets.

We show significant recognition performance improvements on data sets in three languages, without negatively impacting the overall system performance. The improvements are obtained in both offline evaluations as well as on live traffic experiments.

Index Terms: speech recognition, voice commands, contact names, FSTs

1. INTRODUCTION

Voice input has become a critical component of modern mobile smart phones. Over time, its usage has expanded from search by voice to free-form semantic voice commands. Typical examples include phrases such as "call Mike at work" or "send SMS to Frank with the topic arriving late".

These types of voice commands are relatively hard to recognize for several reasons. First, the set of unique names in most languages is very large. Including a large number of names in the language model could result in significant increase of lexicon and language model sizes, negatively affecting the ability to process voice commands accurately and with low latency. Secondly, including names not relevant to a particular user can hurt the recognition performance by causing false triggering of voice commands.

In this paper we present solutions to address these problems based on the ideas of class based language models introduced in [1, 2]. In particular, we create on-the-fly user dependent small language models containing only names relevant to a particular user. These are compiled dynamically based on analysis of the mobile device state. By performing G (language model) level replacement, we solve a major issue of preserving the right phonetic context of the word preceding a class-specific model and the left phonectic context of the word following a class-specific model as described in more detail in section 3. We also address a very important issue of handling out-of-vocabulary (OOV) words in the class-specific language models. In the case of user contacts, it is extremely likely that a large number of contact names will not be part of our main vocabulary. We solve this problem by adding context-independent phones to our main vocabulary as described in more detail in section 3.3.

Since these solutions can increase biasing towards contact names during recognition, it is crucial to monitor false triggering of voice commands. We address this issue by introducing the concept of a contacts insertion reward. As for contacts we do not have enough prior information to build a proper model, we investigate three approaches for setting contacts insertion reward and chose the optimal approach in which the reward can be adjusted on a per-user basis depending on the number of user contacts. This reward is tuned using both positive and negative test sets (see details in section 2.1).

We organize the paper as follows. In section 2, we present how language modeling training is modified to handle name classes, and our finite-state transducer (FST) replacement approach. In section 3, we explain how decoder graph construction has to be modified to efficiently handle dynamic lists of names. In section 4, we describe our strategy to identify the contact names that are relevant to a particular user. Finally, in section 5, we present all our experimental results.

2. LANGUAGE MODELING

The benefit of a class-based n-gram language modeling approach [1] is that it provides a user-independent modeling of the probability of a contact name in a given n-gram context, while making it easy to dynamically update our LM with a user-dependent contact name model. Our approach is closely related to the ones used in [2, 3].

In such a method, we will replace in our training corpora all contact name phrases by a special token $CONTACTS and train an LM like we usually do (section 2.3). At recognition-time, we will build a user-specific contact model based on user metadata (section 2.1). The finite-state replacement mechanism allows us to on-demand substitute visited instances of the class token $CONTACTS with the user-specific contact model (section 2.2).

2.1. Class-specific models

In a class-based approach, for a given class c we need to build a c-specific model that estimates the probability of a phrase w given c:

$$\Pr(w \mid c). \tag{1}$$

For contacts, we do not have enough prior information to build a proper model. For a given user, we only have access to a set P_c of contacts phrases. For a phrase $w \in P_c$, we investigated three approaches for setting the conditional probability of w given c:

1. *No modeling:* $\Pr(w \mid c) = 1$.
2. *Uniform modeling:* $\Pr(w \mid c) = \frac{1}{|P_c|}$.
3. *Scaled uniform modeling:* $\Pr(w \mid c) = \frac{e^{-\alpha}}{|P_c|^{1-\beta}}$.

While in some cases these approaches may lead to an LM that is no longer properly normalized, we will show that the weights of the search graph can be redistributed so that Viterbi search is not affected.

Approach 1 results in assigning the full class probability in the main language model (h) to every class phrase: $\Pr(w \mid h) = \Pr(c \mid h)$. When expressing the LM probabilities as costs (defined in negative-log domain), this can be viewed as applying a class "insertion reward" of $-\log|P_c|$. The resulting model is not properly normalized and the potential large reward interacts very negatively with the Viterbi beam search performed during decoding, unless weight pushing is applied on the decoder graph as described in more detail below.

Approach 2 is the only approach that results in a properly normalized model. However, it does not match well the reality of the task. Users tend to call only a small subset of their contacts, and the larger their contacts list the smaller will be the probability assigned to the contacts they are likely to call. This is made worse by the fact that our users' contacts lists are often extremely large. As a result, the cost assigned by the LM to a contact calling phrase will be too high and such hypotheses will be pruned away during decoding.

Approach 3 results in a class "insertion reward" of $\alpha - \beta \log|P_c|$ compared to Approach 2. This allows us to specify a base reward that can be adjusted on a per-user basis depending on the number of user contacts. For instance, with $\alpha = 0$ and $\beta = 0.5$, for a user with $10,000$ contacts, each contact will be assigned probability $\frac{1}{100}$ (instead of $\frac{1}{10000}$ with Approach 2) whereas for a user with 100 contacts, each one will have probability $\frac{1}{10}$ (instead of $\frac{1}{100}$ with Approach 2).

This approach generalizes the previous ones: Approach 1 corresponds to set $\alpha = 0$ and $\beta = 1$ and Approach 2 to $\alpha = \beta = 0$. It allows us to find a balance between the two previous approaches as we will see in Section 5.2.

We build the negative log-domain finite-state automaton G_c representing our class specific model as follows. We build a minimal deterministic unweighted automaton representing the set of phrases P_c. We use the weight pushing algorithm [4] to redistribute the weights along the path while preserving full-path weights in such a way that at every non-initial state, the $\oplus_{\log}$-sum of the weights of all outgoing transitions is 0 (1 in the probability domain). This property is important to ensure a well behaved beam search. The $\oplus_{\log}$-sum[1] of the outgoing arcs at the initial state is then $-\log|P_c|$. Hence, we can apply Approach 3 by adding $\alpha - (\beta - 1)\log|P_c|$ to the weight of every outgoing transition out of the initial state.

2.2. Class-based Language Models and Replacement

In our class-based approach, we have a top-level language model $G_{\text{\$ROOT}}$ represented by an FST where arcs can be labeled by regular words or class tokens such as \$CONTACTS. For a given utterance, we need to obtain an FST representation of our class-based LM by replacing each transition labeled by a class token by the corresponding utterance-specific FST modeling that class. This can be done using the finite-state replacement operation. The benefit of this approach is that this operation can be performed on-demand for each utterance. This leads to memory saving since $G_{\text{\$ROOT}}$ can be shared among threads[2]. Moreover only the part of replacement that is visited during recognition will be computed leading to low CPU overhead and to further memory saving.

[1]The $\oplus_{\log}$ operation is defined as $a \oplus_{\log} b = -\log(e^{-a} + e^{-b})$.

[2]Our FST decoder is multithreaded.

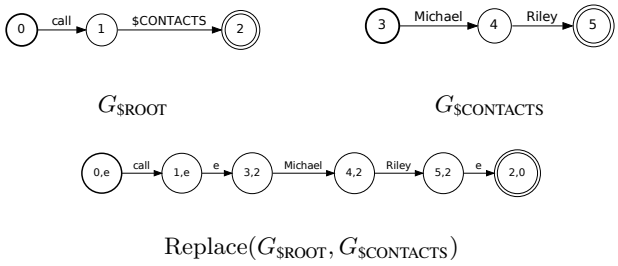

Fig. 1. Replacement algorithm.

The finite-state replacement operation is specified by a triple $R = (N, S, (T_\nu)_{\nu \in N})$ where N is a set of nonterminals, S is the root nonterminal and $(T_\nu)_{\nu \in N}$ is a family of FSTs. Assuming R has an acyclic dependency graph[3], the replacement of R is the finite-state transducer $T = \text{Replace}(R)$ defined as follows. Let $Q = \cup_{\nu \in N} Q_\nu$ where Q_ν is the set of states of T_ν. States in T are of the form $(q, s) \in Q \times Q^{*}$[4]. A state (q, ϵ) is initial (resp. final) in T iff q is initial (resp. final) in T_S. Transitions in T out a state (q, s) with $q \in Q_\nu$ are defined as follows. Given a transition in T_ν from q to r with input label a, output label b and weight w, we create

- if b is a terminal: a transition from (q, s) to (r, s) with input label a, output label b and weight w, or
- if b is a non terminal: transitions from (q, s) to (i_b, sr) with input label a, output label ϵ and weight w and from (f_b, sr) to (r, s) with ϵ input and output labels and 0 weight, where i_b and f_b are the initial and final states of T_b, and sr is ordered concatenation of states s and r.

An implementation of the on-demand replacement operation is available in the OpenFst library [5]. Figure 1 shows the FST obtained by applying the replacement operation to the triple $(\{\text{\$ROOT}, \text{\$CONTACTS}\}, \text{\$ROOT}, (G_\nu)_{\nu \in \{\text{\$ROOT}, \text{\$CONTACTS}\}})$.

2.3. Language Model Training

The top-level $G_{\text{\$ROOT}}$ language model was trained as follows. We first collected a training set of in-domain sentences that included contact related voice commands. For that purpose, we used unsupervised in-domain sentences. Specifically, we filtered anonymized voice search queries for those utterances that included contact names and that would, after being parsed by a semantic parser, trigger a contact voice command event such as contact calling, sending email or sms. We further anonymized the matched training phrases by replacing the contact name with the class label \$CONTACTS.

We used the Bayesian interpolation method of [6] to combine the voice command models with LMs trained on our other training sources. These include typed data sources (such as web search queries) and spoken data sources consisting of ASR results from anonymized utterances filtered by recognition confidence score [7].

3. RECOGNITION TRANSDUCER CONSTRUCTION

Given a language model G (an automaton over words), a phonetic lexicon L (a transducer mapping context-independent phones to

[3]The dependency graph of R is the directed graph (N, E) where there is an edge $(\mu, \nu) \in E$ iff there is a ν-labeled transition in T_μ.

[4]Kleene star operator.

words) and a context-dependency specification C (a transducer mapping context-dependent to context-independent phones), we need to construct a *recognition transducer* T that maps context-dependent phones to words.

When using non-class-based static language models, we follow the construction algorithm from [8]. We first statically build an optimized context-dependent lexicon CL by composing C with a determinized L:

$$CL = C \circ \mathrm{Det}(L). \tag{2}$$

Finally, we use the composition algorithm with reachability filters from [8, 9] to build:

$$T = CL \circ G. \tag{3}$$

This second composition can be performed *statically* or *on-demand*. When using the on-demand approach, the overhead of the recognition-time computation can be further reduced by using the pre-initialization algorithm from [10].

3.1. CLG-level Replacement

Given a class-based language model specification $(G_c)_{c \in N}$, we build an individual recognition transducer T_c for each class label/nonterminal c:

$$T_c = CL \circ G_c \tag{4}$$

and then define the recognition transducer:

$$T = \mathrm{Replace}((T_c)_{c \in N}). \tag{5}$$

A major issue with this approach is that the left and right phonetic contexts of a phrase in c are always the start and end phone (silence). The right phonetic context of the word preceding that phrase as well as the left phonetic context of the word following that phrase will also be incorrect.

3.2. G-level Replacement

An alternative approach is to leverage the fact that the composition algorithm with reachability filters [8] can be applied on-demand to a dynamic language model:

$$T = CL \circ \mathrm{Replace}((G_c)_{c \in N}). \tag{6}$$

One issue with this approach is that having to perform both composition and replacement on demand in this manner significantly increases the run-time overhead. However, the pre-initialization algorithm [10] provides a solution to this problem.

States in T are of the form $(p, (q, s))$ where p is a state in CL and (q, s) a state in $\mathrm{Replace}((G_c)_{c \in N})$. A state of the form (q, ϵ) corresponds to a state in the main language model and hence does not change between utterances. This means we can apply the pre-initialization algorithm [10] as long as we restrict it to only pre-initialize states of the form $(p, (q, \epsilon))$. This allows us to benefit fully from the pre-initialization while keeping the ability to update the class-specific models on a per-utterance basis.

3.3. Handling Out-of-Vocabulary Words

One final issue is how to handle OOV words in the class-specific models. In the case of user contacts, it is extremely likely that for a large population of users, a large proportion of the contact names will not be part of our main vocabulary.

Since CL is built and optimized offline, it is difficult to add new names to it while keeping the result optimized and updating the reachability filter data accordingly. Instead, we choose to use a form of LG-level replacement for the classes susceptible to contain OOVs.

We modify our main L by adding a single-phone *monophone word* for every context-independent phone. We then create an FST M that maps each monophone word to the corresponding (context-independent) phone as in [3].

For the contacts class c or any class for which we want to allow OOVs, we build a c-specific restricted lexicon L_c containing pronunciations for the words in G_c and we build a new replace component for c as:

$$G'_c = M \circ \mathrm{Det}(L_c) \circ G_c. \tag{7}$$

Given $C' \subseteq C$, the set of classes for which we want to allow OOVs, we build the recognition transducer as:

$$T = CL \circ \mathrm{Replace}((G_r, G_c, G'_{c'})_{c \in C \setminus C', c' \in C'}) \tag{8}$$

where r denotes the root nonterminal.

This solves the OOV problem since (7) can be computed on a per-utterance basis just before recognition by the module gathering the user contact names using a grapheme-to-phoneme model.

4. CONTACT SELECTION BIASING

Although typical users have a very large number of contacts in their address books, they usually interact with only a small number of them on a regular basis. The frequency depends on the particular application they use (email, call, SMS, etc.). Using a large number of contacts to build a contacts class based language model can adversely affect recognition performance on contact related voice commands. Furthermore, it increases the risk of false triggering of voice commands. To reduce this risk while maintaining high recognition accuracy, affinity signals can be used to produce a ranked list of contacts per mode of interaction. This list can then be used to modify contact calling phrase LM costs accordingly.

For a call-contact voice command, we use relevant signals such as how many times the user has called a specific person (frequency affinity), and the time that has passed since the last call was made (recency affinity). Contact affinity is calculated as the sum of the frequency and recency affinities. Frequency affinity is normalized by the total number of calls. Recency affinity is an exponential decaying function of the time passed since the last call. The contact selection process that we use is adaptive and is executed on the user's device every time the user makes a query, which achieves maximum freshness of the ranked list, and captures events such as recently added or deleted contacts.

5. EXPERIMENTAL RESULTS

We conducted a large number of experiments by testing each of the improvements that we have made using offline and live testing approaches.

5.1. Corpora

For American English, we collected a data set of 10,670 contact calling voice command utterances. In addition, we collected an "anti-contact" data set consisting of 4,962 utterances. This set includes queries that are not contact calling voice commands but could potentially be recognized as such. It consists of business calling voice commands, or general voice queries (e.g. "call McDonald's", "call of duty"). We used this data set to evaluate overtriggering. Similarly, we collected contact and anti-contact data sets for two additional languages. This collected data was obtained from users who voluntarily opted-in into personalized speech recognition. All of the data sets were manually transcribed.

Language	Test Set	NUtts	CLG WER[%]	G WER[%]
German	CONTACTS	2,928	16.3	12.3
	CONTACTS-W-SEL	2,928	10.2	9.6
	ANTI-CONTACTS	2,937	33.3	26.0
British English	CONTACTS	2,588	22.4	21.2
	CONTACTS-W-SEL	2,588	20.5	16.8
	ANTI-CONTACTS	2,740	47.0	31.7
American English	CONTACTS	10,670	10.7	7.0
	CONTACTS-W-SEL	10,670	7.9	6.0
	ANTI-CONTACTS	4,650	15.7	10.7

Table 1. *Contact recognition experiment results.*

WER[%]		α							
		0		-1		-2		-3	
β	0	9.3	*10.5*	8.2	*10.5*	7.3	*10.5*	6.5	*10.6*
	0.25	6.9	*10.5*	6.4	*10.6*	6.1	*10.7*	6.0	*10.9*
	0.5	6.0	*10.7*	5.9	*11.0*	5.8	*11.6*	5.7	*13.1*
	0.75	5.7	*12.1*	5.8	*14.2*	5.8	*18.4*	6.4	*26.5*
	1	6.1	*21.7*	7.0	*32.4*	8.4	*50.0*	11.1	*73.6*

Table 2. *Sweeping α and β WER results on American English for the Contacts and, in italic, Anticontacts test sets.*

5.2. Contact and Anti-contact Recognition Accuracy

We trained language models and built recognition transducers using methods described in 3.1 and 3.2. The same language modeling pipeline was used to obtain both language models resulting in models of comparable contents and total number of parameters. As seen in Table 1, the class based LM that uses G level replacement outperforms the CLG level replacement LM on CONTACTS data sets for all languages tested: American and British English and German.

We also performed experiments in which we processed the contact lists using affinity signals. We repeated experiments on contact datasets and observed increased recognition accuracy (see rows CONTACTS-W-SEL in Table 1). In addition, we performed experiments on anti-contact data sets (see rows ANTI-CONTACT in Table 1). Again, the first system caused less false triggering than the second for all languages tested.

The results in Table 1 for American English were obtained with the parameters from Section 2.1 set to $\alpha = 0$ and $\beta = 0.5$. These values were chosen based on the results of the sweep reported in Table 2. The optimal values are chosen so that WER increase on anticontacts test sets is minimal while contacts test set WER decrease is still significant. This shows that Approach 3 allows us to obtain a better operating point than Approach 1 ($\alpha = 0$, $\beta = 1$) or Approach 2 ($\alpha = \beta = 0$).

5.3. Live Contact Selection Biasing Experiments

To further validate that our system improvements are beneficial to users, we also ran live traffic "holdback" experiments following the launch of our G-level replacement system using optimal vaules for α and β described above. In such experiments a random sample of utterances was processed by an alternate recognizer that did not use the optimized list of contacts. After several weeks, between 15,000 and 20,000 utterances were processed by the experimental and control servers. This was done anonymously and on-the-fly without saving any user data.

We found that using contact selection biasing described in section 4 increased the "call contact" voice command card triggering rate by 9% relative, indicating better recall (fewer false negatives) for recognizing utterances that are intended to be call-contact commands. This was significant to $p<.05$. The rate at which triggered cards were accepted (e.g., the user actually placed the call) improved by 4% relative, indicating better precision (fewer false contact triggers). This was also significant, $p<.001$.

To further test against false triggering of contacts, we examined the performance of non-contact voice commands (such as "call business") and general voice search. None experienced a significant degradation in either triggering rates or acceptance rates.

6. CONCLUSION

In this paper, we analyze various components of a contacts voice command system. We show that using class based language models and G-level replacement can solve the problem of missing context and OOVs, resulting in improved recognition accuracy. We also show the importance of careful selection of the LM training data as well as the importance of adaptive analysis of contact affinities and insertion reward. Our experimental results confirm that the proposed methodology can significantly improve speech recognition accuracy (up to 20% relative WER reduction) on voice commands containing contact names. Furthermore, this was achieved without causing overtriggering on non-contact related queries.

7. REFERENCES

[1] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[2] Sameer R. Maskey, Michiel Bacchiani, Brian Roark, and Richard Sproat, "Improved name recognition with meta-data dependent name networks," in *ICASSP 2004*, 2004, pp. 789–792.

[3] Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose, "Dynamic grammars with lookahead composition for WFST-based speech recognition," in *Interspeech 2012*, 2012.

[4] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Yiteng Huang Jacob Benesty, Mohan Sondhi, Ed., pp. 559–582. Springer, 2008.

[5] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *CIAA 2007*, 2007, vol. 4783 of *LNCS*, pp. 11–23, `http://www.openfst.org`.

[6] Cyril Allauzen and Michael Riley, "Bayesian language model interpolation for mobile speech input," in *Interspeech 2011*, 2011, pp. 1429–1432.

[7] Françoise Beaufays and Brian Strope, "Language model capitalization," in *ICASSP 2013*, 2013, pp. 6749–6752.

[8] Cyril Allauzen, Michael Riley, and Johan Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *Interspeech 2009*, 2009, pp. 1203–1206.

[9] Cyril Allauzen, Michael Riley, and Johan Schalkwyk, "Filters for efficient composition of weighted finite-state transducers," in *CIAA 2010*. 2011, vol. 6482 of *LNCS*, pp. 28–38, Springer.

[10] Cyril Allauzen and Michael Riley, "Pre-initialized composition for large-vocabulary speech recognition," in *Interspeech 2013*, 2013, pp. 666–670.