DEPLOYING GOOG-411: EARLY LESSONS IN DATA, MEASUREMENT, AND TESTING

Michiel Bacchiani, Françoise Beaufays, Johan Schalkwyk, Mike Schuster, Brian Strope

Google, Inc.

ABSTRACT

We describe our early experience building and optimizing GOOG-411, a fully automated, voice-enabled, business finder. We show how taking an iterative approach to system development allows us to optimize the various components of the system, thereby progressively improving user-facing metrics. We show the contributions of different data sources to recognition accuracy. For business listing language models, we see a nearly linear performance increase with the logarithm of the amount of training data. To date, we have improved our correct accept rate by 25% absolute, and increased our transfer rate by 35% absolute.

Index Terms— business finder, directory assistance, voice search, speech recognition.

1. INTRODUCTION

GOOG-411 [1] is a speech-enabled business finder. Users are prompted to say a city and state, and then a specific business name or a business category (e.g. “hardware stores”). A speech recognition system translates the user’s spoken request into a query that is fed to a web-based business search engine, Google Maps [2]. Maps returns a sorted list of businesses. Depending on how closely these match the user’s query, one to eight results are spoken back to the user, using text-to-speech (TTS). Users can select a specific result, get connected to the business, or request an SMS with business information and map. GOOG-411 currently works only in English, and covers tens of thousands of cities throughout the United States.

The concept of speech-enabling 411 services has been researched for quite a few years (see e.g. [3, 4, 5]), and has been implemented in various services, including 555 Tell [6], Live Search 411 [7], and Free 411 [8] in the US. GOOG-411 was to our knowledge one of the first deployed 411 service that included full business and category search, and did not rely on human operators to handle difficult queries. Our underlying assumption in making this choice was that by taking an iterative approach, with data and appropriate metrics, the system would keep improving over time. Having back-off operators changes the way users interact with the system. To avoid this and focus on the end solution, we decided to not use operators from the start.

After a brief overview of the system architecture, we describe the data pipeline and measurements we put in place to improve GOOG-411. We then focus more closely on two key components, the acoustic models and language models, and we conclude by reviewing high-level metrics and their progress over time.

2. SYSTEM ARCHITECTURE

Figure 1 shows the main components of the GOOG-411 system. These include the telephony network, an application server that executes the voice application, a TTS server, a recognition server with its acoustic, language, and pronunciation models (AM, LM, PM), the Google Maps service to execute business queries, and an SMS gateway to send information to mobile users. Each one of these components contains its own redundancy and load balancing capabilities. Because this is a dialog system, many of the processes interact asynchronously, making the overall system a fairly complex state machine. We leverage the Google infrastructure (machine grid, GFS [9], Bigtable [10]) for redundancy, automated fail-over, and multi-homed implementation to ensure the reliability and scalability of the service. Live probers are used to monitor the system, and real-time unsupervised statistics allow us to monitor its quality.

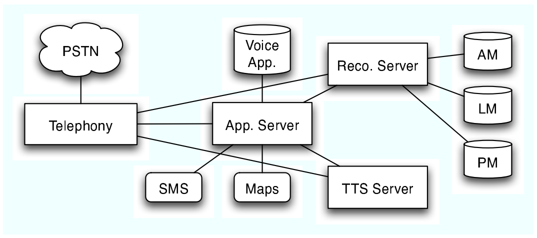

Fig. 1. GOOG-411 Block Diagram.

3. DATA PIPELINE & MEASUREMENTS

An essential aspect of our design is that of data-driven optimization. To this end, we built an extensive data pipeline. All incoming calls are analyzed to monitor the health of the system (check for component failures, etc), and to track its quality (e.g. what fraction of calls reach a given point in the dialog). The data is then saved, transcribed, and used for further supervised analysis as well as to retrain the main components of the system. These are then updated in the live system. At any time, the most recent data is used for testing, everything else is folded into the training sets. This rolling test set approach allows us to stay in tune with UI and infrastructure updates, to track changing usage patterns, and to avoid overfitting to stale test sets.

To measure recognition accuracy, our primary metric is receiving-operating curves (ROC), that show correct-accept (CA) against false-accept (FA) rates. These are evaluated at the sentence-level, over the semantic interpretation of the recognition results, e.g. recognizing "um italian restaurants" instead of "italian restaurant" is considered a correct accept (provided that the confidence value is superior to some predefined threshold, else it would be a reject).

At a system level, we measure the transfer rate, i.e. the fraction of calls in which users connect to businesses or receive an SMS with business details. Though simplistic, this metric is a good first-order approximation to user happiness. As illustrated in Sec. 6, it reflects UI changes, infrastructure improvements, and accuracy increases.

Finally, at a product level, we track the increase in traffic, which is another indication of the success of the service.

In the next two sections, we describe in more details our experiments in acoustic and language modeling. When reporting results, here and in Sec. 6, we deliberately omitted *absolute* performance values, showing relative improvements instead. This is for competitive reasons, and because absolute numbers are easily misquoted when taken out of their exact context. For example, accuracy figures depend on whether we keep or eliminate from the test sets sentences that contain no understandable speech or no speech directed at the service, and how frequent these are. It also depends on whether the back-end search is included in the scoring. Transfer rates depend on whether we include calls where the users hang up before giving any input. We hope that the relative metrics we do expose will nonetheless be informative to the research community.

The experiments described below reflect the performance of our live service, which interested readers can test by calling 1-800-GOOG-411. In general, the service has received positive user feedback, indicating that the underlying technology works at a level accuracy that makes it useful to users, and that is presumably comparable to commercial systems (with semantic sentence-level accuracies in the 50-80% range).

4. ACOUSTIC MODELING

The speech recognition engine is a standard, large-vocabulary recognizer, with PLP features and LDA, GMM-based triphone HMMs, decision trees, STC [11], and an FST-based search [12]. The trainer does maximum-likelihood optimization, and is implemented in a mapreduce framework [13], allowing us to train models in a matter of hours, even with large data sets, currently with a few hundred machines. The acoustic models compared in this section are gender-independent, one-pass, and are trained exclusively with GOOG-411 speech.

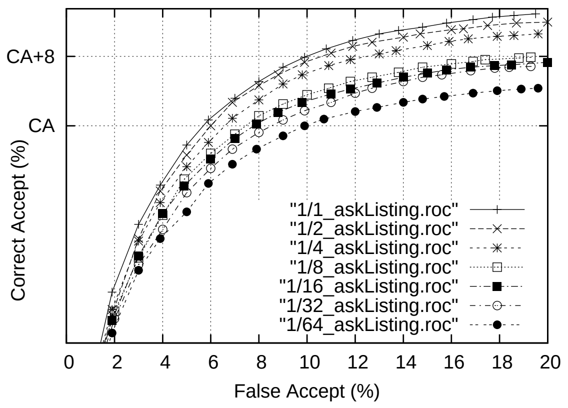

Fig. 2. AskListing performance as a function of the amount of acoustic training data.

Figure 2 shows the relative performance of a series of models trained with increasing amounts of data. The test set consists of roughly 20,000 recently-collected utterances spoken in response to "What business name or category?" (the askListing dialog state), spread over 3,000 cities. The language models and acoustic model architecture are held constant across experiments. The training sets were set up so that we show results with all training material, with the first half we collected, with the first quarter, the first eighth, etc. The largest training set contains on the order of thousands of hours of speech.

Interestingly, recognition performance does not increase dramatically with the amount of training data (8% absolute CA increase at 10% FA for a factor 64 increase in training size). Part of the reason may be that the training data is well matched to the test set, both phonetically and acoustically (the same users may even appear in both training and testing, in different calls of course, but probably on the same device, and sometimes speaking the same query). Another reason may simply be that we haven't explored that space much yet.

5. LANGUAGE MODELING

The language models are a combination of N-gram statistical language models (SLM) and context-free grammars. They are trained from three data sources that are weighted to optimize ROCs on development data sets.

First, we have a set of business and location databases: These provide coverage, but the official business names they contain don't always match what people say, e.g. "google inc." for "google" or "starbucks coffee" for "starbucks".

Second, we have Google Maps web query logs: This is a large corpus of typed queries that are better matched to

GOOG-411 (users have learned they can type just "google" or "starbucks"). Query counts are used to derive LM probabilities, however the priors in Maps and GOOG-411 don't always coincide: for example "real estate" is a frequent web query, but an uncommon speech request.

Third, we have speech data: Transcribed speech from GOOG-411 calls provides the best matched data.

In all experiments below, the language models are pruned to maintain close to real-time recognition, and limit the overall system latency.

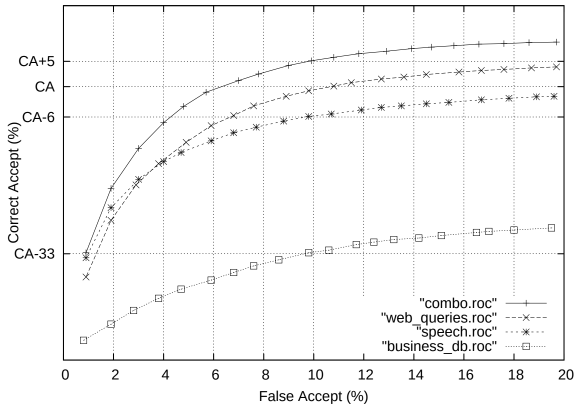

Fig. 3. AskListing performance as a function of the type of LM training data.

Figure 3 shows the performance of the askListing state as a function of the type of LM training data. Currently, the web log data produces the best LM, followed by the speech data, with a difference of 6% absolute CA between the 2 LMs at 10% FA. The business databases are much worse. Combining the 3 data sources (combo) gives an extra 5% CA gain over the web LM.

Figure 4 shows the performance of a business LM trained from speech data only, as a function of the amount of speech training data. Here again the training set sizes vary by factors of 2. Since the corresponding ROCs are roughly equally spaced, we can conclude that the accuracy of the LM grows linearly with the log of the amount of training data. At this rate, the contribution of the speech data will match that of the (current) web queries when we collect just 4 times more speech data (even though this will still be orders of magnitude less than the amount of web data).

Figure 5 shows the performance of the askCityState dialog state, as a function of the type (web, speech, combo) and amount of speech LM training data. Being a simpler task, askCityState improves less than linearly with the log of the amount of data. The speech data performs as well as the web data, which was extensively processed to parse city-states from queries expressed in a variety of formats (full street addresses, etc). Without this processing, the web data showed very poor rejection performance. Combining the speech and (processed) web data gives the best CA. The databases did not provide any additional benefit.

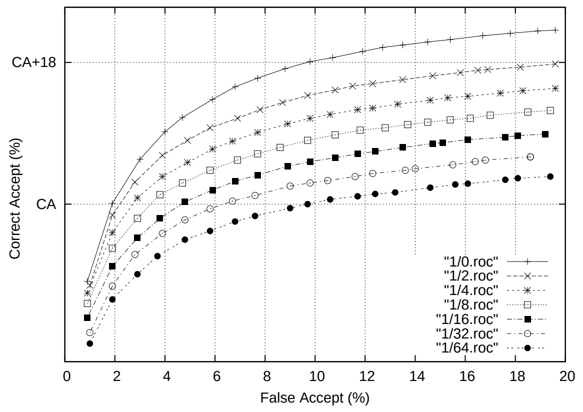

Fig. 4. AskListing performance for the speech-data-trained LM as a function of the amount of LM training data.

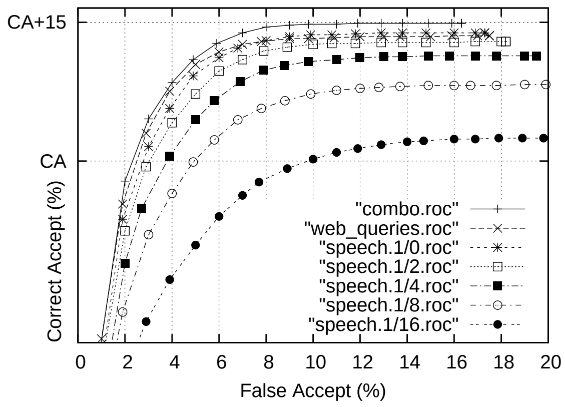

Fig. 5. AskCityState performance as a function of the type and amount of LM training data.

6. OVERALL PERFORMANCE IMPROVEMENTS

This section is intended to give a global perspective on the evolution of GOOG-411 over time. Many factors are combined, including updates of statistical models, UI experiments, infrastructure changes, bug fixes, and also extraneous events such as holidays that affect the usage patterns of the service.

Starting with recognition performance, Fig. 6 illustrates our improvements on the askListing state, as a function of time. Each ROC is measured on a different test set (see the concept of rolling test set in Sec. 3). The figure shows that at an FA rate of 10%, we improved the correct accept rate by roughly 25% absolute over the last 7 months.

Figures 7 and 8 show GOOG-411 daily number of incoming calls, and the daily transfer rate over a period of one year. A few interesting points are annotated. Points A and B on the traffic chart show traffic increases as we increased our advertisement campaign. Point C was a partial system outage.

Point E on both plots marks the official launch of GOOG-411, with a strong increase in traffic, and a large drop in transfer rate: users were experimenting with the system and didn't

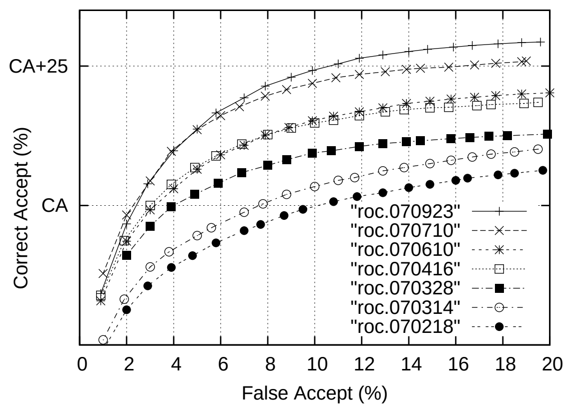

Fig. 6. AskListing performance over time.

care to complete their calls. Point D corresponds to the quiet period before the launch: we stopped advertizing the service (drop in call volume), so callers were mostly power users for whom the service works well (peak in transfer rate).

Point F on the transfer chart is the beginning of a UI experiment where we narrowed down the results presented to the users. This encouraged people to connect more often. At point G, we extended the connection feature to all US states.

Point H resulted from an interesting bug where a fraction of callers were offered incorrect results for a substantial number of queries. Users reacted with a drop in transfer rate. Points J and K indicate telephony infrastructure glitches, also causing transfer rate drops. Point I is the 4th of July: users were presumably more motivated to connect to businesses for their shopping needs.

We could analyze the curves in more details. Clearly traffic and transfer rate profiles won't tell us everything about user happiness, but they reflect a surprisingly broad set of events, and prove very useful in monitoring the health and growth of the system. There is nonetheless a fair amount of noise in the curves that results from a combination of factors that are not always easy to separate out. This complicates online experimentation. A small UI change for example may not clearly impact the transfer rate, even though a few of them eventually will. For these, we need to rely on more targetted metrics as well, and take the leap of faith that they will eventually convert in user happiness improvements, similarly to what we do when optimizing the recognition models.

7. CONCLUSION

In conclusion, we took an iterative-development approach to build, deploy, and grow a fairly complex speech-based system. We showed how, by focussing on data, measurements, and constant system refinements, we can quickly improve low-level metrics such as speech recognition accuracy as well as high-level user-related metrics. In general, we found that having access to the whole product stack with the flexibility to modify it as desired, and having a steady growing stream of data, are key factors in our ability to consistently improve the service over time.

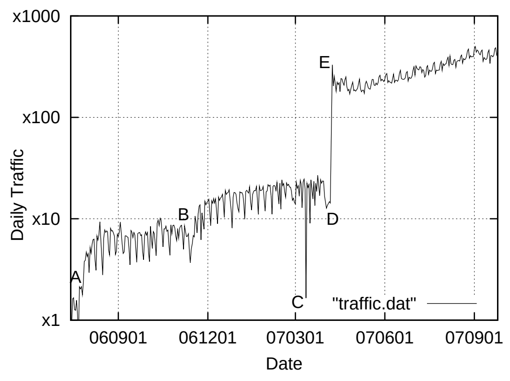

Fig. 7. Daily traffic as a function of time.

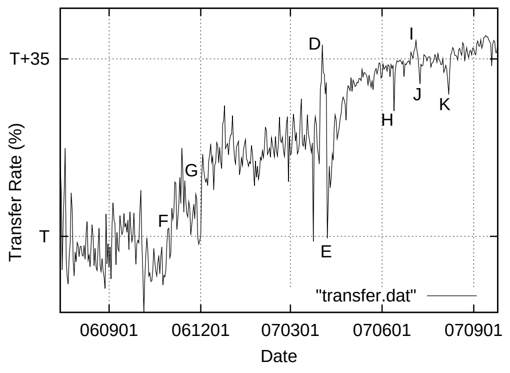

Fig. 8. Transfer rate as a function of time.

8. REFERENCES

[1] "GOOG-411," http://www.google.com/goog411.

[2] "Google Maps," http://maps.google.com.

[3] L. Boves et al., "ASR for automatic directory assistance: The SMADA project," in *Proc. ASR*, 2000, pp. 249–254.

[4] N. Gupta et al., "The AT&T spoken language understanding system," in *IEEE Trans. ASLP*, 2006, pp. 213–222.

[5] D. Yu et al., "Automated directory assistance - from theory to practice," *Proc. Interspeech*, 2007.

[6] "555 Tell," http://www.tellme.com/products/TellmeByVoice.

[7] "Live search 411," http://www.livesearch411.com.

[8] "Free 411," http://www.free411.com.

[9] S. Ghemawat et al., "The google file system," in *Proc. SIGOPS*, 2003, pp. 20–43.

[10] F. Chang et al., "Bigtable: A distributed storage system for structured data," in *Proc. OSDI*, 2006, pp. 205–218.

[11] M.J.F. Gales, "Semi-tied covariance matrices for hidden markov models," *Proc. IEEE Trans. SAP*, May 2000.

[12] "OpenFst Library," http://www.openfst.org.

[13] J. Dean et al., "Mapreduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004, pp. 137–150.