EXPLORING LANGUAGE MODELING ALTERNATIVES WITH GOOG-411

Charl van Heerden[*]

HLT Research group, Meraka Institute, CSIR

Johan Schalkwyk, Brian Strope

Google, Inc.

ABSTRACT

We compare different language modeling approaches for Google's telephone-based business finder, GOOG-411. Specifically we explore accuracy tradeoffs between an architecture using a single large national LM and one using many small models adapted for particular cities. Experimental evaluations show that both approaches lead to comparable overall accuracy, with differences in the distributions of errors. These differences suggest a simple combination approach that leads to accuracy improvements. Finally to enable a less restrictive dialog system, we also evaluate variants of the national LM in the context of more open business queries from the web.

Index Terms— Language modeling, directory assistance, voice search, speech recognition

1. INTRODUCTION

Today successful commercial speech recognition systems typically depend on limited domains and strong language models in order to reach usable accuracy. For example most deployed dialog systems will prompt callers for what they can say (e.g. "what city and state?"). As computational capacity, available data, and model sizes have grown, systems are providing usable accuracy for increasingly open tasks, which in turn provide increasing value to users.

Most directory-assistance applications make use of speech recognition (e.g. 800-555-TELL, 800-FREE-411, 800-CALL-411, http://www.livesearch411.com). Some detail about approaches considered have been presented [1, 2, 3]. These systems currently leverage the existing directory assistance UI-model that starts with "what city and state?" This is in contrast to more general call-routing applications, "how may I help you?" [4], and increasingly in contrast to web search engines for local information (e.g. http://maps.google.com) which have migrated to a single text-input box.

800-GOOG-411 is an automated system that uses speech recognition and web search to help people find and call businesses. By first asking callers for the city and state, the system can make the probabilities for the expected business queries conditional on each city. Removing that conditional dependency opens the domain and moves a step toward increasingly flexible interactions. This direction raises two experimental questions: 1) Can a single large national LM make up for what might be lost when city-specific information is not available with an open dialog? and 2) Can models derived from separate sub-dialog states predict natural combinations of those states?

[*]Work performed during Google intership.

This paper examines some of the language modeling challenges for allowing more flexible user-input for business directory assistance. Section 2 describes the data sources and text processing used to estimate the language models. Section 3 describe the basic design of the two language modeling approaches considered. Section 4 compares experimental results for these two approaches, and for a simple combination strategy. Section 5 describes the extension of the national LM to a system that predicts less restrictive business queries.

2. DATA SOURCES AND TEXT PROCESSING

Table 1 summarizes the three data sources used to train our language models. The GOOG-411 transcripts were collected over a little more than one year, and the sample of Google Maps queries considered business queries collected over the last two years. The business databases are a compilation of commercially available US business listings. Each of these data sources has its own normalization challenges.

Previously [5] we found that the US business listings were not very helpful, and did not consider that source for the national LM described below. This choice is in contrast to others who have made more creative use of similar databases [6].

Table 1. Data sources

Source	Relevance	Uniq #	Total #
GOOG-411 trans.	highest	2M	10M
Google Maps	high	20M	2B
Business DBs	low	20M	20M

Fig. 1 shows an overview of the two-step data preparation phase used to process the data sources. All processing is implemented using MapReduce [7]. The first step is a distributed computation that normalizes and annotates the input data. Common text normalization is performed in this step: for example, converting all characters to lower case, and removing certain punctuation characters.

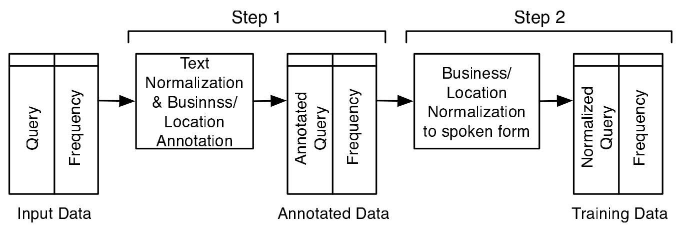

Fig. 1. Steps of the data preparation phase.

Annotation then involves identifying substrings of a given query that are either a business/category or a location. For example, consider the query "looking for restaurants in portland oregon." A business annotator identifies "restaurants" as a business/category, while the location annotator identifies "portland oregon" as a city-state. This separation helps with task-specific data selection (location and business), and also enables location normalization.

Spoken location modeling requires normalizing of state abbreviations. For example the location query "park in" should be normalized as "park indiana", but only when it is tagged as a location. After the data is also annotated, it is possible to perform context-aware normalization on the substrings labeled as business or location. This normalization takes place in the second step of the data preparation phase. We use the output of this step to build the business and location language models described in Section 3.

3. LANGUAGE MODELING ALTERNATIVES

3.1. Single national LM Estimation

To build the national business model, two individual models were trained from our normalized sources using ProdLM [8], Google's distributed language model training framework. The model estimated from Maps queries included words that occurred 3 or more times, while no vocabulary pruning was necessary for the model of GOOG-411 queries. Kneser-Ney smoothing [9] was used for both models.

Interpolation weights were estimated by evaluating perplexity (PPL) on a development set. A weight of approximately 0.8 for the GOOG-411 model was best for this evaluation. The interpolated model was then entropy pruned [10] to fit into the memory of a single machine.

Table 2. N-gram sizes for individual & interpolated models

Model	# 1-grams	# 2-grams	# 3-grams
GOOG-411	0.11M	0.93M	1.8M
Maps	0.50M	5.9M	13M
Interpolated	0.51M	6.1M	14M
Pruned interpolated	0.51M	5.9M	8.2M

The resulting N-gram sizes for the two models, and their combination are shown in Table 2. A similar national N-gram was also estimated for locations, using the GOOG-411 data from that dialog state together with the location information extracted from Maps queries. The resulting model, citystate-LM, is smaller than the business-lm and is described in more detail in Section 5.

3.2. City-Specific LM Estimation

The city-specific system uses a hybrid LM that combines full sentence queries with a weighted unigram. Individual unigram and sentence query weights were estimated from frequency counts in the training data. The weighting of the unigram with respect to the sentence queries was optimized on development tests. Initially the unigram was employed as a decoy path for rejection, but because high-confidence values through the unigram were often correct, we optimized independent confidence thresholds for the distinct grammar paths.

To compensate for data sparcity, the city-specific models were smoothed against increasingly general models of region, state, and country. Interpolation weights for data sources and regionalization were optimized on a development set. For these experiments, the size of the city-specific system was held constant across cities. Smaller cities with fewer queries ended up getting less city-specific data in their models.

The city-specific system also includes a semantic stage for inverse text normalization. This stage maps the query variants like "comp usa" and "comp u s a," to the most common web-text query form "compusa".

The city-specific system was designed earlier and there were a few implementation differences from the national-LM system: it used a different text normalization framework without explicit spell checking; the data sources were sampled at slightly different times; and all optimizations minimized "sentence semantic error" which considered the mis-match of the output of the inverse text normalization.

4. EXPERIMENTAL RESULTS

4.1. Tests and Measures

The recognition tests were held out and manually transcribed GOOG-411 calls. The tests were split into city-state (47K utterances) and business queries (56K). We report three types of measures: perplexity, word error, and sentence error. For sentence error, we ignored inconsistency in spaces and apostrophes (kinko's vs kinkos). All systems were evaluated with the same fairly standard acoustic models (triphones with decision tree clustering, 16 gaussians / state, 3 state units, using STC, and ML estimation followed by MMI, with a PLP-based front-end and LDA).

Table 3. Error analysis: WER/SER (%)

Data group	% of queries	National LM	City-Specific LM	Combination
all	100	21.8 / 29.0	23.1 / 26.8	20.3 / 24.3
national head	41	10.5 / 11.6	12.5 / 11.4	12.2 / 11.0
national body	25	19.4 / 25.4	20.8 / 22.7	18.9 / 20.5
national tail	34	31.5 / 52.8	32.1 / 48.5	26.9 / 43.3
big cities	33	23.1 / 31.2	27.0 / 31.8	22.6 / 27.8
medium cities	33	20.9 / 27.7	21.3 / 24.5	19.1 / 22.8
small cities	33	21.5 / 28.1	20.6 / 23.8	18.8 / 22.3
city head	64	14.5 / 17.9	11.3 / 11.3	11.1 / 11.0
city body	14	26.0 / 38.3	21.6 / 29.5	19.8 / 27.7
city tail	21	34.9 / 56.4	48.2 / 71.6	39.5 / 62.1

4.2. Overall Performance

The top line of Table 3 shows WER/SER for all data for the national LM and city-specific LMs. While the performance is close, each system does better with the measure more closely related to its structure. There are at least two factors to consider in the evaluation. First, more of the word-errors in the city-specific system can be tracked to difference in compounding and apostrophes, which are ignored by the semantic measure used to optimize that system. Second, by using full-sentence query models without a smooth N-gram backoff, the city-specific system is more likely to get the utterance completely right or completely wrong.

An interesting result is that the performance of the large national LM which consumes only a couple GBs of memory is very close to the city-specific system that includes 100s of GBs of memory in total.

4.3. Error Analysis and Combination

Table 3 shows word error and sentence error for different subsets of the data on three systems: the national-LM system, the city-specific system, and a simple combination scheme described below. Each system uses a different approach to cover the tail of the distribution. The national LM covers the tail with a large N-gram, while the city-specific LM tries to cover the tail of the national distribution by having a list of the common queries for each specific city.

The "national" lines in the table show accuracy numbers for increasingly less common queries. The "head" contains queries that are in the most common 10K, the "body" contains queries from the next 290K, and the "tail" is everything else. From these lines, it's clear that the city-specific system provides a sentence error improvement as we move toward the tail of the distribution. The next two sections of the table show that the improvement is from city-specific modeling, and from the full-query structure of the city-specific system.

For the city-size lines in the table we've grouped all utterances by city and sorted those cities into three groups based on their frequency, with each group representing equal numbers of utterances. For this test, we can cover about a third of the data with the most common 70 cities, while it takes thousands of cities to cover the least common third. The result is expected, the national LM system does the best with the large cities that dominated our data, while the city-specific system does increasingly better with smaller cities.

The last lines of the analysis table use a city-specific reference for the definition of head and tail. The "city head" is test utterances that are in the most common 10K query list for each city, the "city body" is utterances covered by the rest of the city-specific query lists, and the "city tail" is utterances not modeled by the city-specific query lists. With this last set the difference is more plain: both systems are lousy, but the national N-gram does much better with the city-tail than the city-specific unigrams. For the rest of the queries, the city-specific query lists do much better than the national N-gram.

Based on these observations we investigated a simple combination scheme where we use the result from the city-specific system unless that result came through the unigram. In that case we take the result from the national LM. This gives us city-specific query models together with a smooth national back-off mechanism. Without optimizing the combination, we see large gains in both word error and sentence error. Most notably, on the tail of the national distribution, the combination improved the sentence error of the national LM by 9.5% and the sentence error of the city-specific system by 5.2% absolute.

5. PREDICTING A COMBINED DOMAIN

The single language modeling approach enables us to explore less constrained user-interfaces for dialog applications. To start modeling these types of interactions, we constructed two language models. The first was an interpolation between a citystate-LM and the large business-LM evaluated above. The second was an hierachical language model (HLM) [11] which explicitly includes paths for a business listing, a city-state or a combination of the two in a single dialog state. The topology language model for the HLM, shown in Fig. 2, was con-

Fig. 2. An HLM "What/Where" model

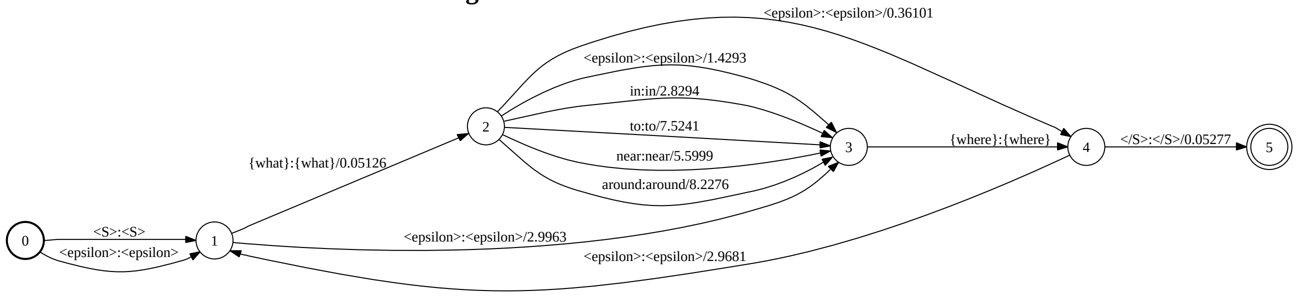

Table 4. Perplexity, accuracy, and combined domain: onebox

Model	# of N-grams	citystate PPL	citystate WER/SER	business PPL	business WER/SER	onebox PPL
citystate-LM	2.4M	14	5.8 / 8.3	-	-	-
business-LM	14M	48	9.4 / 12.7	80	21.8 / 29.0	441
bus-city-interp	14M	23	6.7 / 9.5	88	21.9 / 29.4	284
bus-city-HLM	14M	31	8.0 / 11.0	94	22.1 / 29.4	186

structed manually with arc weights estimated from a small "onebox" evaluation set (Google web queries that get a Maps business result). Table 4 summarizes errors across all models and across three testsets. The HLM does the best predicting more flexible text business queries, without degrading recognition performance on GOOG-411 tests.

6. CONCLUSION

This paper compared two language modeling approaches used for a commercial speech recognition system. The large national-LM system reached similar accuracy levels as the city-specific system. Error analysis suggested a combination stragegy which leads to accuracy improvement over either system. An HLM composition also best predicted a combined domain of more flexible business text queries.

7. ACKNOWLEDGEMENTS

We thank Boulos Harb, Will Neveitt, Vida Ha, Carolina Parada, Mike Schuster, Francoise Beaufays, Ciprian Chelba, and Greg Morris for providing tools, infrastructure, and consultation for the experiments reported in this paper.

8. REFERENCES

[1] N. Gupta, G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi, and M. Gilbert, "The AT&T spoken language understanding system," in *Proc. ASLP*, January 2006, pp. 213–222.

[2] S. Chang, S. Boyce, K. Hayati, I. Alphonso, and B. Buntschuh, "Modalities and demographics in voice search: Learnings from three case studies," in *Proc. ICASSP*, April 2008, pp. 5252–5255.

[3] A. Acero, N. Bernstein, R. Chambers, Y.C. Ju, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig, "Live search for mobile: Web services by voice on the cellphone," in *Proc. ICASSP*, April 2008, pp. 5256–5259.

[4] A. Gorin, G. Riccardi, J. Wright, "How may I help you?" in *Speech Communication* Vol. 23, Issues 1-2, October 1997, pp. 113-127.

[5] M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope, "Deploying Goog-411: Early lessons in data, measurement and testing," in *Proc. ICASSP*, April 2008, pp. 5260–5263.

[6] X. Li, Y.C. Ju, G. Zweig, and A. Acero, "Language modeling for voice search: A machine translation approach," in *Proc. ICASSP*, April 2008, pp. 4931–4916.

[7] J. Dean,S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004, San Francisco, CA.

[8] T. Brants, A.C. Popat, P. Xu, F. Och, J. Dean, "Large language models in machine translation," in *Proc. EMNLP-CoNLL*, June 2007, 858-867.

[9] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proc. ICASSP*, May 1995, pp. 181–184.

[10] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, February 1998, pp. 270–274.

[11] L. Galescu and J. Allen, "Hierarchical Statistical Language Models: Experiments on In-Domain Adaptation", in *Proc. ICSLP*, 2000, vol.1, 186-189.