CONTEXT-SENSITIVE LANGUAGE MODELING FOR LARGE SETS OF PROPER NOUNS IN MULTIMODAL DIALOGUE SYSTEMS

Alexander Gruenstein and Stephanie Seneff

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar St, Cambridge, MA 02139, USA
{alexgru,seneff}@csail.mit.edu

ABSTRACT

We explore several language modeling strategies for increasing the recognition accuracy among large sets of proper nouns in a map-based multimodal dialogue system which provides restaurant information. In particular, we evaluate several mechanisms for exploiting dialogue context, the two most promising of which involve a semi-static *metropolitan-region* based large set of proper nouns *competing* with a smaller, *in-focus* subset. We show that these techniques decrease word, concept, and proper noun error rates under several training conditions. We also present a technique to generalize sparse training data through derived templates to improve language model robustness.

Index Terms— multimodal dialogue system, language modeling, context-sensitive, restaurants, proper nouns

1. INTRODUCTION

The quality of a user's interaction with a spoken or multimodal dialogue system is usually highly dependent on the ability of the system to accurately recognize that user's spoken utterances. In order to improve recognition, many dialogue systems (especially commercially available systems) often take strong *system initiative* in which they actively constrain the set of allowed utterances by asking the user direct questions (for example: *Please provide a departure city*). Generally, such systems will provide language models specific to each dialogue state, prepared only to handle a narrow set of expected utterances. In contrast, many research systems provide more *natural*, *mixed initiative* dialogue. A major difficulty of such systems is that it generally becomes much more difficult to correctly recognize what a user says, as utterances become longer and less constrained.

While many *mixed-initiative* systems use a single, static language model regardless of current dialogue context, an area of active interest is in dynamically updating a speech recognizer's language model based on contextual cues. For instance, two pass systems utilizing small, dialogue-state-dependent language models in the first pass have proved successful in increasing recognition accuracy [1], as have systems which use an interpolation of dialogue-state specific n-gram models with larger domain models (see *e.g.* [2, 3]). And, in [4], we demonstrate that dynamic classes – n-gram classes whose contents can be dynamically modified at run time – can be populated in a context-sensitive manner with just a small set of phrases based on values in a dialogue system's information state to improve recognition accuracy.

This research is sponsored by the T-Party Project, a joint research program between MIT and Quanta Computer Inc., Taiwan.

U_1:	Show me Chinese restaurants in Cambridge
S_2:	There are 15 Chinese restaurants in Cambridge. [*shown on map*]
U_3:	What's the phone number of the Royal East?
S_4:	The phone number of the Royal East is (617) 661-1660.
U_5:	Are there any Italian restaurants along this street. [*draws line*]
S_6:	There are 12 Italian restaurants along this street. [*shown on map*]
U_7:	Show the web page for this one. [*circles a restaurant*]
S_8:	OK. [*displays web page from which the data was harvested*]
U_9:	[*Clicks menu to select "San Francisco" from metro area list*]
U_{10}:	Are there any cheap restaurants near 100 Santa Cruz Avenue in Menlo Park?
S_{11}:	There are five inexpensive restaurants near there. [*shown on map*]

Fig. 1. A very brief example interaction, labeled with U for user, S for system. Gestures and system actions are bracketed. Some system remarks have been shortened for brevity.

In this paper, we further explore techniques that utilize dynamically modifiable n-gram classes to reconfigure the recognizer's language model in real time using dialogue context. We focus here specifically on recognizing restaurant and street names in a mixed-initiative, multimodal restaurant guide system, described in [5]. At any time while interacting with the system, a user may make a request involving one out of several thousand restaurant or street names – correctly recognizing such utterances can prove daunting, an observation confirmed in at least one similar dialogue system [6]. Here, we explore techniques to dynamically bias the system's language model based on the set of restaurants and streets which the system has deemed to be *in-focus* at any given time, hypothesizing that a user is more likely to ask about a restaurant currently displayed on the map in front of her than some unrelated restaurant. Our methods should be generally applicable to many domains involving large sets of proper nouns, especially multimodal systems where subsets of the proper nouns can be graphically presented. Examples include multimodal interfaces to MP3 players (*e.g.* [6, 7]).

2. THE MULTIMODAL RESTAURANT GUIDE

The restaurant guide is a multimodal dialogue system which is accessible via a dynamic webpage. Users navigate to the page in a web browser, and can then interact with the system multimodally by talking, typing, drawing on the map, or clicking on restaurants displayed on the map. A very brief example dialogue shown in figure 1 demonstrates some capabilities of the system. By drawing on the map while speaking, users can indicate geographical areas of interest via circles, points, and lines, and can also circle sets of restaurants. The gestures are then disambiguated in the context of the utterance, as in utterances U_5 and U_7 in figure 1. For a detailed

description of the architecture, please see [5]. Videos demonstrating an interaction with the system can be viewed at `http://www.mit.edu/~alexgru/rest-videos/`.

The system differs from similar dialogue systems we are aware of ([6, 8]) in that it has access to large restaurant databases harvested from the web (see [5]), comparable in size to those found on commercial, web-based restaurant guides. Each database of restaurants encompasses a metropolitan region in the United States. At the time of the experiments discussed here the system supported five major metropolitan areas: Boston, Chicago, San Francisco, Seattle, and Washington DC. These metropolitan areas consist of 50-250 nearby cities, containing approximately 3,000 to 9,000 restaurants (excluding filtered-out fast food restaurants). In addition, a geographical database of street names permits the recognition of arbitrary street addresses and intersections.

3. DYNAMIC LANGUAGE MODELING

We utilize dynamic classes in the class n-gram language model to accommodate several sets of proper nouns used in the restaurant-guide. The within-class weights for the following classes are uniform, with the exception of street names, which are weighted according to the number of restaurants on that street (see [5]).

Restaurant Names The names of all the restaurants in a given metropolitan area. This set is, in fact, larger than the actual number of restaurants since we also include alternative ways of referring to each restaurant. For instance, *Caprice Restaurant and Lounge* can also be referred to as *Caprice Restaurant* or simply *Caprice*. This set of aliases is semi-automatically created, as described in [5], yielding name sets containing 5,860 to 11,247 aliases, depending on the metropolitan area.

Street Names The names of all the streets in a given metropolitan area. In the experiments below, we drop suffixes such as "street" and "road," leaving these outside of the dynamic class. The fact that street names are often reused significantly reduces the size of the set, yielding from 1,177 to 2,243 street names per metropolitan area.

Cities and Neighborhoods The set of cities and neighborhoods in each metropolitan area, approximately in the range of 50 to 250. Since this set is relatively small, we currently only change it depending on the active metropolitan area, according to the *per-metro* configuration described below.

In the next section, we describe four methods we have developed for dynamically manipulating the restaurant and street name classes in the dialogue system's language model. An evaluation of the four methods is given in the following section.

3.1. *Per-metro* configuration

The most straightforward configuration is the *per-metro* configuration, in which each set of proper nouns has a corresponding dynamic class in the language model, which we'll refer to as *$restaurant* and *$streetname*. At run-time, whenever the user switches to a different metropolitan area, the contents of each dynamic class is updated to contain all restaurant and street names in that area. The contents of the classes remains fixed for as long as the user stays in that metropolitan region, yielding an essentially *static* configuration: any restaurant or street is available at any time in a given metropolitan area.

3.2. *Context-specific* configuration

In the *context-specific* configuration – first developed in [9] – each set of proper nouns has a single corresponding class in the language model, which we'll refer to here as *$dynrestaurant* and *$dynstreetname*. However, in contrast to the previous approach, each dynamic class is populated only after some relatively small set of restaurants and streets have been explicitly brought into focus. For instance, while both dynamic classes are empty while recognizing utterance U_1 in figure 1, by the time we reach U_3 they have been populated with the names of the 15 Chinese restaurants in Cambridge and of all the streets in Cambridge respectively.

Critically, the contents of the classes can also be reconfigured *between recognition passes* on a single utterance. Using an efficient two-pass approach of MIT's SUMMIT recognizer (see [10]), a second recognition pass can take place if the system determines that the contents of the dynamic classes should be reconfigured. An example of this is U_{10}, in which the mention of a new city (*Menlo Park*) in the first recognition pass triggers the system to load the names of the restaurants and streets in that city before performing a second pass. To facilitate this capability, each dynamic class always supports an out-of-vocabulary (OOV) model represented acoustically by a phone n-gram. In the first recognition pass for U_{10}, the street name *Santa Cruz* can match this OOV model, resulting in the token *$street_oov* appearing in the first pass recognition result.

The *context-specific* configuration is particularly appropriate if the system can, with high accuracy, choose the correct subset of proper nouns with which to fill the dynamic classes. This can work well if a user can come to understand this limitation. After watching naive users interact with the system, however, it becomes apparent that this is not a natural restriction for users – they do not immediately understand that they can use a specific restaurant or street name in some contexts but not in others. To address this problem, we have developed two further approaches which attempt to meld the *per-metro* and *context-specific* configurations discussed thus far.

3.3. *Independent-competing* configuration

The *independent-competing* configuration is inspired by language model techniques first developed in [4] in which very small context-specific dynamic classes are populated based on dialogue context in the flight reservation domain. Often, these classes *compete* against larger static classes of proper nouns (for instance, recently mentioned airline names *compete* in a separate dynamic class against the set of all airlines). In this approach, co-occurrence statistics for both the small and large class are incorporated directly into the language model by properly tagging instances of each class in the training data. Thus, variations on the relative distributions of the *context-specific* as compared with the larger static class are naturally and precisely captured in the process of training an n-gram language model, because each class has its own independent distribution in the training data.

In this domain, we can apply the same basic principle with a slightly different twist. For both restaurant and street names, we include the dynamic class types developed for both the *per-metro* and *context-specific* configurations, and allow them to compete against one another. For instance, in our language model we will have two independent dynamic classes for restaurant names: *$restaurant* and *$dynrestaurant*, where the former behaves as in the *per-metro* configuration, and the latter as in the *context-specific*.

Training the n-gram now requires first differentiating the *$restaurant* and *$dynrestaurant* classes in the training data. This can be accomplished – as in [4] – by using the dialogue system log files to appropriately tag each street or restaurant name in each utterance as being in one of these two sets. In our case, we tag any restaurant or street name which the system deemed to be *in-focus* during a partic-

ular utterance as an instance of the *context-specific* dynamic classes, while all others are tagged as the *per-metro* classes.

Finally, we note that in this configuration we do not include a corresponding *OOV* class, since all restaurant and street names are active at all times in a given metropolitan area.

3.4. *Tied-competing* configuration

The *tied-competing* approach is similar to the previous approach in that we again allow all restaurant and street names in a metropolitan area to *compete* with a smaller, context-specific set. Unlike the previous approach, however, we here make the assumption that the occurrences of the *context-sensitive* versions of classes are not independent of the *per-metro* versions; instead, they are *tied*. This frees us from having to tag our training data based on the dialogue-system context in which it was collected. Instead, after training, we simply modify the language model directly such that, instead of a single dynamic class, we now have two competing dynamic classes in parallel. The modifications work just as described in [9] for adding the competing *OOV* class in the *context-sensitive* configuration; however instead of a parallel *OOV* model, we now have a parallel *per-metro* dynamic class.

While this configuration does not capture the potential distributional differences between the two types of classes, it is nonetheless quite natural if these distributional differences are not pronounced, or if too little training data is available to characterize them. As a result, it may allow training data to be more effectively utilized than the previous configuration. Similarly, synthetic training data, or user utterances collected outside of actually using the system (*e.g.* through wizard-of-oz experiments) can be used, since properly tagging this data based on dialogue system logs is unnecessary.

4. EVALUATION

We recorded and transcribed utterances from a number of users' interactions with the system, while also logging the dialogue manager's information state as the conversations proceeded. Subjects, who ranged from naive to expert, were asked to use the system to find four appealing restaurants at which they hadn't eaten before: two in the Boston area, and two elsewhere. A total of 546 utterances were collected from 10 different subjects. An earlier set of 228 "pilot" utterances, collected during system development, was also used for language model support. All data collection took place using a Thinkpad X41 tablet in the lab, which allowed subjects to use a pen for multimodal input and a headset for speech input. At the time of data collection, the system was run in the *context-specific* configuration described in section 3.2. The language model was trained on a set of developer utterances, and hence differs from the language models described below. Results comparing the *context-specific* and *per-metro* configurations trained on this set can be found in [5].

The collected utterances were used to train and test language models in the four configurations described in the previous section. Given the limited size of our data set, we partitioned the *test* set into 10 distinct subsets, where each subset contains the utterances collected in a single user's interaction with the system. To test each language model configuration, we then utilized a leave-out-out procedure in which all of the *pilot* dataset, and 9 of 10 *test* subsets were used as the training data for a language model to be tested on the held-out subset.

In addition, we experimented with enhancing our limited training data by generating *synthetic utterances* using *templates* which generalize on the patterns observed in the collected utterances. First,

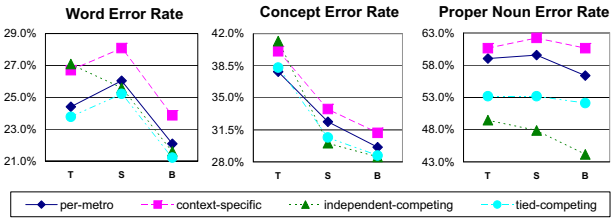

Fig. 2. Word, concept, and proper noun error rates of the 4 configurations trained on **T**ranscripts, **S**ynthetic data, or **B**oth.

a general set of core templates was derived, based on the observed patterns in the *pilot* data. Then, subset-specific templates were crafted from each user's data in turn, based on just those utterances that yielded a high perplexity when measured from a language model derived from the core templates. Finally, ten synthetic sets of 10,000 utterances each were created using the core templates augmented with all but one of the test template sets, using a leave-one-out procedure. Due to the limited training data, it proved difficult to generalize on any distributional differences which the *independent-competing* model aims to capture. In training this model, we guaranteed only an overall unigram statistic on the distribution of the *context-sensitive* compared to the *per-metro* dynamic classes. Specifically, we ensured that approximately 80% of restaurant names and 85% of street names were in the *context-sensitive* class, without attempting to capture any within-utterance distributional differences.

In the evaluations that follow, we trained and tested language models according to each of the configurations described in the previous section using three different sets of training data: (1) actual utterance transcripts, (2) synthetic utterances, and (3) a combination of both synthetic and transcribed utterances. All experiments were performed offline using the collected utterances, which has an important implication: the *in-focus* restaurant and street names in the *context-sensitive* dynamic sets are determined using the system logs from the user's original interaction with the system. As a result, the *context-sensitive*, *independent-competing*, and *tied-competing* configurations depend on the recognition capabilities of the original live system to determine the proper in-focus set for any given utterance to be recognized. Since the data were collected using a fixed language model configuration, trained on a different set of language model training data, our current tests are somewhat dependent on the quality of the original language models used. As the language model training data used here appears to be superior to that used to collect the data (the word error rates of the *context-specific* and *per-metro* model discussed below are lower than those of the original model[1]), we expect that the performance of the models reported here is a *lower bound* because it relies on the inferior models to determine the *in-focus* sets.

4.1. Overall Results

Figure 2 compares the performance of the four configurations as measured by word, concept, and proper noun error rates in each of the three training data conditions. Concept error rate, reported only on the 500 utterances where the transcript could be parsed, is determined by comparing the key concepts identified in the parse of the transcript to those found in the top recognizer hypothesis. The proper noun error rate, measured on *all* utterances in the test set, is calculated by comparing the restaurant and street names which ap-

[1]Unfortunately, the concept error rate reported in [5] is not comparable to the one reported here, since the parser coverage has been changed since the original tests.

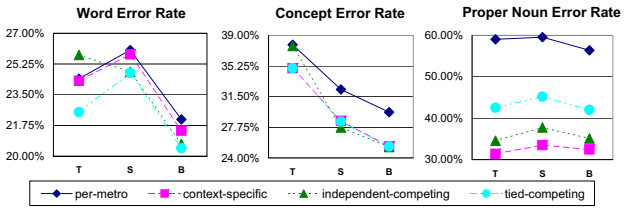

Fig. 3. Error rates given a contextual oracle

pear in the transcript to those found in the top recognizer hypothesis for each utterance, without attempting to parse the strings.

Several interesting observations can be made about figure 2. First, we note that in all three training conditions, the *per-metro* configuration outperforms the *context-specific* configuration in all metrics. This is interesting as it is the inverse of the results obtained in [5]. It seems to indicate that, as the language model training data improves, the *per-metro* benefits more than the *context-specific* configuration. However, this is not terribly surprising since, as we noted above, the performance of the *context-specific* configuration here represents a lower bound, as it depends on the inferior model used to collect the data.

More importantly, we note that the *independent-competing* and *tied-competing* configurations generally outperform the other two approaches, most notably in the proper-noun error rate. Both *competing* configurations reap the benefits of having both the *per-metro* and *context-specific* dynamic classes active. The *independent-competing* configuration performs poorly as measured by word and concept error rate when trained only on the transcripts, indicating that, in situations with only a small amount of training data, this model suffers from the divvying up of each dynamic class into two distinct sets. As the amount of training data increases, the two *competing* models behavior more similarly. However, the *independent-competing* model does outperform the *tied-competing* model in terms of proper noun error rate in all conditions, indicating that accuracy improves even with only (or mostly) unigram probability differences in the two types of dynamic classes. We hypothesize, however, that a penalty – derived from the unigram distribution – on the competing *per-metro* class in the *tied-competing* configuration might yield similar improvements.

In comparing the three sets of training data, it is quite interesting that the word error rate is generally highest when the models are trained only on the synthetic data, yet the concept error rate drops quite a bit nonetheless. A closer examination of the recognition hypotheses shows that a large portion of the degradation in word error rate can be attributed to the phrases "restaurant" or "a restaurant" appearing where "restaurants" should appear. A review of the synthetic data reveals that, in fact, the ratio of the occurrences of the word "restaurant" to "restaurants" is 2.4, whereas it is 1.4 in the transcribed set. Because our parser is generally robust to such distinctions, concept error rate is not adversely affected – in fact, it benefits quite a bit from the synthetic data.

4.2. Oracle contextual condition

The two *competing* configurations have been developed in order to incorporate contextual knowledge into the language model, while at the same time allow for the facts that (1) a dialogue system may make mistakes about what constitutes the current conversational context, and (2) sometimes a person will change contexts unexpectedly. To obtain an upper bound, it is informative to evaluate the language model configurations under an oracle condition, in which the dialogue system has always guessed correctly about the current context.

To simulate this condition, we tested all four language model configurations such that, when recognizing a particular utterance, the *context-specific* dynamic classes were always guaranteed to be populated with the set of proper nouns appearing in the transcript of that utterance. We note that the classes were not populated *only* with those proper nouns, but with whatever appeared in the original system logs, plus the correct proper noun (if that proper noun was not already in the list). Figure 3 shows the results of this experiment, and includes the *per-metro* model for reference (as it was unaffected by these manipulations).

The two *competing* configurations generally do almost as well as or better than the *context-specific* configuration in terms of word and overall concept accuracy. The *independent-competing* model performs almost as well as the *context-specific* model in terms of proper noun error rate. This result is encouraging as it shows that we take only a minor performance hit using the *competing* models, even under conditions where their large competing sets are superfluous.

5. SUMMARY

In this paper, we have explored methods to improve speech understanding in a multimodal dialogue system, by dynamically manipulating the language models. The highest performance is achieved by allowing a large set of proper nouns covering each metropolitan region to compete with a smaller *in-focus* set derived from the dialogue context. While the *independent-competing* configuration generally has the highest accuracy, the *tied-competing* configuration is appealing as it does not require detailed knowledge of the dialogue context to label the training data. Significant performance improvements can be achieved by training on a combination of the original transcripts and a set of synthetic utterances automatically derived from generalized templates.

6. REFERENCES

[1] O. Lemon and A. Gruenstein, "Multithreaded context for robust conversational interfaces: context-sensitive speech recognition and interpretation of corrective fragments," *ACM Transactions on Computer-Human Interaction*, vol. 11(3), pp. 241–267, 2004.

[2] A. Aaron et al., "Speech recognition for DARPA communicator," in *Proc. of ICASSP*, 2001.

[3] F. Wessel, A. Baader, and H. Ney, "A comparison of dialogue-state dependent language models," in *Proc. of ESCA Workshop on Interactive Dialogue in Multi-Modal Systems*, 1999, pp. 93–96.

[4] A. Gruenstein, C. Wang, and S. Seneff, "Context-sensitive statistical language modeling," in *Proc. of INTERSPEECH*, 2005, pp. 17–20.

[5] A. Gruenstein, S. Seneff, and C. Wang, "Scalable and portable web-based multimodal dialogue interaction with geographical databases," in *Proc. of INTERSPEECH*, 2006.

[6] F. Weng et al., "CHAT: A conversational helper for automotive tasks," in *Proc. of INTERSPEECH*, 2006.

[7] I. Kruijff-Korbayová, T. Becker, N. Blaylock, C. Gerstenberger, M. Kaisser, P. Poller, V. Rieser, and J. Schehl, "The SAMMIE corpus of multimodal dialogues with an mp3 player," in *Proc. of LREC*, 2005.

[8] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "MATCH: An architecture for multimodal dialogue systems," in *Proc. of ACL*, 2002.

[9] G. Chung, S. Seneff, C. Wang, and L. Hetherington, "A dynamic vocabulary spoken dialogue interface," in *Proc. of INTERSPEECH*, 2004, pp. 327–330.

[10] I. L. Hetherington, "A multi-pass, dynamic-vocabulary approach to real-time, large-vocabulary speech recognition," in *Proc. of INTERSPEECH*, 2005.