Language Model Data Filtering via User Simulation and Dialogue Resynthesis

Chao Wang †, Stephanie Seneff †, and Grace Chung ‡

† Spoken Language Systems Group
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139, USA
{wangc,seneff}@csail.mit.edu

‡ Corporation for National Research Initiatives
1895 Preston White Drive, Suite 100, Reston, VA 22209, USA
gchung@cnri.reston.va.us

Abstract

In this paper, we address the issue of generating language model training data during the initial stages of dialogue system development. The process begins with a large set of sentence templates, automatically adapted from other application domains. We propose two methods to filter the raw data set to achieve a desired probability distribution of the semantic content, both on the sentence level and on the class level. The first method utilizes user simulation technology, which obtains the probability model via an interplay between a probabilistic user model and the dialogue system. The second method synthesizes novel dialogue interactions by modeling after a small set of dialogues produced by the developers during the course of system refinement. We evaluated our methodology by speech recognition performance on a set of 520 unseen utterances from naive users interacting with a restaurant domain dialogue system.

1. Introduction

One of the major hurdles facing spoken language system deployment is the problem of acquiring adequate coverage of the possible syntactic and semantic patterns to train both the recognizer and the natural language system. Such training material typically comes in the form of an appropriate set of training utterances (e.g., for the recognizer n-gram language model) representing the distribution of typical queries within the application domain. Such utterances are normally acquired either by an intensive and expensive wizard-of-oz data collection effort, or by otherwise cajoling users to interact with a usually poorly performing early instantiation of the system. Neither of these options is attractive, and therefore we have sought an alternative solution to this problem.

Our goal is to find a way to acquire language model training material in the absence of *any* in-domain real user data. We envision that existing corpora available in one information query application can be transformed into queries appropriate for another application. Our approach extends the pioneering work described in [1], where class-based language models were trained from corpora from other domains with in-domain class expansions. Our strategy involves substantial reconstruction of out-of-domain utterances, followed by extensive filtering to obtain a high-quality subset of the generated material. The subject of how to construct domain-specific sentences from out-of-domain data is beyond the scope of this paper; details can be found in [2]. In this paper, we assume the availability of over 130,000 queries for a restaurants information domain, systematically induced from a large existing corpus in the flight domain. This large set of synthetic data can be used in full as a language model, or further processed to yield language models more closely matching the new domain's usage patterns.

This paper concerns explicitly the process of sampling from this rich over-generated corpus to form training data that would realistically reflect frequency distributions found in a corpus of interaction dialogues using the application. We report here on two distinct techniques for selecting the training corpus, *user simulation* and *dialogue resynthesis*, which can be applied individually or in tandem. Both techniques borrow heavily from example-based translation methods [3-8], viewing the utterance filtering process as an English-to-English translation task.

In the following sections, we first describe the example-based generation method of finding semantically related sentences given a meaning representation. Section 3 explains how this method can be used to sample a corpus of sentence templates that would form the balanced language model for the dialogue application. Subsequent sections describe recognition experiments in a restaurant domain, followed by conclusions.

2. Example-based generation

A key technology in our methodology is the use of example-based generation to find semantically related sentences. This idea is inspired by work done in the field of example-based translation, which typically requires a collection of pre-existing translation pairs and a retrieval mechanism to search the translation memory. Similarity can be based on parse trees [3], complete sentences [4], or words and phrases [5, 6, 7]. We have developed an example-based translation framework which mainly uses semantic information as the similarity measure [8]. The same idea is applied here to "translate" a sentence to a semantically similar sentence, though in the same language.

There are two components in this technology: a collection of indexed sentences, and a retrieval mechanism to search the indexed corpus. They are described in detail in the following.

2.1. Generation of indexed corpus

The example-based generation begins with the construction of an indexed sentence corpus. Each candidate sentence is first parsed [9] to yield a meaning representation called a *seman-*

```
{c clarifier
   :topic {q restaurant
             :restaurant_type "restaurant"
             :pred {p adj_price_range
                      :global 1
                      :topic "cheap" }
             :pred {p pred_cuisine
                      :topic "chinese" } }
   :politeness "please" }
```

price_range:	cheap
cuisine:	chinese
clause:	clarifier

Figure 1: *Semantic frame representation for the sentence "Cheap Chinese restaurants please." and corresponding derived key-value representation.*

```
{c eform
   :price_range "cheap"
   :cuisine "chinese"
   :clause "clarifier"
   :sentences
    ("a cheap chinese restaurant"
     "a cheap restaurant that \
         serves chinese food please"
     "cheap chinese restaurants please"
     "how about a cheap chinese restaurant"
     "yes cheap chinese food"
     ... )}
```

Figure 2: *Example of a group of sentences with the same key-value index.*

tic frame, which encodes the hierarchy of semantic and syntactic structure of the sentence. Then, a set of trivial generation rules [10] are created to extract very lean semantic and syntactic information from the semantic frame as key-value (KV) pairs, which can then be used as an index for that sentence.

We will illustrate this process here with an example. Given a sentence "Cheap Chinese restaurants please." the parser will produce a semantic frame as shown at the top of Figure 1. The language generation system transforms this representation into a simple key-value encoding, as shown in the figure, which is then used to index the original input. To improve efficiency, all sentences with the same set of key-value pairs (ignoring order) are grouped together in the indexed corpus. Figure 2 shows a typical group of such indexed sentences.

2.2. Retrieval mechanism

The basic function of the retrieval mechanism is to find a candidate sentence whose KV-index matches the input KV specification. To allow certain flexibility in matching the key-value pairs, keys are differentiated into several categories, depending on whether they are optional or obligatory, and whether they require matching on the key-only level or the key-value level. These are specified in a header file in the indexed corpus, to allow a developer to flexibly modify the matching strategy. Each obligatory key in the input KV specification has to be accounted for in the matching process, while optional keys in the input can be ignored to avoid a matching failure (but will be preferred otherwise). If more than one group of sentences is retrieved, the selection pool includes all the groups.

We will illustrate the retrieval process with an example to highlight some of the distinctions in the different key types. Assume we want to retrieve from the indexed corpus a sentence similar to "Do you know of any inexpensive french restaurants?" The parsing and generation systems will first produce the following key-value pairs:

price_range:	inexpensive
cuisine:	french
clause:	verify

Suppose the corpus contains only the example shown in Figure 2, with `price_range` and `cuisine` as obligatory keys required to match on the key level, while `clause` is an optional key required to match on the key-value level. If the system is configured to take the values of the retrieved sentence, the output could simply be "cheap chinese restaurants please," or "yes cheap chinese food." If instead, the system is configured to substitute the values in the input KV, those two outputs would be "inexpensive french restaurant please, " and "yes inexpensive french food," respectively. If the `clause` were specified as an obligatory key matching on the key-value level, then the search would fail to generate any output. For an input such as "french restaurants," (`cuisine: french clause: clarifier`), the search would also fail because of the extra obligatory key, `price_range`, in the candidates' KV index.

3. Methodology

In this section, we describe how the example-based generation technology is used to reshape the raw data. Our approach assumes the existence of a large number of synthetic sentences specific to the application domain. The details of how to obtain such sentences can be found in [2]. Figure 3 shows some example patterns of synthesized sentences for the restaurant information domain.

```
1. Ok, how about in <NEIGHBORHOOD>?
2. What is the address of <RESTAURANT_NAME>?
3. When do they open?
4. Are there any <CUISINE> restaurants on
   <STREET> in <NEIGHBORHOOD> in <CITY>?
```

Figure 3: *Examples of synthetic sentence patterns in the Restaurants domain. Classes such as <CITY> and <RESTAURANT_NAME> are instantiated with all legitimate surface values.*

The set of induced synthetic sentence patterns aims to cover all combinations of variations in both syntactic constructs and semantic content. However, it does not necessarily represent an appropriate *distribution* in terms of either syntax or semantics. For example, it is likely that, in real dialogues, sentence 4 in Figure 3 would be much less likely to occur than sentences 1-3. One could, in principle, assign weights to different patterns to reflect their likelihoods; however, it is difficult to establish what the values of the weights should be. The raw data also do not have any support for within-class distributions for contents such as cuisine. A practical solution is to assume a uniform within-class distribution, but this can lead to a high degree of confusion, particularly for large classes (e.g., street names).

We propose two methods to address these issues. The first method is designed for the scenario in which there is no "real" data available for adaptation, which is typically the case before the system has actually been deployed. Our strategy then is to utilize user simulation [11] to filter the raw data, with the goal of achieving a more refined distribution in the semantic content, both on the sentence level and on the class level.

The second method assumes that there is a small amount of development data available, which can be hypothesized to

represent typical user behavior. Such utterances can be used as templates to induce other similar utterances, in order to expand the richness of the development corpus in a systematic way. The resulting data are able to extend the linguistic coverage of the development data, while maintaining a similar dialogue-level and sentence-level semantic content distribution.

3.1. Filtering via user simulation

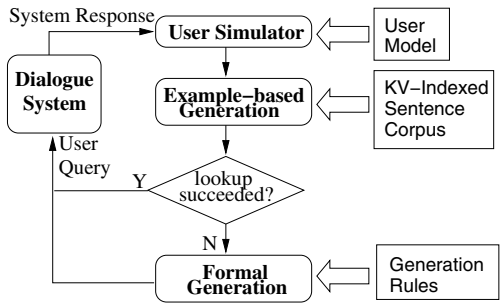

Figure 4: *The process of generating language model training data via user simulation. Note: KV = key-value.*

Figure 4 summarizes the process of filtering the raw data set through dialogue simulation. The raw sentences are first preprocessed into an indexed corpus based on the syntactic and semantic information in each sentence, encoded as KV pairs. A small portion of such a corpus was illustrated in Figure 2. During simulation, given a response from the dialogue system, the user simulator will generate a query, in the form of KV pairs. The KV information is used to retrieve an appropriate template from the indexed corpus, with classes in the template substituted by values specified in the simulator's KV string. The resulting surface string is sent to the dialogue system to push the dialogue interaction forward. In the case of a retrieval failure, perhaps due to gaps in the raw data coverage, a formal generation method [2, 10] can be invoked as a backup mechanism to provide a well-formed query.

A large collection of user queries can be harvested from repeated simulation runs, utilizing a probabilistic model of user behavior. Their semantic content distribution is a result of the complex interactions of different aspects of the user model, as well as the strategies of the dialogue system.

3.2. Dialogue resynthesis

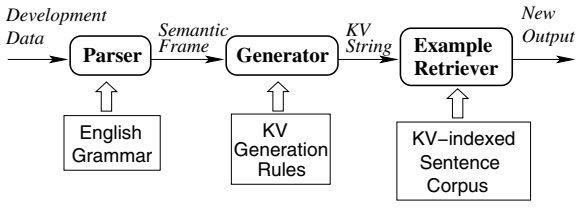

Figure 5: *The process of synthesizing new dialogues by transforming development data. Note: KV = key-value.*

If some set of development data exists, it becomes appealing to consider using it as a guide in sub-selecting from a large corpus of synthetic data. Figure 5 describes the process of transforming such data into new dialogues via example-based generation. The development corpus is parsed utterance by utterance and transformed into a KV representation using the same techniques that were used to create the KV-indexed corpus. During

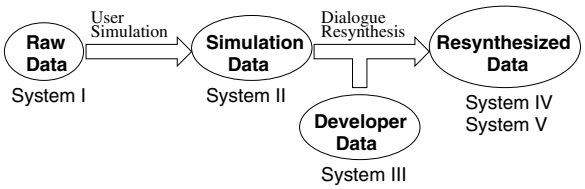

Figure 6: *Illustration of configurations for experimental systems I-V described in the text.*

retrieval, the keys in the retrieved sentence template can either be substituted with values drawn from the development utterance, or left unaltered from their original values in the synthetic corpus. This allows us to experiment with combining probability distributions from different data sources. Specifically, in the first mode, substituting attribute values from the development set into the synthetic will result in a within-class distribution similar to the development data. On the other hand, in the second mode, preserving attribute values of the synthetic data will result in a within-class distribution sampled from the input synthetic data.

4. Experiments

We compared the quality of various sets of language model data via speech recognition performance on a set of test data collected from naive users, who were asked to interact with the restaurant information system via telephone. No specific instructions were provided to the subjects other than a brief introduction to the basic system capability. We excluded recordings which did not contain any speech, but the evaluation data includes utterances with out-of-vocabulary words as well as artifacts such as noise, laughter, etc. The test set consists of 520 utterances collected from 72 phone calls. The data were transcribed manually to provide reference transcripts.

During the course of system development, we have also collected over 3000 sentences from developers interacting with the system, either via a typed interface or in spoken mode. This set of developer/expert data is probably not representative of real data from naive users. Nevertheless, they are of high quality both in terms of the syntactic constructs and the semantic content of the queries. These data can thus serve both as a benchmark against which to reference our synthetic corpus performance and as templates from which to guide a subselection process.

We conducted a number of recognition experiments, as illustrated in Figure 6. These experiments progress through increasingly sophisticated techniques for exploiting the simulation and developer data, generally reflected in improvements in recognition results. Systems I and II correspond to the condition when only synthetic sentences are available for language model training. System I is trained using all the original synthetic data, which totals over 130,000 utterances, and attempts to cover all combinatoric variations in sentence patterns and class values. System II is trained using a set of over 12,500 synthetic utterances obtained by the simulation process described in Figure 4.

System III is a benchmark system based only on the developer data. For systems IV and V, the simulation data are used to generalize utterances drawn from the developer data, in an attempt to broaden its coverage of general language usage, while still maintaining a similar mixture of semantic contents. In other words, we use the developer data as a user model to generate similar but novel dialogues from the synthetic data, following

System	I	II	III	IV	V	Oracle
WER	30.7	22.0	19.1	18.3	17.9	12.2

Table 1: *Word error rates (WER) for different experimental conditions. See Figure 6 and text for details of each system configuration.*

the techniques of Figure 5. The two systems differ only in the way the example sentences are generated: in System IV, the sentence templates are retrieved from the example corpus, but the class values are inherited from the developer data; in System V, the entire sentence is retrieved without modification. Thus, System IV has more-or-less inherited the within-class distribution of the developer data, while System V samples the within-class distribution of the simulation data. Two runs were conducted in each configuration, and the resulting data were combined with the developer data in training the language model.

The recognizer configuration was kept exactly the same for all experiments, except for the language model training data. The recognizer uses class n-gram models, with vocabulary and classes automatically generated from the grammar used for parsing, utilizing techniques described in [12]. In the deployed system, the recognizer utilizes a dynamic class for the restaurant names, which is adjusted based on dialogue context [13]. However, for the off-line experiments conducted here, we configured a static version of the recognizer which uniformly supported all the known restaurant names in our database. The vocabulary size is about 2500 words, with 1100 of these words being unique restaurant names. The acoustic models are trained on about 120,000 utterances previously collected from telephone conversations in the weather and flight information domains.

5. Results and discussion

Our experimental results are summarized in Table 1 in terms of word error rate. We expect that the resulting data from user simulation runs are much more refined than the original data set, both in terms of the semantic content of the sentences (i.e., different types of queries) as well as the probability distribution of the within-class values (e.g., cuisine types, neighborhood names, etc.). This is verified by the experimental results: the word error rate dropped from 30.7% for System I to 22.0% for System II, a 28.3% reduction in error.

System III, which uses only the developer data in language model training, achieved a word error rate of 19.1%, suggesting that the developer data provides a closer model to real user interaction than the approximation modeled by the user simulator.

As indicated in Table 1, both systems IV and V achieve small improvements over the developer-data only system, with larger gains achieved by System V. This seems to suggest that the within-class distribution contributed by the simulation data enhances that of the developer data. We performed a *matched pairs segment word error test* as described in [14] to examine whether the small improvements in word error rate reduction are statistically significant. While we found that the improvement of System IV over the developer-data only system is not statistically significant, the improvement of System V over System III is significant (significance level 0.03).

An "oracle" condition, in which the language model is trained using only transcriptions of test data, yielded a word error rate of 12.2%. This represents a lower bound of word error rate achievable via language model manipulations for this test set. Hence, any further improvement by using more and better language model data sources is likely to be (well) under 5.7% absolute reduction over our best performing non-oracle system.

6. Conclusions

While other work on spoken language modeling has focused on exploiting out-of-domain data via interpolation or adaptation [1, 15, 16], this paper approaches the problem by filtering synthetic domain-dependent data (generated from out-of-domain sources) to achieve a desired frequency distribution of the semantic content, both on the sentence level and on the class level. Our recognition results have shown that a partial match to usage-appropriate semantic content distribution can be achieved via user simulations. Furthermore, limited development data, if available, can be exploited to improve the selection process.

7. Acknowledgements

The research at MIT was supported by an industrial consortium supporting the MIT Oxygen Alliance. The research at CNRI was supported in part by SPAWAR SSC-SD. The content of this paper does not necessarily reflect the position or policy of the Government, and no official endorsement should be inferred.

8. References

[1] L. Galescu, E. Ringger, and J. Allen, "Rapid language model development for new task domains," in *Proc. LREC*, 1998.

[2] G. Chung, S. Seneff, and C. Wang, "Automatic induction of language model data for a spoken dialogue system," submitted to SIGdial Workshop on Discourse and Dialog, Lisbon, Portugal, 2005.

[3] S. Sato, "CTM: an example-based translation aid system using the character-based match retrieval method," in *Proc. COLING*, Nantes, France, 1992.

[4] T. Veale and A. Way, "Gaijin: A template-driven bootstrapping approach to example-based machine translation," in *Proc. NMNLP*, Sofi a, Bulgaria, 1997.

[5] R. D. Brown, "Adding linguistic knowledge to a lexical example-based translation system," in *Proc. TMI*, Chester, England, 1999.

[6] L. Levin, A. Lavie, M. Woszczyna, and A. Waibel, "The Janus III translation system," *Machine Translation*, vol. 15, no. 1-2, 2000, special Issue on Spoken Language Translation.

[7] D. Marcu, "Towards a unifi ed approach to memory- and statistical-based machine translation," in *Proc. ACL*, Toulouse, France, 2001.

[8] C. Wang and S. Seneff, "High-quality speech translation for language learning," in *Proc. of InSTIL*, Venice, Italy, 2004.

[9] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, vol. 18, no. 1, 1992.

[10] L. Baptist and S. Seneff, "Genesis-II: A versatile system for language generation in conversational system applications," in *Proc. ICSLP*, Beijing, China, 2000.

[11] G. Chung, "Developing a flexible spoken dialog system using simulation," in *Proc. ACL*, Barcelona, Spain, 2004.

[12] S. Seneff, C. Wang, and T. J. Hazen, "Automatic induction of n-gram language models from a natural language grammar," in *Proc. Eurospeech*, Geneva, Switzerland, 2003.

[13] G. Chung, S. Seneff, C. Wang, and I. L. Hetherington, "A dynamic vocabulary spoken dialogue interface," in *Proc. ICSLP*, Jeju, Korea, 2004.

[14] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. ICASSP*, Glasgow, Scotland, 1989.

[15] I. Bulyko, M. Ostendorf, and A. Stolcke, "Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures," in *Proc. HLT*, 2003.

[16] M. Bacchiani, B. Roark, and M. Saraclar, "Language model adaptation with map estimation and the perceptron algorithm," in *Proc. HLT*, 2004.