A Case-Based Reasoning Approach for Speech Corpus Generation

Yandong Fan and Elizabeth Kendall

School of Network Computing, Faculty of IT, Monash University, Australia
yandong.fan@infotech.monash.edu.au
Kendall@infotech.monash.edu.au

Abstract. Corpus-based stochastic language models have achieved significant success in speech recognition, but construction of a corpus pertaining to a specific application is a difficult task. This paper introduces a Case-Based Reasoning system to generate natural language corpora. In comparison to traditional natural language generation approaches, this system overcomes the inflexibility of template-based methods while avoiding the linguistic sophistication of rule-based packages. The evaluation of the system indicates our approach is effective in generating users' specifications or queries as 98% of the generated sentences are grammatically correct. The study result also shows that the language model derived from the generated corpus can significantly outperform a general language model or a dictation grammar.

1 Introduction

Stochastic language models have achieved significant success in speech recognition since the last decade [1, 2]. The main underlying technique in stochastic approaches is the use of corpora. The successful utilization of corpora has been proven by many researchers [3,4,5]. However, construction of a corpus pertaining to a specific application is a difficult task, given that there is no pre-knowledge on how users might communicate with the application before the deployment of a system. Research [6] has been conducted to explore the effectiveness of Web-based text corpus generation. Although the proliferation of eText has made the collection of textual material easier than ever, Thompson [7] argues that actually locating eText appropriate to your needs can be quite difficult. Moreover, in the context of conversational systems, the suitability of corpora purely collected from the Internet is controversial due to the difference of written text and spoken language.

Although generally there is a lack of spoken material pertaining to a new application, ample transcriptions do exist in some well-established domains, such as the Air Traffic Information System (ATIS) domain. User modeling has been studied for long and conversations between users and agents of spoken language systems have been recorded and accumulated for decades in these domains. In our project, we seek to develop a speech-enabled mobile commerce application, which we called the MCCS (Mobile Car City system). The system allows a mobile-phone user to specify preferences or initiate queries by speech at the beginning of the conversation. Then all car models conforming to the preferences or queries are retrieved to guide the user in finding specific car models that meet his/her needs. Through carefully examining the spoken transcriptions from the ATIS domain, we believe that user specifications or queries in the MCCS system should share significant similarity in sentence syntactic structure with their counterparts in the ATIS domain. Motivated by this assumption, we believe that the MCCS system can learn a set of sample sentences from the ATIS domain, which can then be used as the knowledge base for case-based reasoning (CBR) to generate a speech corpus pertaining to the MCCS domain.

NLG (Natural Language Generation) research has been dominated by two approaches in the past three decades: template-based and rule-based [8, 9]. Some claim that template-based approaches are not flexible enough while others criticize the sophistication of linguistic grammars implemented in rule-based approaches [8]. A new strand in the arena is learning-based NLG, in which the objective is to learn the mapping from semantics to surface realization through sample sentences. Research [10, 11, 12] suggests that learning-based approaches can balance the inflexibility of template-based methods and the linguistic sophistication of rule-based NLG packages when developing domain-specific generation systems.

In this paper, we explore an incremental learning approach for speech corpus generation. Firstly, a set of sample sentences pertaining to user specifications or queries in the MCCS application are learnt from the ATIS domain. Secondly, a CBR system is built to generate a corpus based on those sample sentences. Finally, an n-gram language model is derived from the corpus by learning the statistical distribution of tokens. The paper is structured as follows. Section 2 introduces the general structure of the corpus generation system. Detailed implementation of the system is described through Section 3-5. Section 6 presents the evaluation results of the generated corpus. Related work is discussed in Section 7. We conclude the study and briefly discuss potential future work in Section 8.

2 System Overview

Our aim is to develop a case-based, domain-specific generation system that can significantly reduce complexity in comparison to rule-based solutions. Case-based reasoning (CBR) systems have long been applied for problem solving in many areas, such as classification, diagnosis, configuration and design, and planning, but only recently has it attracted significant attention from researchers in NLG. Like any other CBR system, a CBR-based NLG system has to include the following components:

- Sample sentence set (Case Base)
- Schema for sentence structure, includes semantic and syntactic (Knowledge Representation)
- Similarity measurement (Acceptance Function)
- Sentence generation (Case Retrieval and adaptation algorithms)

Figure 1 represents the overall structure of our CBR-based corpus generation system. The system implements a pipeline architecture consisting of three stages. Firstly, an initial sample sentence set is created manually to integrate an ATIS sentence base (ASB) and a MCCS phrasal base (MPB). The ASB is a collection of sentences from a well-established corpus in the ATIS domain. The MPB collects phrases in describing car model features, which are abstracted from car manufacturers' websites and marketing brochures. Through careful analysis on the sentences in the ASB and the phrases in the MPB, sample sentences for user queries or specifications pertaining to the MCCS system can be created to form a case sentence base (CSB). The examples (2.1)-(2.3) show an ATIS sentence from the ASB, a MCCS phrase from the MPB, and a sample sentence from the CSB, respectively.

I prefer [**PluralObject** flights] [**ServiceFeature** serve lunch] (2.1)

[**SingularObject** A car] with [**NumOfDoorFeature** four doors] (2.2)

I prefer **cars** with **four doors**. (2.3)

Secondly, these sample sentences in the CSB are annotated to abstract semantic structure and corresponding syntactic structure, which become the case representation for instance learning. Finally, based on the understanding of the characteristics of user queries and specifications, a new input that represents a unique semantic meaning passes through the CBR system. The similarity between the input and a case is calculated and examined. If the distance is within the predefined threshold, adaptation is conducted to generate a new syntactic structure for the semantic input. This procedure is continuously performed until all possible inputs are enumerated. The resultant corpus is then ready for creating an n-gram language model.

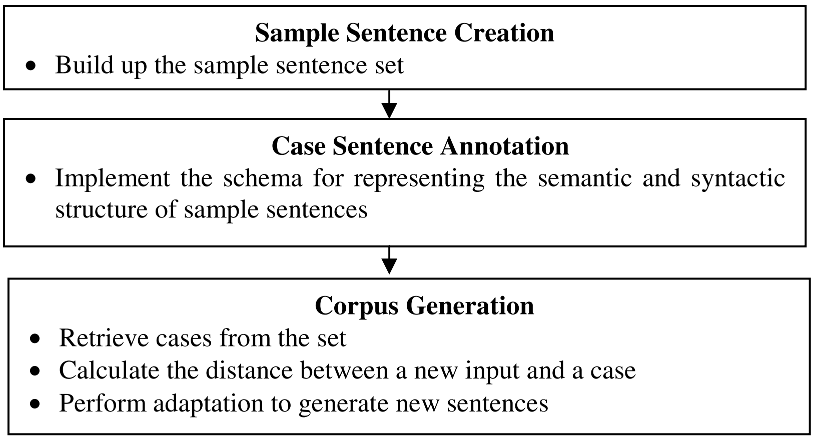
Sample Sentence Creation
• Build up the sample sentence set
Case Sentence Annotation
• Implement the schema for representing the semantic and syntactic structure of sample sentences
Corpus Generation
• Retrieve cases from the set
• Calculate the distance between a new input and a case
• Perform adaptation to generate new sentences

Fig. 1. Main procedures of the CBR-based corpus generation system

3 Sample Sentence Set

We collect utterances pertaining to user preference specifications or queries from the ATIS domain. The utterances are classified into four categories [13] according to their sentence acts. The followings are examples for each category:

Declarative: I prefer a [**TimeFeature** morning] [**SingularObject** flight]. (3.1)

Imperative: Show me the [**PriceFeature** cheapest] [**SingularObject** flight]. (3.2)

Yes-no-question: Can you give me some information for [**Carrier** United]? (3.3)

Wh-question: What [**PluralObject** flights] [**ServeFeature** serve breakfast] and [**HaveFeature** have stops]? (3.4)

Each sentence in the ASB has been annotated by assigning conceptual categories [14] to domain concepts reflecting semantic meanings. Such simple annotations can help us create sample sentences in the MCCS domain by substituting or adjoining operations. The examples (3.5)-(3.8) show the corresponding cases in the CSB rooted from (3.1)-(3.4). There are in total 114 cases in the CSB.

Declarative: I prefer a white car. (3.5)

Imperative: Show me the cheapest sedan. (3.6)

Yes-no-question: Can you give me some information for Honda? (3.7)

Wh-question: What cars can seat more than 5 passengers and have 4 doors? (3.8)

4 Case Sentence Annotation

In our corpus generation system, we implement an annotation scheme for sample sentences in the CSB. Each sample sentence is annotated in two plies. The first ply is the semantic structure representing the domain concept relationships. The syntactic structure is abstracted in the second ply to reflect surface realization.

QueryObject	CAT_Color	CAT_NumOfDoor
CAT_NumOfCylinder	CAT_Make	CAT_BodyStyle
CAT_Transmission	CAT_DriveWheel	
NUM_MadeYearStartValue	NUM_MadeYearEndValue	
NUM_EngineLiterStartValue	NUM_EngineLiterEndValue	
NUM_PriceStartValue	NUM_PriceEndValue	
NUM_NumOfPassengerStartValue	NUM_NumOfPassengerEndValue	

***Note**: CAT means categorical feature while NUM means numeric feature*

Fig. 2. Conceptual category set

We utilize a set of conceptual categories to abstract conceptual meanings in our application domain (Figure 2). The benefit of introducing conceptual categories is that each concept in a case sentence can be instantiated with different values to satisfy word coverage. The semantic ply indicates the sentence act and the number type (singular or plural) of the query object. It also catches the relations between those conceptual categories involved in the annotated sentence. For instance, the semantic ply of Example (3.8) can be described in a Query Language [15] as:

$$\{x|x.\text{Act='wh-question'} \wedge x.\text{QueryObject='car'} \wedge x.\text{ObjectType='Plural'} \wedge car.\text{NumOfDoor='four'} \wedge car.\text{NumOfPassenger.StartValue=5}\} \tag{4.1}$$

The syntactic ply analyzes the syntactic structure of the sentence, which is the formalism for surface realization. The structure of the formalism is adapted from the systemic functional grammar [16], which consists of three layers: clause, phrase and token. Figure 3 represents the syntactic structure of the example (3.8).

```
<SynStructure type=″ComplexClause″>
 <SynClause localID=″c1″>
  <SynPhrase type=″Simple″ fc=″CannedText″ value=″What″ />
  <SynPhrase type=″Simple″ fc=″ObjectThing″ ref=″Head″ value=″VALUE″ />
  <SynPhrase type=″Simple″ fc=″Predicate″ value=″can seat″ />
  <SynPhrase type=″Complex″ fc=″NUMFeature″ ref=″NumOfPassenger″>
    <SynToken type=″Simple″ fc=″CannedText″ value=″more than″ />
    <SynToken type=″Simple″ fc=″Feature″ value=″StartValue″ />
    <SynToken type=″Simple″ fc=″Quantifier″ value=″passengers″ />
  </SynPhrase>
  <SynPhrase type=″Simple″ fc=″PredicateConj″ value=″and″ />
  <SynPhrase type=″Simple″ fc=″Predicate″ value=″have″ />
  <SynPhrase type=″Complex″ fc=″CATFeature″ ref=″NumOfDoor″ />
    <SynToken type=″Simple″ fc=″Feature″ value=″VALUE″ />
    <SynToken type=″Simple″ fc=″Quantifier″ value=″doors″ />
  </SynPhrase>
 </SynClause>
</SynStructure>
```

Fig. 3. The syntactic structure of Example (3.8) in XML

5 Corpus Generation

The general principles for creating a corpus are semantic coverage, syntactic coverage, prosodic coverage and word coverage [11]. As the target outcome of our system is a sentence corpus for language modeling, prosodic coverage is not our focus.

- *Semantic coverage*: the corpus should cover domain concepts and relationships as completely as possible;
- *Syntactic Coverage*: the corpus should reflect many rich syntactic variations, as found in natural language;
- *Word Coverage*: the corpus should cover as many words as possible in the vocabulary.

In this Section, we demonstrate how these three principles have been considered and satisfied during the corpus generation. Although case sentences marked with domain-specific conceptual categories can be used directly for surface natural language generation, as was suggested in [17], it is only capable of handling certain circumstances, such as simple applications (as the NLG1 in [17]) or under the assumption that the corpus is large enough for statistical analysis (as the NLG2 and NLG3 in [17]). In our project, we seek to create a corpus based on a sample sentence set with a limited size. Therefore, a CBR approach with adaptability is more appropriate [11]. An input to the sentence generator is a semantic structure represented in an AVM (attribute value matrix) form, including a sentence act, the query object and its features. Figure 4 shows a typical example of inputs.

$$\left(\begin{array}{ll}\text{Act:} & \text{declarative} \\ \text{Object:} & \begin{bmatrix}\text{head: car} \\ \text{number: singular}\end{bmatrix} \\ \text{Feature:} & \left(\begin{array}{l}\text{color: red} \\ \text{num\_of\_door: 4} \\ \text{body\_style: sedan}\end{array}\right)\end{array}\right)$$

Fig. 4. An example input showing the semantic meaning of a user's specification: "I would prefer a red sedan with four doors"

In order to satisfy the semantic coverage, we explore all possible combinations of features related to the car object. We made a decision to include only those combinations with less than 5 features so that the generated sentence can be kept in a reasonable length. In terms of syntactic coverage, we consider all sentence acts to express a feature combination. In addition, the variations of syntax in describing a numeric feature are explored. The examples (5.1)-(5.4) show four types of phrases with different foci to specify the price of a car. The word coverage is achieved through enumerating all possible values of each feature.

a price no less than [**PriceStartValue** 15,000] dollars (5.1)

a price no more than [**PriceEndValue** 30,000] dollars (5.2)

a price from [**PriceStartValue** 15,000] to [**PriceEndValue** 30,000] dollars (5.3)

a price around [**PriceStartValue=PriceEndValue** 20,000] dollars (5.4)

The generation of sentences is performed according to a procedural algorithm. The algorithm consists of four procedures:

- *Distance measuring*: An input is compared with the semantic representation of an instance in the CSB. Candidates above the threshold are selected for further processing.
- *Feature categorizing*: Through examining the difference between the feature set of the input and that of the case, features are categorized into four groups: *OldFeatures*, *AdjustFeatures*, *OtioseFeatures* and *NewFeatures*. Old features are those shared by the input and the case. Adjust features are those numeric features belonging to both but with different focuses. Features in the case but not in the input are called otiose features, and new features are those that appeared in the input but not in the case.

Sentence Generation Algorithm

Input: The semantic structure of a new sentence: I ;
An instance of cases retrieved from the sentence base: C;
Output: a generated sentence if it's successful; otherwise null.

```
1. If (I.act == C.act) then proceed to 2;
2. If ( isCompatible (I.object, C.object)) then proceed to 3;
3. Calculate Dis (I, C);
```

Denote I (F) as the input, where $F=\{F_1,F_2,..F_n\}$ represents the feature set of the input.
Denote C (A) as the case, where $A=\{A_1,A_2,\ldots A_m\}$ represents the feature set of the case.
Denote k as the number of features $\in F\cap A$.
If we define:

$$d(f_i, C) = \begin{cases} \alpha & \text{if } f_i \in C \\ \beta & \text{if } f_i \notin C \end{cases}$$

$$d(a_i, F) = \begin{cases} \alpha & \text{if } a_i \in F \\ \gamma & \text{if } a_i \notin F \end{cases}$$

then $Dis(I, C) = \Sigma\, d(f_i, C) + \Sigma\, d(a_i, F) = 2\alpha k + (n-k)\beta + (m-k)\gamma$

The distance function (metric) should satisfy the following conditions:

(1) $0 \le Dis(I,C) \le 1$;
(2) if $n=m=k$ then $Dis(I,C)=0$;
(3) if $k=0$ then $Dis(I, C)=1$;
(4) $\beta > \gamma > 0$, given that insertion is a more difficult operation than deletion;

Thus $\alpha=0$;
$1/(m+n) < \beta < 1/n$;
and $\gamma = (1-n\beta)/m$.

Choose $\beta = 1/(n+1)$ then $Dis(I, C) = (n-k)*(1/(n+1)) + (m-k)*(1/(n+1))/m$

```
4. If (Dis(I,C) <0.5) then proceed to 5;
5. For each feature $f_i \in$ {oldFeatures||adjustFeatures}=$F\cap A$, perform value substitution
   or adjustment operation;
   For each feature $f_j \in$ {otioseFeatures}=$(F\cup A)-F$, perform deletion operation;
   For each feature $f_k \in$ {newFeatures}=$(F\cup A)-A$ , perform insertion operation.
6. Surface realize the sentence according to the adaptive syntactic structure.
```

Fig. 5. Sentence generation algorithm

- *Case adapting*: For each type of features, different adaptations to the case are performed to generate a syntactic structure corresponding to the semantic structure of the input.
- *Surface realizing*: A sentence is generated according to the adapted syntactic structure.

Figure 5 depicts the details of the sentence generation algorithm.

6 Evaluation

The evaluation is done at two levels. Firstly, the generated sentences are scored by human evaluators using three ratings: no grammatical error, minor grammatical error and major grammatical error. The ratios generally represent the quality of sentence generation. Secondly, the generated corpus is divided into two sets: a training set and a test set. We use the training set to build an n-gram language model. The language model is applied to a speech recognition engine to test recognition effectiveness. The sentences from the test set are used for this testing. We measure the word error rate to verify the acceptability of the language model.

Table 1. Testing results of language models

Testing Engine	**Language Model**	**I**	**O**	**S**	**N_{sol}**	**Percent correct**	**Accuracy**
Sphinx 4, No speaker training	Our Domain Specific Model	84	16	154	1024	83.4%	75.2%
Sphinx 4, No speaker training	WSJ5K Model (Vocabulary size: 5,000)	82	10	568	1024	43.6%	34.6%
Sphinx 4, No speaker training	HUB4 Model (Vocabulary size: 64,000)	56	28	704	1024	28.5%	23.1%
Dragon Naturally Speaking Preferred (version 3.52), Speaker training	Dictation Grammar	99	26	327	1024	65.5%	55.9%

I: Number of *inserted* symbols **O: Number of *omitted* symbols**
S: Number of *substituted* symbols **N_{sol} : Total Number of symbols for testing**

Two hundred sentences are randomly selected from the generated corpus for grammatical evaluation. Of these sentences, 196 sentences are grammatically correct. Three sentences have major grammatical errors and one has minor error. The effectiveness of the system in generating user specifications and queries is supported by the high percentage of correctness. We then follow the methods introduced in [1] to test the performance of the language model derived from the corpus. We utilize the Sphinx 4 Recognition Engine [18] without speaker training to test our language model and two general models. A further test is conducted to compare our model with the dictation grammar used in the Dragon Naturally Speaking Preferred (version 3.52) Engine with speaker training. Table 1 details the test results, which suggest the language model specific to the MCCS can significantly outperform a general language model or a dictation grammar.

7 Related Work

Generating natural language through learning is a relatively new endeavor. Trainable methods for surface NLG are introduced in [17] to learn the mapping between semantic meaning and syntactic structure so that sophisticated grammars can be avoided. The implicative assumption of trainable systems is the existence of a large corpus. In our project, we can only create a sample sentence set of a limited size, which is not appropriate for training. Our CBR approach differs from trainable methods in that instances in the case base are used for adaptation to generate new sentences directly, instead of for calculating statistical distribution. [10, 12] introduce an approach for instance-based natural language generation. However, instead of adapting instances to generate sentences, instances are just used to compare with sentences generated by a rule-based system for choosing the final output. No adaptation is performed during the generation procedure. [11] presents a surface natural language generator in the real estate domain that employs a case-based paradigm. Its adaptation-guided retrieval makes it ultimately similar to our system. However, our approach differs from it in two respects. Firstly, we employ a quantitative distance measurement for acceptance function. Compared with the qualitative cost-analysis method used in [11], our method provides a numeric value for similarity comparison, which we believe is more straightforward. Secondly, the syntactic structure of cases in our system is represented in systemic functional formalism while graphical tree structure is utilized in [11] to represent the syntactic, lexical, prosodic and acoustic realizations. Our method is simpler and less prone to grammatical error in generating structured sentences.

8 Conclusions

This paper presents a CBR system to generate a speech corpus for the MCCS application. In comparison to traditional NLG approaches, this system overcomes the inflexibility of template-based methods while avoiding the linguistic sophistication of rule-based packages. Our research indicates that CBR learning techniques can perform effectively in generating structured sentences. This approach is particularly useful if the size of the sample sentence set is relatively small. The study results also suggest that a language model pertaining to a specific application is a necessity as general models or dictation grammar cannot satisfy the requirements for recognition accuracy.

This study is part of research to incorporate natural language understanding capacity into a framework to develop speech-enabled mobile commerce applications. We only explore natural language models for understanding user specifications or queries at the beginning of a conversation in the context of mobile commerce. After that, users would be guided by a system-directed dialogue to continue their search for desired products. When a user shows interest in a particular product and selects to listen to the detailed description of the product, the system will play a pre-recorded audio file. We believe speech for product description can be generated through CBR-based NLG system in a similar manner. A NLG method can provide much more flexibility in generating product descriptions in comparison to pre-recorded audio files. In future work, the CBR approach introduced in this paper should be able to be extended for product description generation.

Acknowledgements

The authors would like to thank Benny Nasution and Adrian Ryan for examining the sample sentences, grammatically evaluating the generated sentences and testing the performance of different language models.

References

1. Becchetti, C. and Ricotti, L.P. (1999): Speech Recognition: Theory and C++ Implementation, John Wiley & Sons.
2. Somers, H. (2000): Empirical Approaches to Natural Language Processing, in Handbook of Natural Language Processing (Eds., Dale, R. et al.), pp.377-384. New York, Marcel Dekker.
3. Jurafsky, D. et al. (1994): The Berkeley Restaurant Project. In Proceedings of ICSLP-94, Yokohama, Japan, pp.2139-2142.
4. Lesher, G.W. et al. (1999): Effects of ngram order and training text size on word prediction, In Proc. of the RESNA'99 Annual Conference, Arlington, VA. pp.52-54.
5. Rudnicky, A.I. et al. (2000): Task and Domain Specific Modeling in the Carnegie Mellon Communicator System, in ICSLP2000, Beijing, China.
6. Lesher, G.W. and Sanelli, C. (2000): A Web-Based System for Autonomous Text Corpus Generation, In Proceedings of ISSAAC 2000, Washington DC, U.S.A.
7. Thompson, H.S. (2000): Corpus Creation for Data-Intensive Linguistics. In Handbook of Natural Language Processing (Eds, Dale R. et al.), pp.385-401. New York, Marcel Dekker.
8. Reiter, E. (1995): NLG vs. Templates, In Proceedings of the 5th European Workshop on Natural Language Generation, Leiden, the Netherlands.
9. Oh, A.H. and Rudnicky, A. (2000): Stochastic Language Generation for Spoken Dialogue Systems, In Proceedings of the ANLP/NAACL Workshop on Conversational Systems, May 2000, pp.27-32.
10. Varges, S. and Mellish, C. (2001): Instance-based Natural Language Generation, In Proceedings of the 2nd Meeting of the North America Chapter of the Association for Computational Linguistics (NAACL-2001), Pittsburgh, PA, June 2001.
11. Pan, S. and Weng, W. (2002): Designing a speech corpus for instance-based spoken language generation. In Proceedings of INLG2002, New York, U.S.A.

12. Varges, S. (2003): Instance-based Natural Language Generation, PhD thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh.
13. Jurafsky, D. and Martin, J.H. (2000): Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice Hall.pp.332-334.
14. Sun, J. et al. (2000): A Robust Speech Understanding System Using Conceptual Relational Grammar, In Proceedings of ICSLP'2000, Oct 2000, Beijing, China.
15. Minock, M.J. (2003): A Phrasal Generator for Describing Relational Database Queries, In Proceedings of the 9th European Association of Computational Linguistics workshop on Natural Language Generation, Apr 2003, Budapest, Hungary.
16. Halliday, M.A.K. and Matthiessen, M.I.M. (2004) An Introduction to Functional Grammar, 3rd Edition, ARNOLD.
17. Ratnaparkhi, A. (2000): Trainable Methods for Surface Natural Language Generation, In proceedings of the ANLP/NAACL'00, Seattle, WA. pp.194-201.
18. The CMU Sphinx Group Open Source Speech Recognition Engines. Retrieved Dec 12, 2004. From http://cmusphinx.sourceforge.net/html/cmusphinx.php