# AUGMENTING VARIATION OF SYSTEM UTTERANCES USING CORPORA IN SPOKEN DIALOGUE SYSTEMS

*Ryuichiro Higashinaka*[1,3]*, Rashmi Prasad*[2]*, and Marilyn Walker*[3]

[1] NTT Communication Science Laboratories, NTT Corporation, Japan
[2] Institute for Research in Cognitive Science, University of Pennsylvania, USA
[3] Department of Computer Science, University of Sheffield, UK
*rh@cslab.kecl.ntt.co.jp, rjprasad@linc.cis.upenn.edu, walker@dcs.shef.ac.uk*

## ABSTRACT

Compared to the variation in utterances that users may exhibit in conversation with spoken dialogue systems, system utterances can be very rigid with little variation. One recent approach to dealing with this problem is a trainable sentence planner, which uses natural language generation techniques to create a large number of alternative utterances for a given content, by randomly combining an initial set of basic syntactic structures. However, the amount of variation achieved is limited by the size of the initial set, which is usually specified by hand. We propose augmenting the variation of system utterances by *automatically* incorporating useful sentences obtained from corpora into the initial set used by the generation process. Experimental results show that this approach can successfully create a generator with augmented variation.

## 1. INTRODUCTION

Spoken dialogue systems can now perform various tasks with reasonable task completion rates [1, 2, 3] and have been increasingly developed and deployed in recent years. Since these systems have to be able to handle unconstrained speech to understand users' requests and carry out certain tasks, much work has been devoted to improving the ability of systems to understand speech. As a result, users can express their requests in multiple ways, using a variety of syntactic structures and different sets of vocabulary [4, 5].

On the other hand, compared to the variation in utterances that users may exhibit in conversation with systems, system utterances can be very rigid with little variation. Many practical systems still make use of a limited number of templates or rules for utterance generation [6, 7]. It is therefore unusual for a system to use different forms to deliver the same content. This rigidity persists partly because it is sometimes not desirable to convey the same content in different ways for fear of confusing the users, but mainly because it is costly to prepare different generation templates or rules for the same content. Since humans rarely use the same expression for the same content, we believe it is desirable for systems to exhibit variation in their utterances so that they can communicate with humans more *naturally*.

Recently, the idea of a *trainable sentence planner* has been introduced [8]. A trainable sentence planner generates a large number of *alternative utterances* for a given content and ranks them according to the user's preference so that it can deliver the content in a way the user prefers. This technique is promising for creating variation in system utterances, since, by changing the ranking parameters, the system can produce different utterances for the same content instead of the same utterances again and again. However, since the generation process works by randomly combining an initial set of basic syntactic structures, which are specified by hand, the resulting alternative utterances are sometimes not very different from one another, leaving an open problem as to how to achieve system utterance variation that approaches the variation exhibited by system users.

This paper proposes a technique for increasing the variation of alternative system utterances, by extracting useful sentences from corpora, and converting them into the basic syntactic structures used as the initial set by the generation process of a trainable sentence planner. We focus on restaurant recommendation utterances in MATCH, a multi-modal dialogue system providing entertainment information for New York. Section 2 briefly describes **SPaRKy** (Sentence Planning with Rhetorical Knowledge), the trainable sentence planner used in MATCH [9, 10]. Section 3 describes the problem in detail, and Section 4 mentions related work. Section 5 explains our method for extracting useful sentences from corpora and how the acquired sentences are incorporated into the generation process. Section 6 describes the experiment we performed to verify our approach. The last section summarizes and mentions future work.

## 2. SENTENCE GENERATION IN A TRAINABLE SENTENCE PLANNER

The generation process in a trainable sentence planner aims to generate a large number of alternative utterances for a given content. In SPaRKy, the content is represented by a text plan [9], which is a set of propositions and the rhetorical relations among them. Figure 1 shows a text plan for a recommendation for an Italian restaurant in MATCH. In this example, the proposition that Babbo is the best restaurant (p1) is justified by three propositions (p2-p4). Each proposition is associated with one or more *basic syntactic structures*, that can be realized by the RealPro realizer [11], as a short sentence that asserts that proposition. These basic syntactic structures are linguistic representations in the form of RealPro's deep syntactic structures (DSyntSs), which represent semantic roles and dependency structures. These linguistic representations support the generation of a large number of alternatives by the application of clause-combining operations, which we describe later. (See [11] for details of DSyntS.) Figure 2 shows some potential realizations of p1-p4.

SPaRKy takes the text plan and converts it into a tree called a text plan tree. The text plan tree encodes the ordering of proposi-

| Relations |
|---|
| justify(nuc:p1, sat:p2), justify(nuc:p1, sat:p3), justify(nuc:p1, sat:p4) |

| Propositions |
|---|
| p1. assert-best(Babbo) |
| p2. assert-food_quality(Babbo, superb) |
| p3. assert-decor(Babbo, excellent) |
| p4. assert-service(Babbo, excellent) |

**Fig. 1**. Text plan for a recommendation for an Italian restaurant in MATCH, where p1-p4 are the IDs assigned to the propositions and nuc and sat stand for nucleus and satellite.

| | |
|---|---|
| p1 | Babbo has the best overall quality among the selected restaurants. |
| p2 | Babbo has superb food quality. |
| p3 | Babbo has excellent decor. |
| p4 | Babbo has excellent service |

**Fig. 2**. Possible realizations of propositions p1-p4.

tions and relations among propositions and sets of propositions. The conversion is performed by text plan transformation rules. These rules randomize the order of the propositions and assign a hierarchical structure to the propositions while guaranteeing the validity of the resulting text plan tree in terms of (a) principles of rhetorical structure and (b) principles of entity-based coherence. A text plan tree for the text plan in Fig.1 is in Fig.3 (step 0). The tree is composed of relation nodes (justify, infer) and proposition nodes (p1-p4).

The text plan tree then undergoes the aggregation process to generate an alternative utterance. Figure 3 shows how the aggregation proceeds. The process first finds the left-most bottom-most relation node that has two or more propositions as its children, and randomly selects a clause-combining operation to apply to their associated DSyntS two at a time. The operations include merging two DSyntS with conjunctions, forming relative clauses, or joining two DSyntS by a period. (See [9] for details.) When the period clause-combining operation is applied, the alternative utterance contains more than one sentence. When two nodes are combined, a new node called an *aggregation node* replaces them and appends the two nodes to itself (Step 1). The aggregation node specifies the operation that was applied to the two nodes as well as the resulting aggregated DSyntS. We collectively call propositions and aggregation nodes *aggregatable nodes* because they both have corresponding DSyntS and therefore can be aggregated by clause-combining operations. If there are still aggregatable nodes under the relation node, they are aggregated in the same fashion until there remains only one aggregation node under the relation node (Step 2). In this case, the relation node is replaced by the aggregation node (Step 3). The aggregation continues by processing the next relation node that has two or more aggregatable nodes (Step 4). The process ends when there is only one aggregation node at the root of the text plan tree (Step 5). Each iteration of the above process creates one alternative utterance. With N iterations, N alternative utterances are created for a given text plan.

## 3. PROBLEM

In SPaRKy, variation in the alternative utterances is created in two stages: the random ordering of propositions in the text plan transformation, and the random application of clause-combining operations. Although the two stages can create a large number of different alternative utterances for a given text plan, the generated alternatives are sometimes very similar.

| | |
|---|---|
| alt1 | Babbo has the best overall quality among the selected restaurants since it has excellent service with excellent decor and it has superb food quality. |
| alt2 | Babbo has excellent decor and it has superb food quality with excellent service. It has the best overall quality among the selected restaurants. |
| alt3 | Babbo has the best overall quality among the selected restaurants with excellent decor, excellent service and superb food quality. |
| alt4 | Since Babbo has superb food quality and excellent service with excellent decor, it has the best overall quality among the selected restaurants. |
| alt5 | Babbo has the best overall quality among the selected restaurants because it has excellent service with superb food quality and excellent decor. |

**Fig. 4**. Examples of alternative utterances for a text plan.

Figure 4 shows five of the alternative utterances for the text plan in Fig.1. It is clearly noticeable that the same words appear repeatedly, with the variation arising primarily from changes in ordering, or from the cue words or conjunctions that are used. This is because there is only a limited number of basic syntactic structures in the initial set of DSyntS associated with each proposition. The specification of the mapping from a proposition to a syntactic structure is done by hand, with the result that each proposition may have only one basic syntactic structure associated with it. For example, the proposition that restaurant X has good food is expressed only by "X has good food," and restaurant X has good service by "X has good service." While many randomized operations may be applied to these two basic syntactic structures during the aggregation process, it is inevitable that the resulting alternative utterances are limited in variation by the fact that they are composed from only these basic syntactic structures.

Since preparing the basic syntactic structures by hand is very costly, requiring potentially a large effort of corpus study, and linguistic expertise, to specify structures that have the same meaning as the propositions, we need a systematic method of finding sentences that can be safely associated with the propositions. Our idea is to find a method for *automatically* augmenting the number of basic syntactic structures associated with the propositions, by data mining of corpora, and thereby generate alternative utterances with more variation.

## 4. RELATED WORK

Automatically finding sentences bearing the same meaning has been extensively studied in the field of automatic paraphrasing. Studies have made use of parallel corpora and corpora of sentences describing the same events to collect a set of sentences of similar meanings so that paraphrasing patterns can be derived by multiple sequence alignment (MSA) techniques [12, 13]. Other work has proposed finding predicates of similar meanings from corpora by using the similarity of contexts around the predicates [14].

Although much work has been done, the reported techniques cannot be directly applied for our purpose because they focus only on finding a set of sentences of the same meaning and not on as-

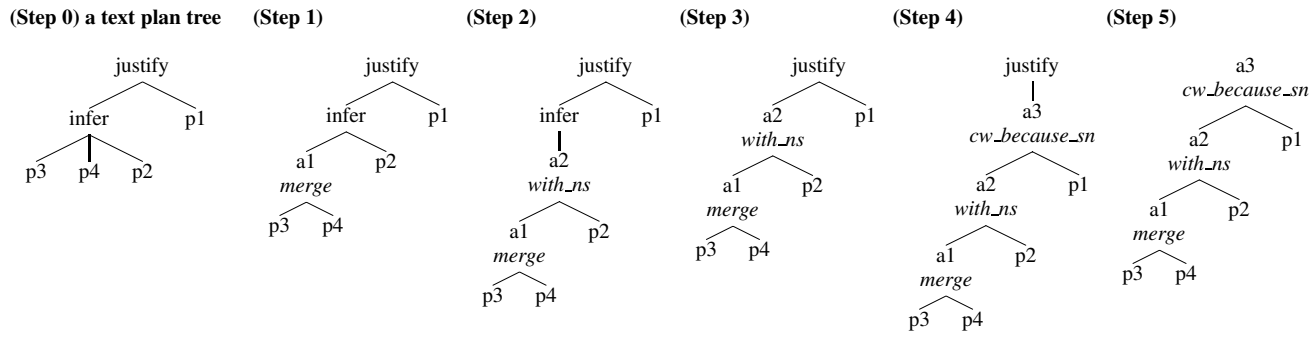


**Fig. 3**. Example of an aggregation process. Justify and infer are relation nodes and a1-a3 are aggregation nodes. Merge, with_ns, and cw_because_ns are the randomly selected clause-combining operations associated with the aggregation nodes. The aggregated sentences for a1-a3 are "Babbo has excellent decor and excellent service," "Babbo has excellent decor and excellent service with superb food quality," and "Because Babbo has excellent decor and excellent service with superb food quality, it has the best overall quality among the selected restaurants," respectively. Here, the aggregated sentence of a3 becomes the alternative utterance.

sociating a specific meaning with the sentences. In other words, previous work assumes some underlying meaning behind the sentences but does not deal with it directly. Although there have been some attempts to use logical forms to represent propositions of the sentences [15, 16], the approach is only used as a way to absorb some surface-level differences of sentences to find accurate answers for question answering systems.

## 5. PROPOSED METHOD

We propose automatically acquiring basic syntactic structures from corpora that can be associated with propositions, and incorporating them into SPaRKy. In this paper, we focus specifically on the restaurant recommendations in MATCH. Since propositions in recommendations concern the features of a restaurant, such as food quality, service, decor, and location, we first need a corpus that includes sentences expressing such features. We create such a corpus by collecting restaurant user restaurant reviews from the web.

Then, to extract only the sentences that can be associated with the propositions from the corpus, we have to determine what the sentences are about. We use the rating information attached to the reviews and some restaurant-related keywords to pinpoint the meaning of the sentences. The rating information is generally coupled with reviews and provides an explicit polarity (e.g., good or bad) for the sentences. We create a *meaning representation* consisting of rating information and keywords for each sentence. We use this meaning representation and several word-level features of a sentence to decide whether to extract the sentence. We extract only those that are strongly considered to be associated with the propositions. When incorporating the extracted sentences into SPaRKy, we need to perform a match between the extracted sentences and the propositions to see if any of the extracted sentences can actually be associated with the propositions. We perform this matching via the meaning representation.

In what follows, we describe the step-by-step procedure used to extract the sentences. The procedure includes collecting user reviews on the web, creating meaning representations for the sentences, screening inappropriate sentences, and converting the sentences into DSyntS, which is the sentence representation used in SPaRKy. Then, we describe how we perform the matching between the propositions and the extracted sentences. Lastly, we describe how to incorporate the automatically constructed DSyntS for the extracted sentences into SPaRKy.

| Ratings |
|---|
| Food=5, Service=5, Atmosphere=5, Price/Value=5, Overall=5 |
| **User review comment** |
| The best Spanish food in New York. I am from Spain and I had my 28th birthday there and we all had a great time. Salud! |

**Fig. 5**. Example of a user review comment with associated ratings

### 5.1. Creating the corpus

We created a corpus of restaurant reviews by scraping user review comments posted at we8there.com (http://www.we8there.com/). Although there are other websites dealing with reviews such as london-eating (http://www.london-eating.co.uk/), we selected this particular one because it includes a 1-to-5 Likert-scale rating of food, service, atmosphere, price/value, and overall rating for each individual user review. Although other information is available, such as whether the restaurant accepts reservations, we did not use this information because it is not related to the recommendation utterances in MATCH. Figure 5 shows an example of review comments with associated ratings. We collected 3004 comments on 1810 restaurants with associated ratings. The corpus contains a total of 18811 sentences.

### 5.2. Determining the meaning of user review sentences

To determine the meaning of the user review sentences, we use the ratings and several keywords associated with them. Each rating has a rating-key and a score. For example, for *food=5*, food is the rating-key and 5 is the score. We define by hand a set of keywords that are considered to be strongly associated with a rating-key (Fig.6). For example, *food* and *meal* are keywords for the rating-key *food*. Note that the keywords need not be complete, because it is not necessary for our purposes to extract every possible sentence from the corpus.

Having defined the keywords, we hypothesize that a sentence containing keywords of a rating-key concerns that rating-key. For example, a sentence containing *food* or *meal* is assumed to be about food. We also hypothesize that the score is reflected in the

| rating-key | keywords |
|---|---|
| Food | food, meal |
| Service | service, staff, waitstaff, wait staff, server, waiter, waitress |
| Atmosphere | atmosphere, decor, ambiance, decoration |
| Price/Value | value, price, overprice, pricey, expensive, inexpensive, cheap, affordable, afford |
| Overall | recommend, place, experience, establishment |

**Fig. 6**. Keywords for rating-keys.

```
token=The root=the pos=DT ne=NULL=NULL
token=best root=best pos=JJS ne=NULL=NULL
token=Spanish root=spanish pos=JJ ne=foodtype=Spanish
token=food root=food pos=NN ne=food=food
token=in root=in pos=IN ne=NULL=NULL
token=New root=new pos=NNP ne=location=New York
token=York root=york pos=NNP ne=location=New York
token=. root=. pos=. ne=NULL=NULL
```

**Fig. 7**. The result of keyword detection and POS-tagging.

sentence. Based on these two hypotheses, we assume that if a sentence has a keyword of a rating-key K and K has a score S, then that sentence has the meaning K=S. For "The best Spanish food in New York" (Fig.5), since there is a keyword *food* and the rating for this sentence has *food=5*, this sentence is assumed to have the meaning *food=5*.

Other than the rating-keys, features of a restaurant also include location, food types, food subtypes (specific food served), and so forth. To see if the sentences contain such information, we again exploit keywords. We created a list of keywords for locations, country names, food types, and food subtypes (these can be extracted from a database for the application or from the vocabulary of the speech recognizer). For example, if the sentence contains a location keyword, we assume that the sentence concerns the location of a restaurant. For "The best Spanish food in New York," there is one food type and one location. Therefore, we assume that the sentence is about the food type and the location of a restaurant. Since this sentence also has *food*, the eventual meaning of this sentence will be food=5, food type, and location, which is described by our *meaning representation*: {food=5, food type, location}.

Using the ratings and the keywords, we created meaning representations for all 18811 sentences in the corpus. For keyword detection, we employed GATE [17], which is a suite of linguistic processors including a tokenizer, a sentence splitter, a part-of-speech (POS) tagger, and a named entity recognizer. We modified the dictionary of the named entity recognizer to detect our keywords as well as those built into GATE such as person names. Figure 7 shows the result of keyword detection for "The best Spanish food in New York." Notice that Spanish, food, and New York are successfully detected as keywords of food type, food, and location, respectively. The result also contains POS tags and base forms for each word.

### 5.3. Screening

The screening process filters out sentences considered to be inappropriate to be associated with the propositions of recommendations in MATCH. The screening is performed based on the six rules below. These rules check the validity of the sentences at different levels and are not independent. They are checked against each sentence one by one, and if one of the rules is found to be applicable, the sentence is discarded.

- If a sentence does not have a keyword of restaurant features, discard the sentence because it cannot be used for the propositions concerning restaurants.
- If a sentence contains unknown words, which include typographical errors, discard the sentence because the meaning of such a sentence is uninterpretable. Unknown words are defined as those not included in the entries of WordNet.
- If a sentence has more than 20 words, discard the sentence because such a sentence is likely to have a complex meaning not covered by the propositions.
- If a sentence has keywords of food subtypes, person names, country names, or named entities of dates (e.g., today, tomorrow, Aug. 26th) and prices (e.g., 12 dollars), discard the sentence because such keywords do not relate to the propositions in MATCH. Although prices do appear in the propositions, they mainly describe the average prices, and prices in our corpus rarely mean average prices.
- If the POS tags for a sentence contain NN (Noun), PRP (Personal pronoun), CD (Numeral), discard the sentence because (1) a noun that is not a keyword may have a meaning not in the propositions, (2) it is not appropriate for the system to use pronouns such as *I* and *we*, and (3) numbers (other than prices) do not appear in the propositions.
- If the POS tags for a sentence have LS (List item marker) or JJR (Comparative adjective), or if the sentence contains *contextual words*, discard the sentence because they all suggest that the meaning of the sentence is dependent on the surrounding sentences, so that it cannot be used independently for a proposition. The contextual words we defined are *however, also, too, then, after, though, now, than, either, neither*. Additionally, *and, because, but, for, however, since, until, as, although, while, if, when, then, so, both, even again, next, other, yes, no, that, needless to say* are regarded as contextual words if they appear at the beginning of a sentence.

The rules may seem very strict. However, for the extracted sentences to be safely associated with the propositions, we have to carefully select only those that could be strongly considered to be useful. Out of 18811 sentences, a total of 735 sentences survived the screening process. This is approximately 3.9% of all sentences.

### 5.4. DSyntS conversion

To be incorporated into SPaRKy, the extracted sentences have to be converted to DSyntS, so we developed a DSyntS converter tool. Since DSyntS have a form similar to dependency structure, we first process the sentences with Minipar [18], a general-purpose dependency parser. A sentence augmented with the dependency structure is then processed by the DSyntS converter. Since we are dealing with sentences of user reviews which are different from the newspaper articles on which Minipar was trained, the output of Minipar for such sentences can be inaccurate, leading to failure in conversion. Out of 735 sentences, 221 sentences failed in conversion, and 514 sentences augmented with DSyntS were obtained (2.7% of all sentences).

### 5.5. Matching propositions with extracted sentences

Propositions used in the recommendation utterances in MATCH have forms such as assert-food_quality(Babbo, superb) and assert-service(Babbo, good). On the other hand, the extracted sentences have meaning representations such as {food=5, service=5}. Since they have different forms, they cannot be directly matched to be associated. Therefore, we convert the propositions into meaning representations so that the matching can be performed on the same level. We prepared several simple conversion rules:

- First, convert assert-food_quality(X,Y) to *food=Y*. Then, convert Y to a number by mapping superb, excellent → 5, very_good → 4, good → 3, decent → 2, and mediocre → 1, yielding *food=N*, where N is a number from 1 to 5. The restaurant name X is omitted in the conversion. The same rule applies to assert-service(X,Y) and assert-decor(X,Y).
- Convert assert-nbhd(X,Y) to *location*. Here, nbhd stands for neighborhood.
- Convert assert-cuisine(X,Y) to *food type*.
- Convert assert-best(X) and assert-price(X,Y) to a null string.

For example, if the proposition is assert-food_quality(Babbo, superb), it is converted to {food=5}. The conversion can also be performed on a set of propositions. For example, assert-food_quality(Babbo, superb), assert-nbhd(Babbo, West Village), and assert-cuisine(Babbo, Italian) yield {food=5, location, food type}, which can be associated with "The best Spanish food in New York" to generate "The best Italian food in West Village."

### 5.6. Incorporating sentences into SPaRKy

Here, we explain how the extracted sentences are incorporated into SPaRKy. We propose replacing the DSyntS associated with the propositions and aggregation nodes by the DSyntS for the matching extracted sentences in the aggregation process. Note that an aggregation node corresponds to a set of propositions because it has propositions as its descendants.

This operation works as a *wrapper* for the aggregation process of SPaRKy. Figure 8 shows the pseudocode for a function *aggregation* which is called every time SPaRKy aggregates two aggregatable nodes to create an aggregation node just as p3 and p4 creates a1, and a2 and p1 creates a3 in Fig.3. Note that when these replacements are made, the fact that the extracted sentences are represented by their DSyntS means that the combination operations can be applied to them in exactly the same way as the DSyntS originally specified by hand in MATCH.

Relation, operation, ag1, and ag2 are the parameters for the function. They are the relation node that governs ag1 and ag2, the randomly selected clause-combining operation, and the two aggregatable nodes, respectively. The replace? is a function that randomly returns *true* or *false*. This function is used to suppress over-replacement. It can be seen from the pseudocode that this function works exactly the same as SPaRKy if the replace? always returns *false*. We call the generation process described here **SPaRKy+**, where + denotes the augmentation by the extracted sentences.

```
function aggregation(relation, operation, ag1, ag2) returns
aggregation node
  sents ← {}
  if relation = infer then
    sents ← find_matching_sentences(ag1,ag2)
  if sents.size > 0 and replace? = true then
    sent ← random_select(sents)
    return new aggregation node(sent)
  else
    sents ← find_matching_sentences(ag1)
    if sents.size > 0 and replace? = true then
      ag1.sent ← random_select(sents)
    sents ← find_matching_sentences(ag2)
    if sents.size > 0 and replace? = true then
      ag2.sent ← random_select(sents)
    return original_aggregation(relation, operation, ag1, ag2)
end
```

**Fig. 8**. Pseudocode for the aggregation process in SPaRKy+.

## 6. EXPERIMENT

We performed an experiment to verify our approach. We compared alternative utterances from SPaRKy and SPaRKy+ (augmented with 514 extracted sentences). We let each generator generate ten alternative sentences for 15 different text plans. Each text plan is about recommending a restaurant in New York. The utterances were forced to be different from one another, and the utterances of SPaRKy+ were controlled so that they incorporated at least one of the extracted sentences in each utterance. Five subjects, all native English speakers, judged the quality of the utterances. Each subject processed one text plan at a time, evaluating 20 utterances (ten from SPaRKy and ten from SPaRKy+), which were randomly ordered. Each subject rated each utterance on a scale of 1-5, where 1 is the worst and 5 the best while looking at the text plans. They were asked to rate the utterances as if they were hearing them spoken by the system and also to rate each utterance on its own merit. We collected 300 scores for 300 alternative utterances from each subject.

### 6.1. Results

Table 1 shows the distribution of the scores as well as the average scores for SPaRKy and SPaRKy+. One can see that the distribution of the scores is very different. SPaRKy utterances tend to be rated mostly in the range of 2-4, whereas SPaRKy+ utterances have a wider range. This may be because of the difference in the quality of the extracted sentences. For example, some sentences fit the propositions completely, while others do not fit, causing a number of highly rated alternative utterances as well as lowly rated ones. It should also be mentioned that the subjects rated very differently from one another. For example, subject 2 rated the alternative utterances of SPaRKy+ very low.

However, because the alternative utterances later undergo a ranking process in the context of a trainable sentence planner, the number of highly rated alternative utterances is more important than the average score. Using fully automatically extracted DSyntS, SPaRKy+ produced as many 5-rated alternative utterances as SPaRKy did. Since SPaRKy+ subsumes SPaRKy, and SPaRKy+ can generate additional alternative utterances without a loss in quality, we claim that SPaRKy+ is a better generator in terms of variation. Figure 9 shows some of the 5-rated alternative utterances of SPaRKy+. Notice the difference from Fig.4.

**Table 1**. Distribution of scores and average scores for SPaRKy and SPaRKy+ alternative utterances.

| score | subject 1 | | subject 2 | | subject 3 | | subject 4 | | subject 5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SPaRKy | SPaRKy+ | SPaRKy | SPaRKy+ | SPaRKy | SPaRKy+ | SPaRKy | SPaRKy+ | SPaRKy | SPaRKy+ |
| 1 | 9 | 25 | 10 | 43 | 0 | 4 | 4 | 11 | 0 | 0 |
| 2 | 51 | 47 | 23 | 22 | 24 | 26 | 20 | 36 | 8 | 16 |
| 3 | 61 | 40 | 25 | 21 | 49 | 60 | 61 | 48 | 86 | 67 |
| 4 | 26 | 26 | 34 | 22 | 65 | 42 | 43 | 39 | 49 | 52 |
| 5 | 3 | 12 | 58 | 42 | 12 | 18 | 22 | 16 | 7 | 15 |
| avg. | 2.75 | 2.69 | 3.71 | 2.99 | 3.43 | 3.29 | 3.39 | 3.09 | 3.37 | 3.44 |

| | |
|---|---|
| ex1 | Babbo has the best overall quality among the selected restaurants. The food is excellent, the wait staff is professional and the decor is beautiful and very comfortable. |
| ex2 | Babbo has the best overall quality among the selected restaurants since the food, service and atmosphere are all excellent. |
| ex3 | Great atmosphere, excellent food and friendly waiters. Babbo has the best overall quality among the selected restaurants. |
| ex4 | Ruby Foo's has the best overall quality among the selected restaurants. Waitstaff is friendly and knowledgeable and the Chinese, Japanese, Thai atmosphere is wonderful. The food is good and generous. |
| ex5 | Ruby Foo's has the best overall quality among the selected restaurants. Really good authentic Chinese, Japanese, Thai food. The best kept secret in Manhattan. |

**Fig. 9**. Examples of 5-rated alternative utterances generated by SPaRKy+.

## 7. SUMMARY AND FUTURE WORK

We proposed augmenting the variation of system utterances by incorporating sentences from corpora into the generation process in a trainable sentence planner. Experimental results show that our approach can successfully create a generator with augmented variation. We believe our approach is unique in that it incorporates natural occurring texts in system utterance generation and that it uses rating-annotated corpora to pinpoint the meaning of sentences. As future work, we plan to train a sentence ranker based on the scores we obtained in the experiment so that a workable restaurant recommendation system based on our approach can be created. We also plan to apply our approach to other types of utterances and domains where the corpora with ratings are available such as cinema and book recommendations.

## 8. REFERENCES


[1] Janienke Sturm, Els den Os, and Lou Boves, "Dialogue management in the Dutch ARISE train timetable information system," in *Proc. Eurospeech*, 1999, pp. 1419–1422.

[2] Alexander I. Rudnicky, Chirstina Bennett, Alan Black, Ananlada Chotomongcol, Kevin Lenzo, Alice Oh, and Rita Singh, "Task and domain specific modelling in the carnegie mellon communicator system," in *Proc. ICSLP*, 2000, vol. 2, pp. 130–134.

[3] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may I help you?," *Speech Comm.*, vol. 23, pp. 113–127, 1997.

[4] Takaaki Hori, Chiori Hori, and Yasuhiro Minami, "Fast on-the-fly composition for weighted finite-state transducers in 1.8 million-word vocabulary continuous speech recognition," in *Proc. ICSLP*, 2004, vol. 1, pp. 289–292.

[5] Anna Corazza, Renato De Mori, Roberto Gretter, and Giorgio Satta, "Computation of probabilities for an island-driven parser," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 936–950, 1991.

[6] Mariët Theune, "From monologue to dialogue: natural language generation in OVIS," in *AAAI 2003 Spring Symposium on Natural Language Generation in Written and Spoken Dialogue*, 2003, pp. 141–150.

[7] Lauren Baptist and Stephanie Seneff, "GENESIS-II: A versatile system for language generation in conversational system applications," in *Proc. ICSLP*, 2000, vol. 3, pp. 271–274.

[8] Marilyn Walker, Owen Rambow, and Monica Rogati, "SPoT: A trainable sentence planner," in *Proc. 2nd NAACL*, 2001, pp. 17–24.

[9] Marilyn Walker, Rashmi Prasad, and Amanda Stent, "A trainable generator for recommendations in multimodal dialog," in *Proc. Eurospeech*, 2003, pp. 1697–1700.

[10] François Mairesse and Marilyn Walker, "Learning to personalize spoken generation for dialogue systems," in *Proc. Eurospeech*, 2005, pp. 1881–1884.

[11] Benoit Lavoie and Owen Rambow, "A fast and portable realizer for text generation systems," in *Proc. 5th Applied NLP*, 1997, pp. 265–268.

[12] Regina Barzilay and Kathleen McKeown, "Extracting paraphrases from a parallel corpus," in *Proc. 39th ACL*, 2001, pp. 50–57.

[13] Regina Barzilay and Lillian Lee, "Learning to paraphrase: An unsupervised approach using multiple-sequence alignment," in *Proc. HLT and 4th NAACL*, 2003, pp. 16–23.

[14] Dekang Lin and Patrick Pantel, "Discovery of inference rules for question answering," *Natural Language Engineering*, vol. 7, no. 4, pp. 343–360, 2001.

[15] Yutaka Sasaki, Hideki Isozaki, Koji Kokuryou, Tsutomu Hirao, and Eisaku Maeda, "NTT's QA Systems for NTCIR QAC-1," in *In NTCIR Workshop 3 Question Answering Challenge (QAC)*, 2002.

[16] Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano, "COGEX: A logic prover for question answering," in *Proc. HLT and 4th NAACL*, 2003, pp. 87–93.

[17] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan, "GATE: A framework and graphical development environment for robust NLP tools and applications," in *Proc. 40th ACL*, 2002.

[18] Dekang Lin, "Dependency-based evaluation of MINIPAR," in *Workshop on the Evaluation of Parsing Systems*, 1998.