MOUNTAIN: A Translation-based Approach to Natural Language Generation for Dialog Systems

Brian Langner and Alan W Black

Language Technologies Institute
Carnegie Mellon University, Pittsburgh PA 15213, USA
{blangner,awb}@cs.cmu.edu

Abstract. This paper describes the MOUNTAIN language generation system, which is designed as a domain-independent, machine translation-based approach to generate utterances for use within a spoken dialog system. We describe the method used to train the generation engine from a corpus of in-domain human responses, and show typical output of the MOUNTAIN system. The results of our initial evaluation suggest this approach can be a viable method of language generation. We also discuss potential applications where this approach is likely to be most useful, as well as planned improvements to the system and future research using MOUNTAIN.

1 Introduction

Recent years have seen noticeable improvement in dialog systems, but even state-of-the-art systems still have limitations. Improvements in speech recognition and dialog management have made it possible to have more natural interactions, but frequently the potential of dialog systems to have truly natural conversations will be held back by their rudimentary language generation components. This observation is not new, having been noticed for years [Rambow et al., 2001,Chambers and Allen, 2004], but continues to be an issue.

Templates and canned text, because they are conceptually simple and require minimal expertise to write, are still one of the most commonly encountered methods of language generation in dialog systems. They are far more frequently seen than more state-of-the-art language generation systems, which tend to require a linguistics expert to be able to use effectively. This is despite the drawbacks of using templates, such as generally unnatural and repetitive output, significant creation cost, and a lack of portability between applications. Efforts to provide more varied template output [Oh and Rudnicky, 2000] have been able to reduce the perceived repetition, but the end result is still noticeably different from natural, human-generated output. Further, as the number and complexity of the templates increases, they become increasingly more expensive to design, create, and maintain.

The motivation for this work is to allow for spoken dialog systems with more human-like output than templates are typically capable of achieving, while maintaining the general simplicity of creating a typical template-based system.

Some users of spoken dialog systems will interact with a system using a human metaphor [Edlund et al., 2006], and these users expect human-like responses. For such users, the quality of the dialog system they are interacting with in part depends on how natural its responses are [Edlund et al., 2008].

2 Background

2.1 Approach

Because language generation for dialog systems has several key differences from general text generation, as discussed in [Horacek, 2003], we feel a dialog system should have language generation that is tailored for its needs. In particular, for many dialog platforms, including the Olympus framework [Bohus et al., 2007], generation is primarily only for realizing natural language surface forms that correspond to some internal state; often this is referred to as *tactical* generation. In practical terms, what is done is to convert the machine's representation of the dialog state into fluent and natural-sounding sentences. If one thinks of the dialog state representations as a highly structured (and possibly simplistic) language, then this task can be viewed as a *translation* problem, where the goal is to translate from the highly structured internal language of states to fluent and natural English (or any other language) sentences.

What is then required is a parallel data set, so that a mapping between the internal "language" and the natural surface forms can be learned. Unlike some other advanced NLG systems, we are not concerned with any significant amount of linguistic details, such as part of speech, agreement, or semantic relations, among others, in our input data. While these can clearly be helpful in producing high-quality output, there are significant costs for systems that require a high level of linguistic expertise or detailed annotation to be able to use; not all developers of dialog systems will be able to devote resources to have an expert design and annotate corpora for their generation module. Therefore, one of the considerations in our approach is that its implementation require only similar developer skills as writing templates.

Since most dialog applications exist within a known domain – that is, they do not have fully open-ended conversations – the set of things that can be talked about is closed, if possibly large. Thus, for tasks whose domain can be covered with a template-based system, it should also be possible to create a reasonably-sized parallel training corpus.

2.2 Related Work

There has been increasing interest in applying machine learning to spoken dialog research. Nearly all of the typical components of a dialog system have had some effort made to use machine learning to improve them; these are nicely summarized in [Lemon and Pietquin, 2007]. It seems, though, that trainable language generation for dialog has seen comparatively less work than other modules like ASR, dialog management, and related areas like user simulation.

Corpus- and statistically-based approaches to language generation, however, have been around for some time now, though only recently have they been starting to be applied to dialog NLG. These approaches are appealing due to their ability to leverage machine learning as has been done for other NLP tasks, to provide better results than typical approaches. The Nitrogen generation system [Langkilde and Knight, 1998], for example, derived statistical information from a corpus in order to rank and select grammar-generated surface forms. The work by [Ratnaparkhi, 2000] describes a generation system that can be trained from an annotated corpus, learning to produce surface forms using only a semantic representation. [Marciniak and Strube, 2005] describe using a corpus annotated with semantic and grammatical information, which is then used as a linguistic knowledge base for generation. Amalgam [Corston-Oliver et al., 2002] uses a similar classfication approach as Nitrogen, but takes logical forms as input. Work by [Zhong and Stent, 2005] is more similar to our proposed approach, directly using unannotated data to learn surface realizations. Likewise, a similar approach as we describe has used statistical translation methods for summarization and headline generation [Banko et al., 2000], with some degree of success. Additionally, [Sripada et al., 2003] generated weather forecasts using a parallel corpus of raw weather data and human-produced forecasts; following up this work, [Belz, 2005] compares N-gram selection and treebank based methods of statistical NLG in this domain.

Many of these approaches use significant amounts of linguistic knowledge (in some cases from annotations, and in some cases trained from data) in order to improve the natural language output they produce. The work we describe here attempts to use a broadly similar method – automatically learning generation output from a corpus of examples – but without any explicit linguistic annotation of the corpus.

3 MOUNTAIN: Machine Translation NLG

We present MOUNTAIN, a **m**achine **t**ranslation approach for **n**atural **l**anguage **g**eneration. In our implementation, we have used the Moses machine translation system [Koehn et al., 2007], which makes use of the Giza++ [Och and Ney, 2003] translation model training toolkit and the SRILM [Stolcke, 2002] language model toolkit. Though we have used Moses as the translation engine, because it and its support tools are freely available, nothing in the approach for MOUNTAIN restricts us to this specific engine.

MOUNTAIN requires only a parallel corpus of states in an internal language aligned with corresponding natural language surface forms. This parallel corpus is used to train a translation model which is capable of translating from the structured internal language to appropriate natural language. Additionally, the natural language corpus is used to train a language model for the target language. As described above, the internal language can be a structured representation of the dialog state – what the dialog manager intends to convey to the user.

Once the models have been trained, MOUNTAIN uses the translation engine to generate output utterances, given "sentences" from the internal language. Moses uses the trained models to translate into the target natural language; the

	Mon	Tue	Wed	Thu	Fri
6:00a	*avail*	*avail*			*avail*
7:00		*avail*	*avail*	*avail*	*avail*
8:00			*avail*		*avail*
9:00		*avail*		*avail*	*avail*
10:00			*avail*	*avail*	*avail*
11:00			*avail*	*avail*	*avail*
12:00p	*avail*	*avail*			
1:00	*avail*			*avail*	*avail*
2:00		*avail*			
3:00	*avail*		*avail*	*avail*	
4:00	*avail*		*avail*		*avail*
5:00		*avail*	*avail*	*avail*	

Fig. 1. A partial example of the presented schedule.

resulting output is the best result from the translation engine. Because of the way Moses works, the output may not only consist of examples lifted straight from the training corpus, but can also combine several examples to form novel sentences in the target language.

It should be noted that the entire process used by MOUNTAIN, from training to generation, does not require any specific linguistic analysis or domain knowledge, and thus can be considered a domain-independent approach. In fact, MOUNTAIN is also *language*-independent, provided the target language is able to be used by the training tools (such as the tokenizer and language model trainer).

4 Training and Use of MOUNTAIN

4.1 Application

To demonstrate our generation approach, we chose a fairly simple, but reasonable application: a scheduling task for a limited resource. In this case, the resource is a tennis court, available to reserve for one-hour blocks throughout the day. We have collected a corpus of human-generated responses in this domain. Given a schedule showing the availability status of a tennis court for the week, people were asked to answer questions from someone trying to reserve the court at various times. The requests generally were to reserve the court for one hour out of a several hour block by specifying a general time range (e.g. Wednesday afternoon, Monday evening, etc.). These included examples where the court was available for the entire range, only for part of the time, and where it was completely unavailable. Responders were told to answer naturally, as if someone had said to them, "I want to play on <day-time range>, what time can I reserve a court for?" Figure 1 shows an excerpt from an example schedule. Though trivial, this interaction can easily be seen as part of a larger spoken dialog application.

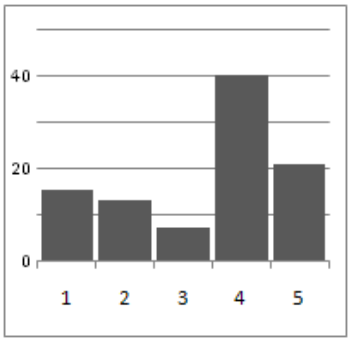

Fig. 2. Histogram of ratings given to the collected corpus.

4.2 Corpus

The collected corpus consists of about 800 responses, which are labeled with information about the specific schedule situation they describe. That information, which can be thought of as a dialog manager state that would be passed to an NLG module, is effectively an internal language. Thus, this corpus is, in fact, an aligned, parallel bilingual data set, defining equivalent English surface forms for the internal language. Sentences in the internal language consist of 3 tokens: a code corresponding to the day, a code corresponding to the time period, and a string that represents the court availability during that time. Example sentences include "111111 d2 t3" and "001110 d5 t1".

Approximately 20 people provided responses to form this corpus, all of whom were young adult, native speakers. Human responses to these requests were varied, but generally consistent. For cases where no reservation could be made, many responses used some form of the phrase "I'm sorry". It is also interesting to note that several responses offered suggestions for other similar reservations that could be made instead.

Nearly one-quarter of the corpus has been scored by a human evaluator. In keeping with the task design, the evaluator was shown a schedule and request, along with the response from the corpus, which was presented as their co-worker's answer. They were then asked to rate that answer on a 5-point Likert scale, and additionally provide their own answer to the request. Figure 2 shows a histogram of the ratings. Most responses are rated well; however, a significant portion of the corpus has a low score. This is likely due to errors made by the human responders. For example, some responders misread the schedule and reversed the availability (instead of all slots free, they responded as if all slots were full).

4.3 Training for Generation

To make the collected corpus more complete, we first make several fairly standard modifications. Since it seems reasonable in this domain to have the output be independent of the day-of-week – that is, a sentence used to describe a full court on Monday afternoon can also be used for Thursday afternoon, changing only

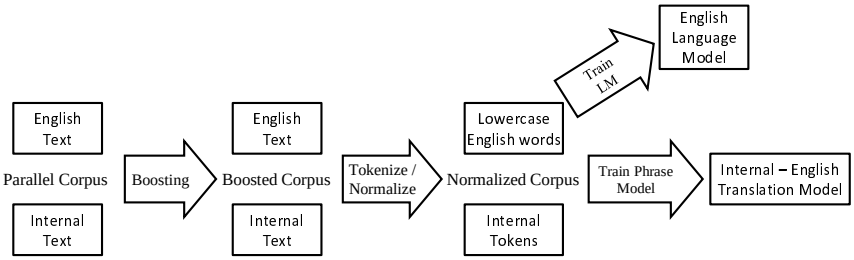

Fig. 3. The MOUNTAIN training process from initial parallel corpus to trained models.

the word for the day – we boosted the training data by cloning responses for all 7 days, changing the day word in the sentence as appropriate. Additionally, since the internal language uses a fixed token to correspond to particular days (i.e., d3 = Wednesday), we added these translations directly to the bilingual dictionary. Likewise, the tokens for time (i.e., t1 = morning) were similarly added. The result is that the training corpus size is increased to about 4500 state-response pairs.

With the modified parallel corpus, we then train a translation model using the Moses tools. The first step is to tokenize and case-normalize both the internal and English training sentences. Due to the designed structure of the internal language, the English tokenizer can be used for both sets of sentences. Once this is complete, we train a trigram language model for the target language (English in our case) using the SRILM tool. Finally, we train a phrase model using the Moses toolkit, which produces the necessary phrase and reordering tables for translation. This process is shown in Figure 3.

4.4 Output

Table 1 shows several example input-output pairs. Though several outputs have come entirely from single examples in the training data, likely due to the presence of these exact pairs in the parallel corpus, most of the outputs have combined

Table 1. Example output from the MOUNTAIN system

000000 d5 t3	friday evening is completely closed
100000 d2 t2	the only time available is noon
111111 d4 t1	the court is open all morning
111111 d1 t3	you can reserve a court anytime on monday evening
100011 d5 t3	six , ten or eleven
010011 d3 t2	you can reserve a court at 1pm , 4pm and 5pm on wednesday
011001 d4 t3	any time but 6 , 9 and 10
111011 d7 d2	afternoon except the 3pm block
111100 d1 t2	you can reserve a court is free anytime from noon until 3
110111 d6 t3	saturday evening . ooh , that

phrases from two or more examples, producing a novel English sentence unseen in the training data. Overall, more than three quarters of the generated responses are not present in the original corpus.

The examples in the table show the variety of ways availability can be expressed. Though the method we are using here generates the same response for a particular input, similar but non-identical inputs can produce noticeable surface variation – but similar semantics – in the resulting output. The source of this is likely the variation present in the training corpus.

Though many of the generated responses are interesting and appropriate answers, there are also responses that contain errors of varying severity. Some, like the second-to-last example in the table, are relatively trivial with some grammaticality problems. These are obviously not ideal, though with appropriate prosody when spoken, they could be made to sound like a person's natural, unplanned, conversational response. Other errors, however, are more problematic, such as the last example in the table. Here we see what appears to be a partial translation failure, since the day and time are correctly rendered in the response; however, the remaining important content information has been omitted. Minimizing or eliminating these sorts of errors is crucial to the effectiveness of MOUNTAIN as a generation method.

5 Evaluation of MOUNTAIN Generation

While an informal examination of MOUNTAIN output seems to show it is more or less capable of generating acceptable responses (though with some errors), it is clear that a more structured evaluation is required. How best to evaluate language generation systems has been an unresolved question for some time now. [Dale and Mellish, 1998] discussed many of the pertinent issues and dimensions, of which we are most concerned with output assessment and application potential – in particular, the notions of quality and accuracy, and possibly fluency as well. However, our approach is also corpus-based and informed by machine translation; BLEU score [Papineni et al., 2002] is a standard reported metric, and has been used in some corpus-based NLG evaluation as well [Belz, 2005]. BLEU effectively measures N-gram agreement between reference and test strings.

5.1 Automatic Evaluation: BLEU

To evaluate MOUNTAIN using BLEU, we used a held-out test set of about 120 responses from the collected corpus that were not part of the training process. These responses were taken from throughout the corpus at various intervals – approximately every eighth entry, to ensure the resulting test set was representative of the corpus and domain. The internal language half of this bilingual set is used as input to MOUNTAIN to produce the test output; the English half of the bilingual set is used as a reference. Individual N-gram scores for the default MOUNTAIN system are as follows:

1-gram	2-gram	3-gram	4-gram	5-gram
0.3198	0.1022	0.0525	0.0300	0.0202

While it is not unusual to see scores decrease with larger N-grams, our default system has a fairly steep dropoff. However, there are several potential areas for improvement available to us. First, recall that the training corpus contains some amount of error, as described in Section 4.2, and that some of the corpus has been scored by human evaluators. We can use the scores to exclude some responses from the training set, with the assumption being that removing poor or incorrect responses will improve the resulting generation. We performed this analysis using various exclusion thresholds; these results are shown in Table 2. The training corpus for these systems includes only responses with ratings above the indicated threshold, plus any unscored responses from the original corpus.

These results show clear improvement over the baseline system, with statistically significant improvement ($p < 0.1$) for all systems; values in bold are significant with $p < 0.05$. Removing only the poorest examples from the training data does not help as much as removing more – even some which were considered good by a human. However, once the threshold becomes too high, the BLEU scores begin to decrease, likely due to a combination of data sparsity and the well-rated examples being overwhelmed by unrated examples in the corpus. The systems with thresholds of 2 and 3 show similar performance, though the lower threshold is marginally superior with larger N-grams, and the higher threshold is better with smaller N-grams. Because we are trying to generate sentences, we feel this gives a small preference to the system with a threshold of 2.

We also investigated the effects of the model's *distortion limit* on the resulting output. Distortion limit refers to the amount of word reordering allowed when translating source sentences to the target language. The default value in the Moses training process is 6 – that is, words may appear up to 6 words away from their source-language neighbor in the target language. Using our baseline system, we examined values from 0 (no reordering) to 8, as well as unlimited reordering. We found identical BLEU scores for systems with a limit $\geq$3, and minimal differences (mostly less than .001) between systems with other values. Though we had expected a performance effect as the distortion limit changed, our results showed almost no impact whatsoever. Our original intuition was that a higher distortion limit would improve results, due to the nature of our source and target sentences: 3-token source sentences map to much longer target sentences. We had thought that a distortion limit that was too low might cause some correct (and possibly preferred) translations from being generated; however, this does not seem to be the case. Additionally, varying the distortion limit with our improved systems that excluded training data also showed no noticeable difference.

Table 2. BLEU scores for systems with excluded training data based on human scoring

System	1-gram	2-gram	3-gram	4-gram	5-gram
Baseline	0.3198	0.1022	0.0525	0.0300	0.0202
Rating > 1	0.4376	0.1729	0.1079	0.0746	0.0597
Rating > 2	**0.4491**	**0.1919**	0.1169	0.0872	0.0747
Rating > 3	**0.4742**	**0.1963**	0.1212	0.0866	0.0722
Rating > 4	**0.4596**	0.1762	0.1023	0.0693	0.0611

Table 3. METEOR scores for systems with excluded training data

System	Precision	Recall	f1	Total
Baseline	0.4225	0.2013	0.2727	0.1950
Rating > 1	0.4489	0.2097	0.2859	0.2028
Rating > 2	0.4533	0.2248	0.3009	0.2218
Rating > 3	0.4834	0.2148	0.2974	0.2146
Rating > 4	0.4481	0.2030	0.2794	0.1971

5.2 METEOR: A Different Automatic Measure

Though BLEU has been shown to be well-correlated to translation results, it may not be the best metric to use for evaluating NLG output because it tends to view output as a set of N-grams rather than sentences. This may be well correlated to translation quality, but it seems possible that NLG quality may have other dependencies. METEOR [Banerjee and Lavie, 2005] may be a useful measure for our needs, as a translation metric that takes more sentence-level information into account than BLEU.

We used METEOR to score the same systems described in Section 5.1, trained with some of the corpus data excluded. Results from the same test set, showing precision, recall, f1, and total METEOR score, are in Table 3. As with the BLEU scores, there is an improvement seen over the baseline system by excluding training data, and this improvement decreases when the exclusion threshold is too high. Likewise, the systems with thresholds of 2 and 3 are similar, with the former system being marginally better in performance.

5.3 Human-scored Evaluation

However, the evaluation method still most often used for NLG systems is scoring by a human evaluator, despite its expense. We performed a small scale evaluation using 4 human evaluators, all young adult, native speakers. The evaluation task was structured similarly to the corpus evaluation described in Section 4.3,

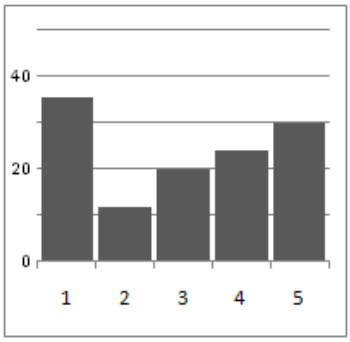

Fig. 4. Histogram of ratings for MOUNTAIN-generated responses.

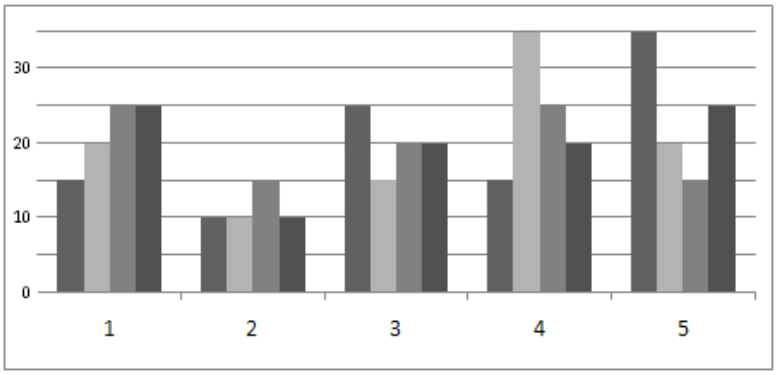

Fig. 5. Per-person histogram of ratings for MOUNTAIN-generated responses. Each color represents an individual evaluator.

substituting MOUNTAIN-generated output for the original human responses in the corpus. Overall, the MOUNTAIN output has a lower average rating than the human responses (3.1 compared to 3.4), but a broadly similar rating pattern. Figure 4 shows a histogram of these ratings (compare to Figure 2 for human-generated responses). The main difference is a much higher incidence of a poor rating (1) for the machine-generated output, but similar rates for good (4) and excellent (5) scores. This seems to indicate MOUNTAIN can generate output similar in quality to human answers, but when it fails, it fails miserably.

The limited amount of human evaluation done precludes a formal examination of inter-person agreement. However, a per-person histogram (Figure 5) shows similar rating patterns from each of the human evaluators. Future plans for this work include a significantly larger-scale human evaluation, which should give us a more complete subjective measure of the output quality.

6 Discussion

Our results show that MOUNTAIN is capable of generating natural, human-like output. In the examples, the day name is frequently not present in the output. Though a slot-filling generation system will typically fill in specific information such as that, in a human-human conversation once the day has been established it does not need to continue to be said. MOUNTAIN, because it is trained from a natural corpus, is able to produce this sort of human-like output.

Compared to other NLG systems, MOUNTAIN is requires relatively little time to set up. The largest and most expensive part is corpus collection; once the training corpus is available, the training time itself is minimal. For the test application described here, it was about 20 seconds, though a real application would clearly require a larger training corpus. Still, for well-defined, and mostly closed-vocabulary tasks – which most spoken dialog systems could be described as – the training time should be measurable in minutes. The time required to generate output is nearly instantaneous, or similar to a template-based system. Obtaining

a suitable training corpus is still expensive, though there are potential solutions. Besides including responses from the system developers, which is similar in cost and skill to template-writing, other data sources such as transcribed Wizard-of-Oz interactions could be used. Potentially, any available human-human dialogs for the application domain could also be included in the training corpus, as long as they could be transcribed and annotated with the internal language.

As described in Section 3, the output of MOUNTAIN discussed in this work was the single-best result from the translation engine. Though this can result in reasonable generation for many inputs, it also means that an input will always have the same generated response, which will have the same repetitiveness problems as template-based NLG. However, Moses can be configured to output N-best lists of translation results, rather than a single-best result. If MOUNTAIN instead selects its responses from the N-best list, that will provide more variation in the resulting output, and possibly prevent a human listener from perceiving it as unnaturally repetitive. Further investigation of this, including determining an appropriate size for the list and evaluating user perception of the generation, is planned.

Currently, MOUNTAIN's biggest limitation is that its gross errors occur too frequently for it to be used comfortably in a real application. To a certain extent this can be explained by the approach still being relatively new – there are many possible improvements that have not yet been attempted. However, it must be solved before MOUNTAIN can be considered a viable generation system for dialog applications. We have already considered several possibilities, including weighting well-rated examples more heavily in the training data rather than just using a simple threshold to exclude poorly-rated ones, increasing the "vocabulary" of the internal language to reduce the complexity of mapping a single token to long English phrases, as well as the obvious get-more-training-data approach. We plan to implement these potential improvements as soon as possible. Additionally, Moses provides methods for tuning the translation model that we have not yet done. This may provide the potential for improving the model, though the relatively small vocabulary size in this domain may limit its effectiveness.

Finally, we would like to test MOUNTAIN in a new application with a "real" dialog system. Since the method is domain-independent (as long as the internal language can be specified), it should be relatively simple to implement this generation method for a new system.

References

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, Ann Arbor, MI.

[Banko et al., 2000] Banko, M., Mittal, V. O., and Witbrock, M. J. (2000). Headline generation based on statistical translation. In *ACL 2000*, pages 318–325, Hong Kong.

[Belz, 2005] Belz, A. (2005). Statistical generation: Three methods compared and evaluated. In *ENLG 2005*, pages 15–23, Aberdeen, Scotland.

[Bohus et al., 2007] Bohus, D., Raux, A., Harris, T., Eskenazi, M., and Rudnicky, A. (2007). Olympus: an open-source framework for conversational spoken language in-

terface research. In *HLT-NAACL 2007 workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*.

[Chambers and Allen, 2004] Chambers, N. and Allen, J. (2004). Stochastic language generation in a dialogue system: Toward a domain independent generator. In *5th SIGdial Workshop on Discourse and Dialogue*, pages 9–18, Cambridge, MA, USA.

[Corston-Oliver et al., 2002] Corston-Oliver, S., Gamon, M., Ringger, E., and Moore, R. (2002). An overview of Amalgam: A machine-learned generation module. In *INLG 2002*, pages 33–40, New York.

[Dale and Mellish, 1998] Dale, R. and Mellish, C. (1998). Towards evaluation in natural language generation. In *Proceedings First International Conference on Language Resources and Evaluation*, pages 555–562.

[Edlund et al., 2008] Edlund, J., Gustafson, J., Heldner, M., and Hjalmarsson, A. (2008). Towards human-like spoken dialogue systems. *Speech Communication*, 50(8-9):630 – 645.

[Edlund et al., 2006] Edlund, J., Heldner, M., and Gustafson, J. (2006). Two faces of spoken dialogue systems. In *Interspeech 2006 Dialogue on Dialogues Workshop*, Pittsburgh, PA.

[Horacek, 2003] Horacek, H. (2003). Text generation methods for dialog systems. In *2003 AAAI Spring Symposium*, pages 52–54, Palo Alto, CA.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: open source toolkit for statistical machine translation. In *ACL 2007: Interactive Poster and Demonstration Sessions*, pages 177–180, Prague, Czech Republic.

[Langkilde and Knight, 1998] Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *ACL/ICCL 1998*, pages 704–710, Montreal, Quebec, Canada.

[Lemon and Pietquin, 2007] Lemon, O. and Pietquin, O. (2007). Machine learning for spoken dialogue systems. In *Interspeech 2007*, Antwerp, Belgium.

[Marciniak and Strube, 2005] Marciniak, T. and Strube, M. (2005). Using an annotated corpus as a knowledge source for language generation. In *Workshop on Using Corpora for Natural Language Generation*, Birmingham, UK.

[Och and Ney, 2003] Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

[Oh and Rudnicky, 2000] Oh, A. and Rudnicky, A. (2000). Stochastic language generation for spoken dialogue systems. In *ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, Seattle, WA.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318, Philadelphia, PA.

[Rambow et al., 2001] Rambow, O., Bangalore, S., and Walker, M. (2001). Natural language generation in dialog systems. In *HLT 2001*, pages 1–4, San Diego, CA.

[Ratnaparkhi, 2000] Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. In *NAACL 2000*, pages 194–201, Seattle, WA.

[Sripada et al., 2003] Sripada, S., Reiter, E., Hunter, J., and Yu, J. (2003). Exploiting a parallel text-data corpus. In *Corpus Linguistics 2003*, Lancaster, UK.

[Stolcke, 2002] Stolcke, A. (2002). SRILM – An extensible language modeling toolkit. In *ICSLP 2002*, pages 901–904, Denver, CO.

[Zhong and Stent, 2005] Zhong, H. and Stent, A. (2005). Building surface realizers automatically from corpora. In *Workshop on Using Corpora for Natural Language Generation*, Birmingham, UK.