

# Training a Maximum Entropy Model for Surface Realization

Hua Cheng<sup>1</sup>, Fuliang Weng<sup>2</sup>, Niti Hantaweepant<sup>1</sup>, Lawrence Cavedon<sup>1</sup>, Stanley Peters<sup>1</sup>

<sup>1</sup>CSLI, Stanford University, Stanford, CA 94305, U.S.A.

<sup>2</sup>RTC, Robert Bosch Corp., Palo Alto, CA 94008, U.S.A.

huac,lcavedon,peters@csli.stanford.edu, niti@stanford.edu

fuliang.weng@rtc.bosch.com

## Abstract

Most existing statistical surface realizers either make use of hand-crafted grammars to provide coverage or are tuned to specific applications. This paper describes an initial effort toward building a statistical surface realization model that provides both precision and coverage. We trained a Maximum Entropy model that given a predicate-argument semantic representation, predicts the surface form for realizing a semantic concept and the ordering of sibling semantic concepts and their parent, on the Penn TreeBank and Proposition Bank corpora. Initial results have shown that the precisions for predicting surface forms and orderings reached 80% and 90% respectively, on a held-out part of Penn TreeBank. We use the model to generate sentences from our domain representations. We are in the process of evaluating the model on a corpus collected for our in-car applications.

## 1. Introduction

Designing spoken interface systems that can converse with the user like a human speech partner has attracted much attention in both academic and industrial research environment in recent years. In our in-car applications, we are interested in a dialog interface that will help reduce the driver's overall cognitive load by allowing them to operate in-car devices, such as obtaining navigation information and information about local points of interest through conversation. Such a system should understand the driver's requests and produce responses based on the driver's knowledge, the conversational context, and the external situation. In addition, it is very desirable to have dialog system modules, including the generation component, portable to different applications.

In this paper, we focus on the response generation part of the dialog interface, in particular, the realization of a given semantic frame to an English sentence. This process is normally called surface realization. The surface realizer under development are designed to handle applications, such as:

- Navigation: provide turn-by-turn navigation instructions that make references to landmarks.
- MP3 player operation: provide information about the driver's MP3 collection, and help the driver to organize and operate the MP3 player.
- Restaurant reservation: help the driver to filter through a large number of restaurants to find the best option.

Because the surface realizer is to be deployed for different in-car applications, it needs to be robust and domain independent. Previous surface realizers combine statistical and symbolic techniques. Two examples of this hybrid approach

are FERGUS [1] and HALogen [6, 7]. FERGUS (Flexible Empiricist/Rationalist Generation Using Syntax) uses XTAG grammar [13] to compose phrase-structure trees using substitution and adjunction. It takes as input a dependency tree for a sentence, where each node is marked only with a lexeme. The Tree Chooser tags each input node with a TAG tree using a stochastic tree model; the Unraveler then produces a word lattice of all possible linearizations for the semi-specified TAG tree; and finally the Linear Precedence (LP) Chooser chooses the most likely traversal of the lattice according to a trigram language model. HALogen is also a broad-coverage generator that first transforms a labeled feature-value structure into a forest of possible expressions using a hand-crafted grammar, and then ranks the expressions using a 250-million word n-gram language model trained on WSJ newspaper text.

Pure statistical realizers have only been applied to small applications such as the ATIS (Air Travel Information Service) domain. The most prominent work in this direction is described in [12], which compares several realization methods: an n-gram model, a trained dependency model, and a combination of a hand-built dependency like grammar with content-driven conditions for applying rules and corpus statistics. These models are used to find the word sequence with the highest probability that mentions all of the input attributes exactly once. The result based on human judgment shows that the hybrid approach achieves the best result, and the other two are close.

One problem with these experimental statistical models is that they are hard to scale up because they directly predict the word form from domain semantic representation and the space of possible words in a real application is huge.

In this paper, we continue to explore the use of statistical approaches for surface realization because [12] has shown promising results in this direction. In particular, we aim at designing a domain-independent and scalable statistical model that can achieve comparable results with existing systems, and in the mean time reducing the need for hand-crafted grammars that are time-consuming to build and maintain. We intend to train such a model using existing resources, e.g., corpora annotated with both syntactic and semantic information, because they not only facilitate automatic generation rule induction, but also enable the use of existing metrics (e.g., [10]) for automatic evaluation of the generation results. This paper describes our initial effort in constructing the model, but we are yet to evaluate its performance.

## 2. Model for Surface Realization

We target at the generation of dependency trees rather than constituency trees because of the direct correlation between dependency tree and functional semantic representation, which we as-

sume is the format for our input.

We formulate the problem of surface realization as creating a syntactic representation in the form of a dependency tree given a semantic frame, which is a recursive structure that contains one or more head concepts and their argument concepts at each level. We decompose this problem into three parts:

1. Decide the phrase type that is used to realize each semantic concept;
2. Order a head and its children concepts into a linear sequence;
3. Surface smooth to take care of such issues as verb-noun agreement and number variations.

In this paper, we focus on the first two parts. The third aspect can be addressed with the technique described in [11]. We assume that generation decisions for a concept only depend on that of its head (i.e., parent) and the concept just ahead of it, and are independent of any concept not adjacent to it or coming after it. Context information that can be used for realization include the features of the head, and the semantic and functional features (e.g., semantic roles [4]) about the concept to be realized. In the model decomposition, whether the decision of phrase type should come before or after ordering is subject to experiment.

If the order of two semantic components,  $o(a_i, a_j)$ , is determined before their phrase types are decided, the features that may affect the decision of the ordering of the two components with respect to each other as well as to their head may include the head concept (*head*) and its phrase type (*pos*, mostly whether it is a verb phrase or a noun phrase), the semantic role of the first component (*role*), its concept (*con*) and length (*len*), as well as the same features for the other component, which can be expressed as:

$$p(o(a_i, a_j)|head, pos, role_i, con_i, len_i, role_j, con_j, len_j) \quad (1)$$

The length of a concept at this point is the number of semantic concepts subsumed by the target concept in the input semantic representation.

The features that may affect the decision of the phrase type,  $pt_n$ , used to realize a semantic concept include the head concept and its phrase type (again mostly a verb phrase or noun phrase), the semantic role of the component, its concept and length, as well as the semantic role, phrase type and length of the component just ahead of it, which can be expressed as:

$$p(pt_n|head, pos, role_n, con_n, len_n, role_{n-1}, pt_{n-1}, len_{n-1}) \quad (2)$$

If the ordering decision comes in later, we just need to substitute the *con* features with *pt* in the ordering probability, and vice versa in the phrase type probability.

Suppose we have an input semantic structure (*Fs*) with a head and three arguments,  $a_1, a_2$  and  $a_3$ . At the top level, the probability can be computed as:

$$\begin{aligned} & p(pt_1, pt_2, pt_3, o(a_1, a_2, a_3)|Fs) \\ = & p(o(a_1, a_2)|Fs) \cdot p(o(a_2, a_3)|Fs) \cdot p(o(a_1, a_3)|Fs) \cdot \\ & p(pt_1, pt_2, pt_3|o(a_1, a_2, a_3), Fs) \\ = & p(o(a_1, a_2)|Fs) \cdot p(o(a_2, a_3)|Fs) \cdot p(o(a_1, a_3)|Fs) \cdot \\ & p(pt_{1'}|o(a_1, a_2, a_3), Fs) \cdot p(pt_{2'}|o(a_1, a_2, a_3), pt_{1'}, Fs) \cdot \\ & p(pt_{3'}|o(a_1, a_2, a_3), pt_{2'}, Fs) \\ = & p(o(a_1, a_2)|Fs) \cdot p(o(a_2, a_3)|Fs) \cdot p(o(a_1, a_3)|Fs) \cdot \\ & p(pt_{1'}|nil, Fs) \cdot p(pt_{2'}|pt_{1'}, Fs) \cdot p(pt_{3'}|pt_{2'}, Fs) \end{aligned}$$

The above equation means that the probability of the syntactic subtree can be calculated as the production of the probability of the complete order of the arguments, and those of the phrase types given the complete order and the phrase type to its adjacent left.  $pt_{1'}$  means the phrase type of the argument ordered the first after the ordering process, so its adjacent left sibling is *nil*.  $o(a_1, a_2, a_3)$  is omitted from the last line of the equation because the order is implied in the new argument sequence.

This computation is done at each level of the input structure in a top-down manner until there is no node to expand. The probability of the complete tree can be computed as the production of all probabilities derived from the input structure.

### 3. Maximum Entropy Model Training

We adopt Conditional Maximum Entropy modeling because it is a mathematically sound approach and it has the flexibility of incorporating different features. We aim at constructing maximum entropy models to estimate the conditional probabilities (1) and (2), based on the formulation below [2, 14]:

$$p(y|x) = sum(y|x)/Z(x) \quad (3)$$

where,

$$sum(y|x) = exp(\sum_j \lambda_j f_j(x, y)) \quad (4)$$

$$Z(x) = \sum_y sum(y|x) \quad (5)$$

To train these models, we need a corpus annotated with dependency relations and semantic features for each sentence component. Two commonly used resources for training stochastic language models are PropBank [5] and FrameNet [3]. We chose PropBank to make use of both the syntactic and semantic information presented in the corpora. In this section, we describe how we train our MaxEnt models using PropBank.

#### 3.1. Bosch MaxEnt Toolkit

The MaxEnt training toolkit developed at the Research Technology Center of Robert Bosch Corp. [14] was used to train our models. The training data to the toolkit takes the following form:

$$(x_1, x_2, \dots, x_{10}; y_1, c_1; y_2, c_2, \dots; y_n, c_n)$$

Where  $x_1, x_2, \dots, x_{10}$  is a ten dimensional input vector (the conditional component), and  $y_i$  and  $c_i$  are an output and the corresponding count. Each input vector can be mapped to one or more outputs with different counts. These counts are the frequencies of the corresponding outputs given the context.

The output of the toolkit are the weights,  $\lambda_i$ , in Equation (4). These weights represent the importance of the corresponding features, which have the value of 1 or 0 depending on whether the features present in the corpora.

A number of training parameters can be specified to achieve different training results, for example, the maximum feature size (we used 20,000 and 11,000 for phrase type and order models respectively), feature templates (what patterns a good feature is likely to fall in, used to cut down the search space), and cutoff counts (in our case 4).

The input and output data used by the toolkit are all encoded with respect to words, semantic roles, length, phrase types and orderings. The output model needs to be decoded before being used by the generation module.

### 3.2. Data Preparation

For simplification, we treat all verb arguments and adjuncts as well as noun modifiers as modifiers. We assume that there is a one-to-one mapping between a concept and a stemmed word.

We follow these steps to prepare training data:

#### Obtaining dependency trees with semantic information:

We have mentioned that we need dependency relations in the training data, whereas Penn Treebank [9] only contains syntactic constituency annotations. Therefore, we use the tool developed by Rebecca Hwa at Maryland University to convert the Treebank trees into the dependency formulation. We then insert Propbank semantic role information into the dependency trees, marking only the head node of a constituent with a role, which means that the entire subtree headed by this node has the noted role. PropBank provides explicit frame files for each verb, which list the syntactic and semantic variations of that verb. Sometimes labels following the naming conventions of theta-role theory are also given in addition to the verb-specific mnemonic labels. In this case, we use these labels rather than Arg0, . . . , Arg5.

**Encoding:** We use stemmed words as concepts, and Treebank part-of-speech (POS) tags as phrase types. We encode all stemmed words, POS tags and semantic roles appearing in our dependency trees. However, PropBank only have verb level semantic annotations; the same level of classification does not exist for noun modifiers. In addition, no other corpus with both syntactic and NP level semantic annotations is available to us. So we adopt the Treebank functional labels as roles, in most cases *mod*. This simplification might affect the performance of the trained models; however, we believe the part of speech information and word information may supplement the insufficient granularity to a good extent.

**Collecting ordering patterns:** For each modifier, we want to predict how it is ordered in a linear string with respect to its head, and all sibling modifiers. We collect all pair-wise orderings of modifiers with the same head. These partial pair-wise orderings together determine the full order among modifiers and their head. Suppose we have two semantic concepts  $a$ ,  $b$  and their parent  $h$ , we count occurrences of the six possible sequences (e.g., *ahb* and *hba*) in the corpora and keep those counts that are not 0. Each tree in the corpora is traversed top-down to collect ordering patterns at all levels.

**Collecting patterns for phrase type:** The idea is that given a stemmed word and a POS/phrase type, we can usually determine the surface form of the phrase except for prepositional phrases (PP), in which case an appropriate preposition needs to be chosen based on the semantic functionality of the phrase. In order to capture the preposition usage in the corpora, we classify PPs into more refined categories such as IN-in and IN-until, meaning that these are PPs headed by the preposition *in* and *until* respectively. We only single out the top 35 most frequent prepositions appearing in the corpora to avoid efficiency problem due to a large output space. So the overall number of phrase type categories is around 80. We use IN as a general category for less frequent prepositions, and OTHER for all other POS, mostly such tags as FN and EX, which are not important to generation.

### 3.3. Training Results

We separate Treebank and PropBank data into two parts, 9/10 for training and 1/10 for testing. The oracle is calculated by using the most frequent output for a given input vector. The oracles for phrase type and ordering prediction are both 98%.

The accuracy of the trained models on the testing data are given in Table 1. The table shows that the ordering model achieves higher accuracy when using lexical information rather than phrase types. The phrase type model remains the same in both cases. Therefore, the surface realizer should first order the semantic components and their head, and then determine the phrase type for realizing each component. The feature size column gives the number of features used when the best performance is achieved.

Model	Accuracy	Feature Size
Type-Order		
Type	0.80	15000
Order	0.875	6000
Order-Type		
Type	0.80	15000
Order	0.906	8000

Table 1: Accuracy of the MaxEnt models

Several reasons might contribute to the relatively low accuracy of the phrase type model. The lack of semantic annotation for NP modifiers is likely to be a major cause. Currently we use the functional annotation from the dependency tree converter to supply semantic information for NP modifiers. A better semantic annotation is likely to improve the performance of the model. It is also possible that we are missing important context information. We are in the process of analyzing the data.

## 4. Surface Realization based on the MaxEnt Models

In this section, we describe how we use the trained models for surface realization. This is work in progress, and we are yet to evaluate the performance of the module.

Our semantic representation is based on the HALogen labeled feature-value structure<sup>1</sup>, adapted to use the PropBank semantic notations. In this representation, the semantic content of a sentence is represented by a tree whose nodes are each marked with a concept and a semantic role. For example, the sentence "You should turn left at the next intersection" is represented as:

```
(e1 / turn
  :agent (l1 / listener)
  :direction (l2 / left)
  :location (i1 / intersection
    :mod (n1 / next))
  :modal should)
```

Given a semantic structure like this, our surface realizer traverses the structure top-down, first ordering the children of each head and then determining the phrase type for realizing each child. This process makes use of the two MaxEnt models trained from PropBank, and the conditional probabilities are calculated using Equations (3), (4) and (5).

Suppose a head has four child semantic components, whose semantic roles are  $r_1$ ,  $r_2$ ,  $r_3$  and  $r_4$ . To get a complete order of these components, we use a matrix as in Table 2.

We first calculate the probability of each cell, and only keep the  $N$  best orderings (in our case  $N$  is 3). Then from these cells,

<sup>1</sup>Detailed description of the labeled feature-value structure can be found at <http://www.isi.edu/licensed-sw/halogen/interlingua.html>.

Roles	$r_2$	$r_3$	$r_4$
$r_1$	$o(r_1, r_2)$	$o(r_1, r_3)$	$o(r_1, r_4)$
$r_2$	x	$o(r_2, r_3)$	$o(r_2, r_4)$
$r_3$	x	x	$o(r_3, r_4)$

Table 2: Ordering matrix

we pick the one with the highest probability (say  $o(r_1, r_3)$ ) and start to extend this partial ordering by looking at the probabilities of its surrounding cells. We pick the cell with the next highest probability (say  $o(r_1, r_4)$ ) and merge the partial orders to get an extended partial order with one more semantic component  $o(r_1, r_3, r_4)$ . We continue looking at the surrounding cells in the same manner until there is no cell left to merge. At each step, we only keep the N best results to reduce the amount of computations needed. This process results in N best complete orderings of all semantic components and their head.

It is possible that two partial orders are inconsistent, and therefore impossible to merge. We penalize each final ordering that does not incorporate all semantic components based on the number of missing components. Sometimes no ordering incorporating all semantic components can be found, in which case we generate a sentence only with the consistently ordered components. An alternative is to attach extra components at the end of the sentence.

Based on the N best orderings, we then predict the N best phrase types for each semantic component. The probability of a result syntactic tree is calculated using the method described in Section 2. Finally, the tree with the highest probability is linearized to produce the surface word string.

At the moment, our surface realizer can generate sentences such as "Turn left at the next intersection" and "The mother made a cake for her daughter yesterday". Its robustness needs to be enhanced to enable a full scale evaluation.

## 5. Future Work

We have achieved good accuracy on training Maximum Entropy models for predicting the ordering between semantic components and the phrase type for realizing them. We are yet to see how well these models perform in generating good English sentences. Although a comparison of generated sentences with the original Treebank sentences will give us some sense of the quality of the output, human judgment is inevitable for evaluating different realizations of the same semantic representation.

Our future work will include a surface smoothing process to take care of verb-noun agreement and number variations. This process will again follow the over-generation and ranking methodology, that is, first generating a word lattice containing all possible word level variations given a phrase type and a concept, and then scoring these variations using an n-gram language model.

We will also need to connect our surface realizer with a content planner and a referring expression generation component to enable end-to-end generation. This complete system will be evaluated on the corpus collected for our in-car applications.

## 6. Acknowledgements

The work described in this paper is a part of the NIST ATP funded project Driving Your Car with Conversational Language. We would like to thank NIST for funding the project.

We would also like to thank Dr. Rebecca Hwa for making her syntactic tree conversion software available to us.

## 7. References

- [1] Bangalore, S. and Rambow, O., "Exploiting a Probabilistic Hierarchical Model for Generation", Proceedings of the International Conference on Computational Linguistics (COLING), 2000.
- [2] Berger, A., Della Pietra, S., and Della Pietra, V., "A Maximum Entropy Approach to Natural Language Processing". Computational Linguistics, 22 (1): 39-71, 1996.
- [3] Fillmore, C. and Baker, C., "Framenet: Frame Semantics Meets the Corpus", The 74th Annual Meeting of the Linguistic Society of America, 2000.
- [4] Fillmore, C., "Frame Semantics and the Nature of Language", Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech, vol. 280, pp. 20-32, 1976.
- [5] Kingsbury, P., Palmer, M. and Marcus, M., "Adding Semantic Annotation to the Penn TreeBank", Proceedings of the Human Language Technology Conference. San Diego, California, 2002.
- [6] Langkilde, I., "Forest-based Statistical Sentence Generation", Proceedings of the North American Meeting of the Association for Computational Linguistics (NAACL), 2000.
- [7] Langkilde, I. and Knight, K., "Generation that Exploits Corpus-based Statistical Knowledge", Proceedings of COLING-ACL, 1998.
- [8] Lavoie, B. and Rambow, O., "A Fast and Portable Realizer for Text Generation Systems", Proceedings of the 5th Conference on Applied Natural Language processing, Washington DC., 1997.
- [9] Marcus, M., Santorini, B. and Marcinkiewicz, M., "Building a Large Annotated Corpus of English: the Penn Treebank", Computational Linguistics, vol.19, 1993.
- [10] Papineni, K., Roukos, S., Ward, T., Zhu, W., "Bleu: A Method for Automatic Evaluation of Machine Translation", IBM Research Report, RC22176(W0109-022), 2001.
- [11] Rayner, M., Carter, D., Bouillon, P., Digalakis, V. and Wren M. (eds). Spoken Language Translator. Cambridge University Press, 2000.
- [12] Ratnaparkhi, A., "Trainable Approaches to Surface Natural Language Generation and Their Application to Conversational Dialog Systems", Computer Speech and Language, 16:435-455, 2002.
- [13] XTAG Research Group. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, the Institute for Research in Cognitive Science, University of Pennsylvania, 2001.
- [14] Zhou, Y., Weng, F., Wu L. and Schmidt, H., "A Fast Algorithm for Feature Selection in Conditional Maximum Entropy Modeling", Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp. 153-159, 2003.