Human-like Natural Language Generation Using Monte Carlo Tree Search

Kaori Kumagai **Ichiro Kobayashi**
Ochanomizu University
{kaori.kumagai,koba}@is.ocha.ac.jp

Daichi Mochihashi
The Institute of Statistical Mathematics
daichi@ism.ac.jp

Hideki Asoh
AIST
h.asoh@aist.go.jp

Tomoaki Nakamura **Takayuki Nagai**
University of Electro-Communications
{tnakamura,tnagai}@ee.uec.ac.jp

Abstract

We propose a method of probabilistic natural language generation observing both a syntactic structure and an input of situational content. We employed Monte Carlo Tree Search for this nontrivial search problem, employing context-free grammar rules as search operators and evaluating numerous putative generations from these two aspects using logistic regression and n-gram language model. Through several experiments, we confirmed that our method can effectively generate sentences with various words and phrasings.

1 Introduction

People unconsciously produce utterances in daily life according to different situations. When a person encounters a situation in which a dog eats a piece of bread, he or she retrieves appropriate words and creates a natural sentence, retaining the dependent relationships among the words in proper order, to describe the situation. This ability of natural language generation (NLG) from situations will become essential for robotics and conversational agents in the future.

However, this problem is intrinsically difficult because it is hard to encode what to say into a sentence while ensuring its syntactic correctness. We propose to use Monte Carlo tree search (MCTS) (Kocsis and Szepesvari, 2006; Browne et al., 2012), a stochastic search algorithm for decision processes, to find an optimal solution in the decision space. We build a search tree of possible syntactic trees to generate a sentence, by selecting proper rules through numerous random simulations of possible yields.

2 NLG with MCTS simulations

2.1 MCTS

MCTS combines random simulation and best-first search in its search process (Kocsis and Szepesvari, 2006). It has been successfully applied as an algorithm for playing Go game and similar planning problems. In fact, both Go game and NLG share the same characteristic: their outputs can be evaluated only when their process reaches the last state. Therefore, we think that the process of NLG can be represented in MCTS simulations.

MCTS uses the upper confidence bounds one (UCB1) value to determine the next move from a viewpoint of multi-armed bandit problem (Katehakis and Veinott, 1987):

$$\mathrm{UCB1} = v_i + C\sqrt{\frac{\log N}{n_i}}. \qquad (1)$$

Here, v_i is the winning rate of candidate i, C is an adjustment coefficient, N is the total number of simulations, and n_i is the number of visits to the candidate i. The first term of equation (1) corresponds to exploitation and the second term corresponds to exploration in simulation, achieving a balanced search between the two factors (Auer et al., 2002).

2.2 Algorithm

MCTS provides opportunities for selecting various syntactic structures and words in a generated sentence in our case. We use context-free grammar (CFG) rules obtained from the *Brown corpus* as a search operator in MCTS. The MCTS algorithm is shown in Figure 1

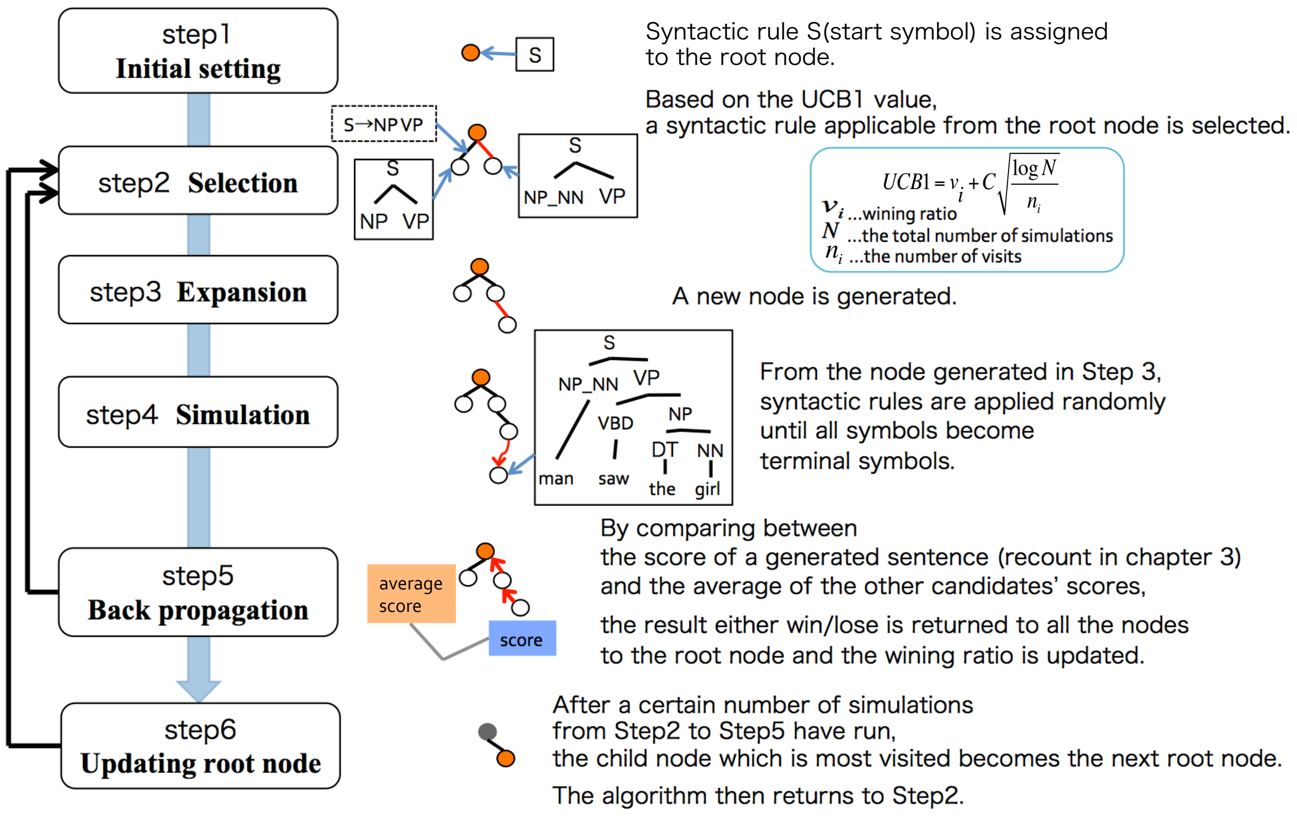

step1
Initial setting
step2 Selection
step3 Expansion
step4 Simulation
step5
Back propagation
step6
Updating root node
S
S→NP VP
NP VP
NP_NN VP
Syntactic rule S(start symbol) is assigned to the root node.
Based on the UCB1 value, a syntactic rule applicable from the root node is selected.
$UCB1 = v_i + C\sqrt{\frac{\log N}{n_i}}$
v_i ...wining ratio
N ...the total number of simulations
n_i ...the number of visits
A new node is generated.
VBD
NP
DT
NN
man
saw
the
girl
From the node generated in Step 3, syntactic rules are applied randomly until all symbols become terminal symbols.
average score
score
By comparing between the score of a generated sentence (recount in chapter 3) and the average of the other candidates' scores,
the result either win/lose is returned to all the nodes to the root node and the wining ratio is updated.
After a certain number of simulations from Step2 to Step5 have run,
the child node which is most visited becomes the next root node.
The algorithm then returns to Step2.

Figure 1: MCTS algorithm for NLG

Essentially, our MCTS builds a search space of possible derivations, and starting from the initial symbol S we iteratively determine what rule to apply to extend the current tree, by simulating numerous possible derivations from the candidate rules.

3 Evaluating generated sentences

When using MCTS in NLG, it is important how the simulation result, i.e., a generated sentence, is evaluated. In generating a sentence, unlike playing Go game, it is not easy for a machine to decide whether a generated sentence is natural for us because the result cannot be naturally represented by a win or a lose. This necessitates giving machines the ability to evaluate whether a generated sentence is natural or not. Regarding this problem, Okanohara and Tsujii (2007) proposed a method to use a semi-Markov class model to identify the grammaticality of the sentence. Similarly, in this study we have introduced two evaluation scores: one for syntactic structure and the other for the n-gram language model.

3.1 Evaluation of syntactic structure

For this purpose, we use logistic regression with partial syntactic trees of a sentence as its features for identifying whether it is natural or not. Figure 2 illustrates the procedure of building a classifier for structure evaluation.

We used the *Brown corpus*[1] and extracted 4,661 sentences consisting of three to seven words other than punctuation marks. Those extracted sentences were parsed using the *Stanford parser*[2], and a set of *CFGs* was created based on its result. The *CFGs* contained 7,220 grammar rules and 5,867 terminal symbols.

As the training data for the classifier, we regard syntactic subtrees of sentences in the *Brown corpus* as the positive examples, and subtrees of the sentences generated from random simulation of *CFGs* as negative examples.

As we see in Figure 2, we have prepared 46,610

[1]http://clu.uni.no/icame/browneks.html
[2]http://nlp.stanford.edu/software/lex-parser.shtml

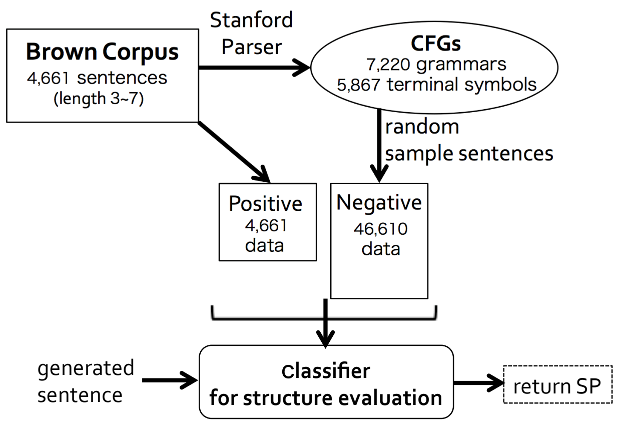


Figure 2: Building a classifier for structured sentences.

syntactically incorrect sentences as negative examples – the reason why there are ten times as many negative examples as positive examples is that it is highly possible that more syntactically incorrect than correct sentences can be generated using MCTS simulations with *CFGs*.

We use FREQuent Tree miner (FREQT)[3] to extract syntactic subtrees of sentences (Abe et al., 2002; Zaki, 2002). From all of the subtrees obtained, we use the subtrees from which only the terminal symbols have been removed as the features for the classifier, because of our exclusive focus on syntactic structure with nonterminal symbols. We call the probability of the output from this classifier as Syntactic Probability (SP) (see, Figure 2).

We evaluated the classifier with 10-fold cross validation and obtained 98% accuracy for the test data.

3.2 Evaluation for n-gram language model

To evaluate the word sequence in a generated sentence, we conducted an experiment to compare the accuracy of evaluation between two kinds of the n-gram based scores. One is the score calculating the perplexity of trigrams with Kneser-Ney smoothing. We call this score 'PP'. The other is the score called Acceptability proposed in Lau et al. (2015), which measures the acceptability of a sentence for an English native speaker. In this study, we use the Acceptability (AP) below for a sentence s:

$$Acceptability(s) = \log\left(\frac{p(s)}{p_{\text{uni}}(s)}\right)^{\frac{1}{|s|}} \quad (2)$$

[3]http://chasen.org/~taku/software/freqt/

As an n-gram language model $p(s)$, we use trigrams with Kneser-Ney smoothing (Kneser and Ney, 1995). In (2), $p_{\text{uni}}(s)$ denotes the probability with a unigram distribution and (2) measures a relative fluency per word as compared to baseline probability p_{uni}.

3.3 How to decide the win/lose of sentences

At the Step 5 in the MCTS algorithm, the final decision of a win or a lose (1 or 0) about the sentence is returned by the score based on the 'SP' and 'PP or AP' decisions as follows: (i) if it wins on both 'SP' and 'PP or AP', the score is 1; (ii) if it fails with 'SP', the score is 0; (iii) if it wins on 'SP' but fails on 'PP or AP', the score is 0.5. This reflects our assumption that the generated sentence from *CFGs* must be syntactical at least. Those processes of (i), (ii), and (iii) are summarized in Table 1.

Table 1: How to determine the final decision

SP	PP or AP	score
0	0	0
0	1	0
1	0	0.5
1	1	1

4 Generation with Situational Information

We have so far discussed "*how to say*" part of NLG. Next, we consider "*what to say*" in terms of how to flexibly choose words that are suitable for a given situation. We will explain how words are chosen in a generated sentence with the linguistic resources shown in Figure 3.

Let us assume that some words have been specified as the content to speak about, say "dog" and "run" (*Given words* in Figure 3) and we consider how to incorporate them into the sentence to generate. There are multiple ways to describe the content with natural language sentences. For example, we could say "dog" as "puppy" or "run" as "dash". Therefore, considering the possibility of flexibly choosing words, we used *word2vec* (Mikolov et al., 2013) trained on Wikipedia to determine the set of similar words whose cosine distance $>$ 0.5 as the dictionary (*Similar words*). Further, in order to adding the peripheral words of the given words to

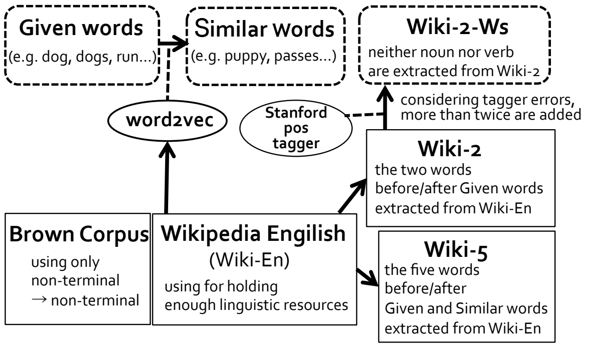

Given words
(e.g. dog, dogs, run...)
Similar Words
(e.g. puppy, passes...)
Wiki-2-Ws
neither noun nor verb are extracted from Wiki-2
word2vec
Stanford pos tagger
considering tagger errors, more than twice are added
Wiki-2
the two words before/after Given words extracted from Wiki-En
Brown Corpus
using only non-terminal → non-terminal
Wikipedia Engilish (Wiki-En)
using for holding enough linguistic resources
Wiki-5
the five words before/after Given and Similar words extracted from Wiki-En

Figure 3: Relationship between linguistic resources. Boxes with dashed lines are used as dictionaries for generation within MCTS.

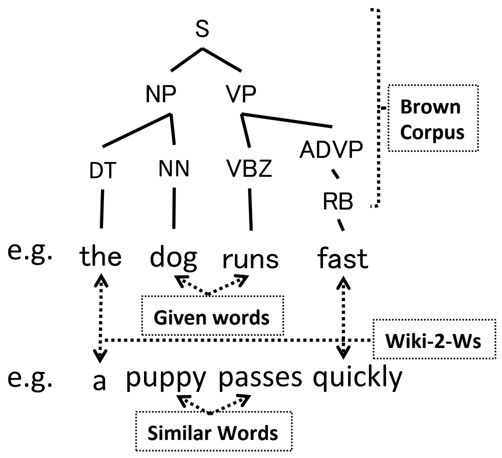


Figure 4: Example of a generated sentence.

the dictionary, we prepared another dictionary from two words window before/after of the given words on Wikipedia as the candidate words other than noun or verbs (*Wiki-2-Ws*). During generation, we generate sentences using these dictionaries depending on the part-of-speech of each word to reduce the search space, and classified a sentence as "lose" when it contains words out of the *Given words* and *Similar words*. *Wiki-5* is the statistics from five-words window before/after of the *Given words* and the *Similar words* on Wikipedia to compute AP or PP.

Figure 4 illustrates a generated syntax tree example with the linguistic resources shown in Figure 3.

Note that selecting proper linguistic resources is a nontrivial problem for generation: because there are huge number of possibilities to use them with different syntactic trees, it requires a ingenious method like MCTS to effectively combine them with a grammatical tree as well as retaining fluency with respect to n-gram probabilities. We used the information to feed as a bag of words for simplicity, and aim to use more sophisticated use of input as a distinct problem from the proposed algorithm.

5 Related studies

As for the nondeterministic approach to NLG, some studies view NLG as a planning problem. Koller and Stone (2007) used automated classical planning techniques to derive a plan converted into a sentence. Kondadadi et al. (2013) consolidated macro and micro planning as well as surface realization stages into one statistical learning process. As another way to handle the indeterminate characteristics of NLG, Lemon (2008; 2011) and Rieser and Lemon (2009) modeled dialog as Markov decision processes (MDPs) and solved them by means of reinforcement learning (Sutton and Barto, 1998).

Similar to our approach, McKinley and Ray (2014) considered the NLG process as an MDP with a suitably defined reward function to achieve efficient sentence generation using an MCTS algorithm. As another nondeterministic approach using a neural language model (Bengio et al., 2003), Wen et al. (2015) used the Long Short-term Memory generator, which can learn from unaligned data by concurrently optimizing sentence planning and surface realization using a simple cross-entropy training criterion and easily achieve language variation by sampling from output candidates. However, this method predicts just a word sequence and does not consider syntactic structures.

As another search-based algorithm to generate a sentence considering syntactic structures, Liu et al. (2015) proposed a syntactic linearization of given words using beam-search for an appropriate structure of a sentence. However, it just treats the problem of word ordering and does not consider generations with the given words, which does not always include the given words in themselves. Technically, their method employs a beam search with a predefined beamwidth. On the other hand, MCTS realizes an efficient search that does not restrict the search range in advance.

Moreover, Silver et al. (2016) developed AlphaGo which defeated a top level professional Go player.

They combined MCTS with a deep reinforcement learning framework and then provided MCTS with learning ability; both policy and value networks of the system are trained to predict human expert behaviors using deep reinforcement learning. This framework is expected to be applied to NLG in the future.

6 Experiments

In this section, we conducted experiments with two cases where we evaluate only syntactic structure of a generated sentence, and evaluate both syntactic structure and n-gram language model characteristic of a generated sentence.

6.1 Experimental settings

We used the *CFGs* and the classifier to evaluate the structure of a sentence mentioned in section 3.1.

In addition, we set the number of MCTS simulations at a node as the number of wins that reach five times as many as other candidate nodes at that time. The reason we used a dynamic change of simulation number is that the next root node must be chosen based on a clear difference in winning percentage compared to other candidate nodes.

6.2 Evaluation for syntactic structure

First, we focus on only syntactic structure, and conducted generation experiments to evaluate it. As the evaluation score for a generated sentence, we employ only 'SP'. Table 3 shows some generated sentences.

Table 3: Generated sentences based on the evaluation for only syntactic structure

Generated sentences	SP
all mass nudged no teacher	0.999
this principle observed all super-condamine	0.999
all kay sank all round	0.999
some camping departs	0.994
those rim made these amount	0.999

Looking at the above sentences, we see that they are syntactically correct – they have syntactic structure of either SVO or SV. The scores of them are approximately 0.99, therefore, we see that correct syntactic structure are apparently generated based on the classifier.

6.3 Generation with two evaluation indices

Next, we conducted an experiment based on both evaluation criteria for syntactic structure and an n-gram language model. The 'win' or 'lose' is decided as explained in section 3.3.

Furthermore, in order to confirm that we can generate sentences of various lengths, we introduce a constraint on sentence length: if a generated sentence has a length shorter than the predefined length, the simulation result is regarded as a lose. Moreover, as mentioned in section 3.2, we used PP and AP to compare the results evaluated by them.

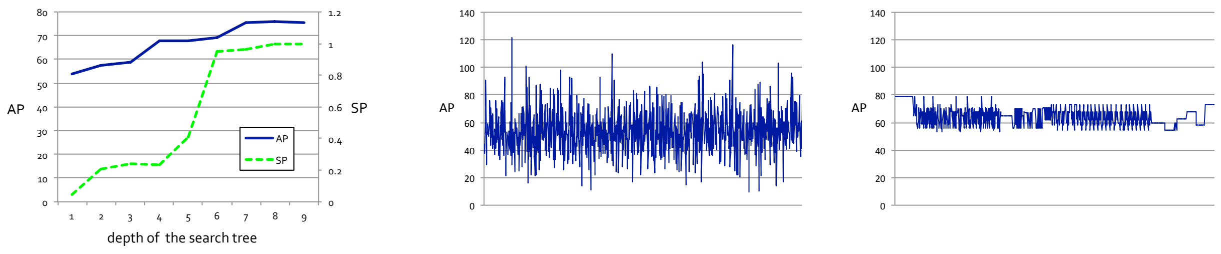
Here, because lower perplexity is better, the simulation result is regarded as a win when the score is less than the average of those of other candidate nodes. Table 2 shows some generated sentences.

From the results shown in Table 2, we see that syntactically correct sentences are generated. In the case of using AP, we also see that low frequency words were selected in generating sentences, and a sentence is generated without any influence from word frequency. On the other hand, we have confirmed that when a generated sentence is evaluated by PP, the sentence is influenced more by word fre-

Table 2: Generation with AP and PP

Length	Generated sentences	SP	AP
5	those memorial neglected neither contraction-extension	0.999	85.46
6	all marketing half-straightened neither contraction-extension understandingly	0.999	91.79

Length	Generated sentences	SP	PP
5	no theirs defied no improvement	0.999	1008.63
6	no one said his own work	0.999	156.85



(a) Depths of MCTS search space with respect to acceptability and syntactic correctness. (b) Acceptability scores: Initial 1,000 simulations. (c) Acceptability scores: Last 1,000 simulations.

Figure 5: Statistics during MCTS simulations to generate a sentence from the situation $\{$*boy, play, basketball*$\}$.

quency than when AP is used.

7 Experiment with situational information

In this experiment, we aim to generate sentences with specific words as given situational information.

7.1 Experimental settings

We use the same linguistic resources mentioned in section 4. In the experiment, we dealt with three cases where the content for a generated sentence has the words, either “dog, run”, “dog, eat, bread”, or “boy, play, basketball”.

As for lexical selection, we put a constraint that a word to generate must have positive bigram counts from the preceding word in the Wiki-5 statistics. Setting this constraint avoids unlikely words in advance to achieve more appropriate lexical selection within MTCS framework.

Furthermore, as for the constraints on the number of simulations and on the length of a generated sentence, they are the same settings mentioned in section 6.1 and 6.3, respectively.

7.2 Experimental results

Table 4 shows an example of generated sentences from different situations. Comparing AP with PP, when AP is used, a wide variety of words are selected. As a concrete example, in the case where the words “dog”, “eat” and “bread” are specified, when PP was used for evaluation of the n-gram language model, the word “every” was selected as an adjective many times. In contrast, when AP was used, words such as “another”, “neither” and “all” were selected.

Figure 5(a) shows the trends in the average values of AP and SP whenever the root node is updated in MCTS simulations with an example of generating a sentence with the specified words $\{$*boy, play, basketball*$\}$. We see that the value of SP is approximately 0.1 initially and then converges around 0.99 as exploration deepens. As for AP, we have not observed any clear convergence in the exploration process, however, at the initial stage of exploration we have observed instead that generated sentences do not satisfy the generation constraints, e.g., whose length is too short or too long, therefore, the values of more than 100 or less than 20 have been observed. Figures 5(b) and 5(c) shows the values of AP of the initial and final 1,000 simulations, respectively. From these figures, we see that AP converges to a particular value.

For the output of situation (a) in Table 4 “*every dog runs her cat*”, we have observed the sentences that resulted in “lose” during the generation in Table 5.

Table 5: Sentences that resulted in “lose” to generate *every dog runs her cat* during the MCTS generation.

Sentence	SP	AP
be more in	0.126	28.03
be shall run or dog american	0.056	41.41
either dog was puppy	0.999	46.76
le dog runs his mr. three	0.999	34.14

8 Conclusions

In this paper, we proposed the first attempt to exploit MCTS for natural language generation. Because MCTS allows a stochastic search using the possible yields, namely the sentence from the current point of search, we can leverage both the syntactic structure (CFG) and statistical fluency (n-grams)

Table 4: NLG with situational information. Situation of (a) = {*dog*,*run*}, (b) = {*dog*,*eat*,*bread*}, (c) = {*boy*,*play*,*basketball*}.

Situ.	Len	Generated sentences	SP	AP
(a)	4	either dog runs his cat	0.999	39.95
	5	every dog runs her cat	0.999	37.13
(b)	5	every dog eats his bread	0.999	37.58
	5	another dog eats his bread	0.999	42.73
	6	neither dog eats its own bread	0.999	42.71
	6	all dog eats its original bread	0.999	41.33
(c)	5	girls tennis played the rugby	0.998	59.65
	5	volleyball boys played both rugby	0.998	72.03
	6	girls tennis was played senior football	0.996	71.96
	6	girls tennis played played and los	0.996	66.55

Situ.	Len	Generated sentences	SP	PP
(a)	4	this cat is run	0.999	76.87
	5	some dog runs his cat	0.999	350.72
(b)	5	every dog eats his bread	0.999	310.92
	5	no dog eats its flour	0.999	383.59
	6	every dog eats its first flour	0.999	380.06
	6	every dog eats its original bread	0.999	358.97
(c)	5	boys soccer played the tennis	0.999	317.28
	5	girls tennis played an football	0.999	448.10
	6	le boy plays her own tennis	0.999	549.45
	6	boys tennis was played to all the	0.996	114.92

through a logistic regression to determine the "win" or "lose" of generated sentences.

While our results are still preliminary using limited linguistic resources, we believe this method is beneficial for future NLG integrating both the syntax and semantics in an ingenious statistical way.

References

K. Abe, S. Kawasoe, T. Asai, H. Arimura, and S. Arikawa. 2002. Optimized substructure discovery for semi-structured data. In *PKDD 2002*, volume 6, pages 1–14.

P. Auer, N. Cesa-Bianchi, and P. Fischer. 2002. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47:235–256.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. Survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1).

M. N. Katehakis and A. F. Veinott. 1987. The multi-armed bandit problem : Decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*, volume 1, pages 181–184.

L. Kocsis and C. Szepesvari. 2006. Bandit based monte carlo planning. In *ECML 2006*, pages 282–293.

A. Koller and M. Stone. 2007. Sentence generation as a planning problem. In *International Natural Language Generation Workshop*, volume 12, pages 17–24.

R. Kondadadi, B. Howald, and F. Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *ACL 2013*, volume 51, pages 1406–1415.

J. H. Lau, A. Clark, and S. Lappin. 2015. Unsupervised prediction of acceptability judgements. In *ACL 2015*, volume 53, pages 15–1000.

O. Lemon. 2008. Adaptive natural language generation in dialogue using reinforcement learning. In *Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, pages 141–148.

O. Lemon. 2011. Learning what to say and how to say it:joint optimization of spoken dialogue management and natural language generation. *Computer Speech and Language*, 25(2):210–221.

Y. Liu, Y. Zhang, W. Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *NAACL 2015*, pages 113–122.

N. McKinley and S. Ray. 2014. A decision-theoretic approach to natural language generation. In *ACL 2014*, volume 52, pages 14–1052.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*, pages 3111–3119.

D. Okanohara and J. Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *ACL 2007*, volume 45, pages 73–80.

V. Rieser and O. Lemon. 2009. Natural language generation as planning under uncertainty for spoken dialogue systems. In *EACL 2009*, pages 683–691.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.

R. S. Sutton and A. G. Barto. 1998. Reinforcement learning: An introduction. *MIT Press*.

T. Wen, M. Gašić, N. Mrkšić, P. Su, D. Vandyke, and S. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP 2015*, pages 1711–1721.

M. J. Zaki. 2002. Efficiently mining frequent trees in a forest. In *KDD 2002*, volume 8, pages 71–80.