Generalized Hierarchical Word Sequence Framework for Language Modeling

Xiaoyi Wu†, Kevin Duh†† and Yuji Matsumoto†

Language modeling is a fundamental research problem that has wide application for many NLP tasks. For estimating probabilities of natural language sentences, most research on language modeling use n-gram based approaches to factor sentence probabilities. However, the assumption under n-gram models is not robust enough to cope with the data sparseness problem, which affects the final performance of language models. In this paper, we propose a generalized hierarchical word sequence framework, where different word association scores can be adopted to rearrange word sequences in a totally unsupervised fashion. Unlike the n-gram which factors sentence probability from left-to-right, our model factors using a more flexible strategy. For evaluation, we compare our rearranged word sequences to normal n-gram word sequences. Both intrinsic and extrinsic experiments verify that our language model can achieve better performance, proving that our method can be considered as a better alternative for n-gram language models.

Key Words: Language Modeling, Sparseness Problem, Cognitive Grammar, Hierarchical Word Sequence (HWS), Word Association Measures

1 Introduction

Probabilistic Language Modeling is a fundamental research direction of Natural Language Processing. It is widely used in various application such as machine translation (Brown, Cocke, Pietra, Pietra, Jelinek, Lafferty, Mercer, and Roossin 1990), spelling correction (Mays, Damerau, and Mercer 1990), speech recognition (Rabiner and Juang 1993), word prediction (Bickel, Haider, and Scheffer 2005) and so on.

Most research about Probabilistic Language Modeling, such as Katz back-off (Katz 1987), Kneser-Ney (Kneser and Ney 1995), and modified Kneser-Ney (Chen and Goodman 1999), only focus on smoothing methods because they all take the n-gram approach (Shannon 1948) as a default setting for modeling word sequences in a sentence. Yet even with 30 years worth of newswire text, more than one third of all trigrams are still unseen (Allison, Guthrie, Guthrie, Liu, and Wilks 2005), which cannot be distinguished accurately even using a high-performance

† NARA Institute of Science and Technology
†† Johns Hopkins University
smoothing method such as modified Kneser-Ney (abbreviated as MKN).

An alternative solution is to factor the language model probabilities such that the number of unseen sequences are reduced. It is necessary to extract them in another way, instead of only using the information of left-to-right continuous word order.

In (Guthrie, Allison, Liu, and Guthrie 2006), skip-gram (Huang, Alleva, Hon, Hwang, and Lee 1993)\(^1\) is proposed to overcome the data sparseness problem. For each n-gram word sequence, the skip-gram model enumerates all possible word combinations to increase valid sequences. This has truly helped to decrease the unseen sequences, but we should not neglect the fact that it also brings a greatly increase of processing time and redundant contexts.

In (Wu and Matsumoto 2014), we propose a heuristic approach to convert any raw sentence into a hierarchical word sequence (abbreviated as HWS) structure, by which much more valid word sequences can be modeled while remaining the model size as small as that of n-gram. In (Wu and Matsumoto 2015; Wu, Matsumoto, Duh, and Shindo 2015), instead of only using the information of word frequency, we also use the information of direction and word association to construct higher quality HWS structures. However, they are all specific methods based on certain heuristic assumptions. For the purpose of further improvements, it is also necessary to generalize those models into one unified framework under an integrated theory.

In this paper, inspired by the viewpoint of sentence generation process from cognitive grammar, proposed by Langacker (Langacker 1986, 1999, 2008), we propose a generalized hierarchical word sequence framework, by which various word association scores can be used for rearranging word sequences for language modeling. Then we use both intrinsic and extrinsic experiments to verify the effectiveness of these strategies.

The paper makes the following scientific contributions:

1) We present a generalized hierarchical word sequence framework, extending our previous work on specific HWS methods.

2) We adopt various association scores for rearranging word sequences and empirically verify their better performance in comparison to n-grams (including skip-n-grams) in language modeling.

This paper is organized as follows. In Section 2, we introduce the basic idea of cognitive grammar structure and convert it into a special dependency structure. We also discuss its advantage in language modeling. Then inspired by this idea, we propose the HWS language model and present a generalized framework in Section 3. In Sections 4 and 5, we show the effectiveness

\(^{1}\) The k-skip-n-grams for a sentence \(w_1, \ldots, w_m\) is defined as the set \(\{w_{i_1}, w_{i_2}, \ldots, w_{i_n} | \sum_{j=1}^{n} i_j - i_{j-1} < k\}\).
of our model by both intrinsic experiments and extrinsic experiments. Finally, we summarize our findings in Section 6.

2 Cognitive Grammar Structure and Its Advantage

2.1 Cognitive Grammar Structure and CG-based Dependency Structure

The structure of cognitive grammar is proposed by Langacker (Langacker 1986, 1999, 2008). In cognitive grammar, it is presumed that complex expressions are formed from schematic patterns, such as ‘Vs X in the Nb’ and ‘a N1 + less N2’, which are called ‘schemas’.

“They are not themselves full-fledged expressions but patterns abstracted from them and potentially used in forming new ones. To this extent they are grammar-like, since grammar by definition comprises the patterns used in forming complex expressions.” (Langacker 2008, p. 20)

This quote not only means that schemas are the core units of cognitive grammar, but also indicates that schemas are relative concepts. From a schema, a more specific expression can be instantiated, while a more schematic expression can also be abstracted.

An example is given in (Langacker 2008, p. 21) as ⟨Vs X in the Nb⟩ → ⟨kick X in the shin⟩ → ⟨kick my pet giraffe in the shin⟩. In this example, ⟨Vs X in the Nb⟩ is the schema with the highest schematicity, while the schema ⟨kick X in the shin⟩ is a more specific one. The structure of cognitive grammar is shown in Figure 1.

For the purpose of applying this structure to language modeling, we use following processing to simplify and convert it into a special dependency structure.

Fig. 1 An example of structure of cognitive grammar

2 Vs represents ‘strike motion’, while Nb represents ‘body part’.
3 Schemas.
4 In this example, Langacker regards ‘my pet giraffe’ as one whole unit. In fact, it can also be further abstracted as schema ‘my Npossession’ on the principle of cognitive grammar.
In the cognitive grammar structure, schemas are gradually specified from more schematic patterns to less schematic ones. Locally, constituencies are hierarchically generated from certain positions of patterns. Taking Figure 1 as the example, ‘shin’ depends on the category ‘\(N_b\)’, which is a part of schema \(\langle V_s \ X \ in \ the \ N_b \rangle\). Since \(N_b\) is always on the certain position of schema \(\langle V_s \ X \ in \ the \ N_b \rangle\), if we keep the word order information, then we can remove the category part from this schema and simply assume that ‘shin’ depends on ‘in the’.

Further, in a categories-removed schema, such as ‘in the’ in above step, we assume following words depends on their preceding words, then ‘the’ depends on ‘in’.

Applying above processing hierarchically, the cognitive grammar structure given in the Figure 1 can be converted into a special dependency structure as shown in Figure 2. We call such kind of dependency structure as CG-based dependency structure in order to distinguish it from the conventional dependency grammar structure and cognitive grammar structure.

2.2 The Advantages of CG-based Dependency Structure in Language Modeling

Due to the generativity (also called productivity) of language, it is difficult to calculate the probability of a whole sentence directly. A more practical way is to divide the sentence into some sequences of words and to compute the joint probability using the Chain Rule. Thus, the performance of language models depend largely on how to arrange those word sequences, more specifically, for each word in one sentence, how to determine its context. Theoretically, the more accurately we can determine the context for each word, the more those word sequences can reflect the real process of sentence generation, consequently, the higher performance the language models can achieve.

We take an example from WSJ corpus and convert it into different structures to demonstrate the difference among n-grams, dependency grammar and CG-based structure in arranging word
sequences. In each structure of Figure 3, arrows represent the sentence generation process (in n-gram model, generation process is equivalent to word order), thus, the parent nodes could be considered as the context of their child node, and word sequences can be arranged as (parent nodes, child node).

As shown in Figure 3 (a), n-gram language models adopt an **utterance-oriented** way to determine word sequences, which assume that the preceding n-1 words are the context of each word in the sentence. In the actual utterance, signals are indeed generated one by one, however, for each word in a sentence, not all its previous n-1 words actually take part in its generation. Words can also be generated from their following words or even long-distance words. This is why n-gram models suffer from a severe sparseness problem. Taking ‘as important as’ in this sentence as an example, the second ‘as’ coexists with the first ‘as’ but not generated from ‘important’.

![Diagram](image)

**Fig. 3** A demonstration of the difference among (a) n-gram, (b) dependency Grammar and (c) CG-based structure on sentence generation process
Thus, relative to ‘as...as’, ‘important’ here is quite easy to be replaced by other words, such as ‘fast’, ‘high’, ‘much’ and so on. Consequently, even using 4-gram or 5-gram, sequences consisting of ‘important’ and its nearby words tend to be low-frequent because the connection of ‘as...as’ is interrupted.

By contrast, dependency grammar adopt a predicate-oriented way to determine word sequences, which assumes compliments and adjuncts depend on predicate words, modifiers depend on modified words. The dependency relations are discontinuous so that long distance information can be used for predicting and generating the next word. However, the predicate-oriented assumption doesn’t work accurately in every case. For example, In Figure 3 (b), the period ‘.’ depends on predicate ‘is’. However, ‘.’ appears in every declarative sentence, while not all declarative sentences use ‘is’ as their predicate word, which indicates that period ‘.’ is not actually generated from predicate ‘is’. Similar dependency problem also happens on coordinating conjunction ‘and’. Such kind of inaccuracy is inevitable as long as predicate-oriented assumption is adopted.

At this point, CG-based dependency structure is based on pattern-oriented assumption, which treats schematic patterns as grammatical units and assumes that words are hierarchically generated from certain positions of them. As a result, it can use long distance information to generate and predict words, and words which do not actually take part in the generation can be filtered out from contexts. As shown in Figure 3 (e), ‘is ... as’ is regarded as a pattern context to generate ‘important’, which is closer to our linguistic intuition. Practically, even ‘important’ is replaced by another word, the expression ‘is as...as’ won’t be affected. Furthermore, even ‘is’ is replaced by other verbs, we can still hierarchically back-off to pattern ‘as ... as’, which make it more flexible to the data sparseness problem.

The pattern-oriented assumption also brings another advantage. Different from strictly-defined predicate words, a schematic pattern is a relative concept, which makes it possible to be modeled unsupervisedly. Thus, compared to other structure-based language models, such as class-based language model (Brown, Desouza, Mercer, Pietra, and La 1992), structured language model (Chelba 1997), factored language model (FLM) (Bilmes and Kirchhoff 2003) and dependency tree language model (Shen, Xu, and Weischedel 2008; Chen, Zhang, and Li 2012), the pattern-oriented language model doesn’t use any specific linguistic knowledge or any abstracted categories, which makes it possible to be trained in a totally unsupervised fashion.
3 A Generalized Framework for HWS

3.1 HWS Language Model

As described in Section 2.1, cognitive grammar structure is a hierarchical pattern structure, in which specific expressions are hierarchically generated from more schematic patterns. Thus, in CG-based dependency structure, which is converted from cognitive grammar structure, specific words hierarchically depend on more schematic words. Since the more frequently a word is used, the more probable it becomes part of a pattern, heuristically, we can use the information of word frequency and word order to approximately construct a CG-based dependency structure.

Based on this idea, in (Wu and Matsumoto 2014), we constructed the HWS structure in an unsupervised way as follows:

Step 1: Calculate word frequencies from training data and sort all these words by frequency. Then we can get a frequency-sorted list \( V = \{v_1, v_2, ..., v_m\} \).

Step 2: According to \( V \), for each sentence \( s = w_1, w_2, ..., w_n \), the most frequently used word \( w_i \in s(1 \leq i \leq n) \) is determined.\(^5\) Then use \( w_i \) to split \( s \) into two substrings \( s_l = w_1, ..., w_{i-1} \) and \( s_r = w_{i+1}, ..., w_n \).

Step 3: Set \( s' = s_l \) and \( s'' = s_r \), then repeat Step 2 separately. The most frequently used words \( w_j \in s_l(1 \leq j \leq i-1) \) and \( w_k \in s_r(i+1 \leq k \leq n) \) can also be determined, by which \( s_l \) and \( s_r \) are split into two smaller substrings separately.

Executing Step 2 and Step 3 recursively until all the substrings become empty strings, then a binary tree \( T = (\{w_i, w_j, w_k, \ldots\}, \{(w_i, w_j), (w_i, w_k), \ldots\}) \) can be constructed, which is defined as an HWS structure.

In an HWS structure \( T \), assuming that each node depends on its preceding n-1 parent nodes, then special n-grams can be trained. Such kind of n-grams are defined as HWS-n-grams.

Thus, the HWS language model (Wu and Matsumoto 2014) is essentially a special n-gram model with a different assumption. A normal n-gram model assumes that each word depends on its preceding n-1 words, while an HWS model assumes that each word depends on its n-1 nearby high-frequency words, which can be regarded as an approximation of the pattern-oriented assumption.

3.2 Generalized Priority List

HWS-n-grams are extracted from HWS structures, while HWS structures are constructed by a frequency-sorted vocabulary list \( V = \{v_1, v_2, ..., v_m\} \). Once \( V \) is given, the HWS structures and

\(^5\) If \( w_i \) appears multiple times in \( s \), then select the first one.
HWS-n-grams are deterministic. Thus, the performance of HWS model is totally determined by $V$. $V$ is actually a priority list for constructing HWS structures. Given a sentence $s$, we first judge whether $v_1 \in s$, if not, then we check whether $v_2 \in s$, until we find $v_i \in s$ ($1 \leq i \leq m$) and use it to divide $s$ into two substrings.

Since the priorities of $V$ are only arranged by word frequency, which is independent, it is not sufficient to construct high-quality hierarchical pattern structures because the words constitute a pattern, such as ‘... too ... to ...’, are always dependent and word-order sensitive. For the purpose of improving the construction of HWS structures, we generalize the priority list $V$ with directional context information.

Context $c$ is defined as $a^D$ ($D \in \{L, R, N\}$), where $a$ represents a word and $L, R$ represents the relative word-order direction (left or right) of the words generated from it (which can be discontinuous). We also use a special tag $N$ to represent that $a$ has no directional relations but only priority relations with its following words. We define $ROOT^R$ as the default context where every word generated and $\varepsilon$ as an empty context where no more words generated. Obviously, each priority list starts from $ROOT^R$ and ends with $\varepsilon$. Thus, in original HWS method, priority list $V = \{v_1, v_2, ..., v_m\}$ is actually $\hat{V} = \{ROOT^R, v_1^N, v_2^N, ..., v_m^N, \varepsilon\}$, which is a special case of this generalization.

Suppose word $a_1$ is the highest priority word under $ROOT^R$, then we have three new contexts $\{a_1^L, a_1^R, a_1^N\}$. Under each context, suppose the next highest priority word is $a_2, a_3$ and $a_4$ separately, then we have three branched priority lists start from $\{ROOT^R, a_1^L, a_2\}$, $\{ROOT^R, a_1^R, a_3\}$ and $\{ROOT^R, a_1^N, a_4\}$. Hierarchically, word $a_2, a_3$ and $a_4$ can be treated as new contexts by adding $\{L, R, N\}$ and the priority lists are further branched until no more words generated. Then we can get a multiset\(^6\) of word nodes $M = \{ROOT, a_1, a_2, a_3, a_4, ...\}$ and a set of edges $E = \{(ROOT^R, a_1), (a_1^L, a_2), (a_1^R, a_3), (a_1^N, a_4), ...\}$. Finally, a tree $T = (M, E)$ can be constructed, which is defined as a Node Selection Tree (abbreviated as NST).

A mini NST is shown in Figure 4, given this NST and a substring between ‘as’ and ‘as’, which corresponds to contexts ‘\{as^R, as^L\}’, we first judge whether ‘soon’ is in this substring, if not, then we check ‘well’ in turn. Thus, NST is essentially a decision tree to decide the word priority used for constructing the HWS structures. In an NST, each branch represents the relative position to all its parent nodes, the path from root to node represents hierarchical ‘priority list’.

With the generalized priority list NST, The HWS framework can be generalized as following 4 steps (Figure 5):

\[^6\] As the pattern ‘...as...as...’, the same word are allowed to repeat as different nodes.
Step 1: Construct an NST. Instead of frequency-sorted word list, we construct an Node Selection Tree (NST) and use the whole tree structure as the ‘priority list’ \( V \). We will discuss how to construct an NST from a raw corpus in Section 3.3.

Step 2: Construct HWS structures via NST. Then we use the NST constructed in Step 1 to construct HWS structures from raw sentences. We will introduce how to convert raw sentences into HWS structures by an NST in Section 3.4.

Step 3: Convert HWS structures to HWS-n-grams. Then we can extract HWS-n-grams from both training data and test data via HWS structures. In Section 3.5, we introduce how to extract HWS-n-grams from HWS structures.
Step 4: Build HWS language model. Finally, instead of conventional n-grams, we use HWS-n-grams to build language models.

3.3 Construction of NST

Suppose we are given a set of sentences $S$ for training, NST is constructed by the following steps.

Step 1: Adding ‘⟨ROOT⟩’ (which corresponds to the $ROOT^R$ defined in Section 3.2) to the beginning of each sentence and set it as the default context $c$ (also the root node of NST).

Step 2: For each word $w \in \Omega$ (where $\Omega$ is the vocabulary of $S$), we use a function $score(c, w)$ to decide the word priority under a context $c$.

For the purpose of constructing context-dependent hierarchical pattern structure as we described in Section 3.2, in this paper, we adopt two different word association scores as the function $score(c, w)$: Dice coefficient (Dice 1945) (Equation (1)) and T-score (Equation (2)).

$$score_{DB}(c, w) = \frac{2 \times C(c, w)}{C(c) + C(w)}$$  \hspace{1cm} (1)

$$score_{TB}(c, w) = \left(\frac{C(c, w) - \frac{C(c) \times C(w)}{V}}{V}\right) \div \sqrt{C(c, w)}$$  \hspace{1cm} (2)

Step 3: According to these scores, we choose the word with maximum score $\hat{w}$ as the child node of the root node ‘⟨ROOT⟩’. For each sentence $s \in S$, if $\hat{w} \notin s$, then we put $s$ under the ‘N’ arc of $\hat{w}$, otherwise, we use the first $\hat{w}$ appeared in $s$ to split $s$ into 2 substrings $s_l$ and $s_r$, then put $s_l$ and $s_r$ under the ‘L’ arc and ‘R’ arc of $\hat{w}$ separately. Finally, all strings (or substrings) of $S$ are splitted by three types of arcs of $\hat{w}$.

Step 4: Since each type of arc is branched by ‘L’, ‘R’ and ‘N’, which represents the relative directions from $\hat{w}$, $\hat{w}$ itself can be considered as the context $\hat{w}^L$, $\hat{w}^R$ and $\hat{w}^N$ of each arc separately. Thus, for each arc, we set $\hat{w}$ as new context $c'$, all strings (or substrings) under this arc as a new corpus $S'$. Repeating Step 2 and Step 3 recursively, an NST is constructed.

This recursive process is shown in Algorithm 1.

3.4 Conversion of Raw Sentences into HWS Structures by Using a Constructed NST

With the NST constructed from a corpus, any sentence can be converted into an HWS structure.

Suppose we use the NST shown in Figure 4 to convert the sentence ‘as soon as possible’ into an HWS structure. Firstly, we start from the node ‘as’ and check whether it exists in this
Algorithm 1: Constructing an NST from given corpus $S$

function $\text{GetNode}(c, S')$
  initialize a set $\Omega$ as the vocabulary of $S'$
  $\hat{w} = \text{arg max}_{w_i \in \Omega} (\text{score}(c, w_i))$
  return $\hat{w}$
end function

function $\text{GetTree}(c, S')$
  if $S' = \emptyset$ then return $\emptyset$
  $\hat{w} = \text{GetNode}(c, S')$
  initialize an empty set $L$
  initialize an empty set $R$
  initialize an empty set $N$
  for each $s_i \in S'$ do
    $s_i = w_1, w_2, ..., w_n$ \triangleright split string $s_i$ into words
    if $\hat{w} \notin s_i$ then
      $N = N \cup s_i$
    else
      $j = \text{index of the first } \hat{w} \text{ in } s_i$
      $l = w_1, w_2, ..., w_{j-1}$
      $r = w_{j+1}, ..., w_n$
      $L = L \cup l$
      $R = R \cup r$
    end if
  end for
  initialize a subtree $T'$
  $T' = T' \cup (\hat{w}, 'N', \text{GetTree}(\hat{w}, N))$
  $T' = T' \cup (\hat{w}, 'L', \text{GetTree}(\hat{w}, L))$
  $T' = T' \cup (\hat{w}, 'R', \text{GetTree}(\hat{w}, R))$
  return $T'$
end function

initialize a tree $T$
$T = T \cup ('\text{ROOT}', 'R', \text{GetTree}(\text{ROOT}', S))$
return $T$

sentence. Although two ‘as’s are observed in this sentence, we use the first one to divide it and take the right substring ‘soon as possible’ to the ‘R’ arc. Since the child node ‘as’ of ‘R’ arc can also be observed in this substring, ‘soon’ and ‘possible’ are classified to its ‘L’ arc and ‘R’ arc respectively. Finally, ‘as soon as possible’ is converted to the HWS structure as shown in Figure 6. Similarly, sentence ‘as fast as possible’, ‘as well as me’ can also be converted into HWS structures by this NST.
Notice that even a well-trained NST cannot cover all possible situations. For instance, suppose we use the above NST to convert sentence ‘as far as I could remember’, although ‘I could remember’ is correctly classified as ‘R’ arc of the second ‘as’ node, it cannot be further analyzed because this NST doesn’t offer more details in this arc. In this case, we use the original HWS approach (by word frequency) to select nodes from substring ‘I could remember’ as a covering of our method. This process is shown in Algorithm 2.

3.5 Conversion of HWS Structures into HWS-n-grams

For the root node of an HWS structure, we define that it depends on symbol ‘⟨s⟩’. We also use the symbol ‘⟨/s⟩’ to represent the end of generation. Then we can achieve special n-grams under the our framework. Taking the HWS structure of Figure 6 as the example, we can train 3-grams like \{⟨s⟩, ⟨s⟩, as⟩, ⟨s⟩, as, as⟩, ⟨s⟩, as, possible⟩, ⟨s⟩, as, possible, ⟨/s⟩⟩, ⟨s⟩, as, soon⟩, ⟨s⟩, soon⟩, ⟨/s⟩⟩.}

In (Wu and Matsumoto 2015), we verified that the performance of hierarchical word sequence language model can be further improved by using directional information. Thus, in this paper, we defaultly use directional information to model word sequences. Then the above 3-grams should be \{⟨s⟩, ⟨s⟩, as⟩, ⟨s⟩, as-L, ⟨/s⟩⟩, ⟨s⟩, as-R, as⟩, ⟨s⟩, as-R, as-L, ⟨/s⟩⟩, ⟨s⟩, as-L, soon-R, ⟨/s⟩⟩, ⟨s⟩, as-R, possible⟩, ⟨s⟩, as-R, possible-L, ⟨/s⟩⟩, ⟨s⟩, as-R, possible-R, ⟨/s⟩⟩ and the probability of the whole sentence ‘as soon as possible’ can be estimated by the product of conditional probabilities of all these word sequences.
Algorithm 2 Converting a sentence $s$ into HWS structure using an NST $T$

function $\text{toHWS}(T', s')$

if $s' = \emptyset$ then return $\emptyset$

end if

if $T' = \emptyset$ then
    return $\text{ConvertByFrequency}(s')$

end if

$r = \text{the root node of } T'$

if $r \in s'$ then

    $j =$ index of the first $r$ in $s'$
    $lStr = w_1, w_2, ..., w_{j-1}$
    $rStr = w_{j+1}, ..., w_n$
    $lTree = T'[r][\text{L}]$
    $rTree = T'[r][\text{R}]$

    return $(\text{toHWS}(lTree, lStr), r, \text{toHWS}(rTree, rStr))$

else

    $nTree = T'[r][\text{N}]$

    return $\text{toHWS}(nTree, s')$

end if

done function

$s = w_1, w_2, ..., w_n$ \hspace{1cm} \triangleright \text{split sentence } s \text{ into words}$

$hws = \text{toHWS}(T, s)$

return $hws$

4 Intrinsic Evaluation

4.1 Settings

We use two different corpora: British National Corpus and English Gigaword Corpus.

British National Corpus (BNC)\(^7\) is a 100 million word collection of samples of written and spoken English from a wide range of sources. We use all the 6,052,202 sentences (100 million words, 629,881 different types) for the training data.

English Gigaword Corpus\(^8\) consists of over 1.7 billion words of English newswire from 4 distinct international sources. We choose the $wpb\_eng$ part (162,099 sentences, 20 million words, 181,523 different types, OOV = 78,854) for the test data.

As preprocessing of the training data and the test data, we use the tokenizer of NLTK (Natural Language Toolkit)\(^9\) to split raw English sentences into words. We also converted all words to

\(^7\) http://www.natcorp.ox.ac.uk
\(^8\) https://catalog.ldc.upenn.edu/LDC2011T07
\(^9\) http://www.nltk.org
In (Wu and Matsumoto 2014), we use coverage score to perform evaluation. The word sequences modeled from training data are defined as TR, while that of test data as TE, then the coverage score is calculated by Equation (3). Obviously, the higher coverage score a language model can achieve, the more it can relieve the data sparseness problem (reduce the unseen sequences).

\[
\text{score}_{\text{coverage}} = \frac{|TR \cap TE|}{|TE|} \tag{3}
\]

If we enumerate all possible word combinations as word sequences, then we can achieve considerable coverage score. However, the processing efficiency of a model become extremely low. Thus, we also use usage score (Equation (4)) to estimate how much redundancy is contained in a model.

\[
\text{score}_{\text{usage}} = \frac{|TR \cap TE|}{|TR|} \tag{4}
\]

A balanced measure between coverage and usage is calculated by Equation (5).

\[
F\text{-Score} = \frac{2 \times \text{coverage} \times \text{usage}}{\text{coverage} + \text{usage}} \tag{5}
\]

As intrinsic evaluation of language modeling, perplexity (Manning and Schütze 1999) is also a common metric used for measuring the usefulness of a language model. In the following section, we also use perplexity to compare our models with other models.

4.2 Results

We compare word sequences modeled under our framework with conventional n-gram sequences.

First, we evaluate coverage and usage on unique word sequences, which means we count each word sequence only once in spite of the amount of times it really occurs. The result is shown in Table 1.

According to the results, the effectiveness and efficiency of NST-based bi-grams is nearly the same as that of normal bi-grams. But as for tri-grams, each NST-based method improves a lot on both coverage and usage.

We also execute the same experiment on total word sequences, which actually affect the final performance of language models. The result is shown in Table 2. The coverage and usage of NST-based unigrams are a little larger than normal unigrams because NST-based methods are not continuous and consequently has more ending symbol ‘(/s)’s. Different from unique
Table 1  Coverage and Usage on Unique Word Sequences

<table>
<thead>
<tr>
<th>Models</th>
<th>Coverage (%)</th>
<th>Usage (%)</th>
<th>F-score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uni-gram</td>
<td>56.560</td>
<td>16.300</td>
<td>25.307</td>
</tr>
<tr>
<td>frequency-based-uni</td>
<td>56.560</td>
<td>16.300</td>
<td>25.307</td>
</tr>
<tr>
<td>dice-based-uni</td>
<td>56.560</td>
<td>16.300</td>
<td>25.307</td>
</tr>
<tr>
<td>tscore-based-uni</td>
<td>56.560</td>
<td>16.300</td>
<td>25.307</td>
</tr>
<tr>
<td>bi-gram</td>
<td><strong>46.471</strong></td>
<td><strong>12.015</strong></td>
<td><strong>19.093</strong></td>
</tr>
<tr>
<td>frequency-based-bi</td>
<td>46.137</td>
<td>10.225</td>
<td>16.74</td>
</tr>
<tr>
<td>dice-based-bi</td>
<td>46.136</td>
<td>9.273</td>
<td>15.442</td>
</tr>
<tr>
<td>tscore-based-bi</td>
<td>46.435</td>
<td>11.387</td>
<td>18.288</td>
</tr>
<tr>
<td>tri-gram</td>
<td>27.164</td>
<td>5.626</td>
<td>9.321</td>
</tr>
<tr>
<td>frequency-based-tri</td>
<td>35.877</td>
<td>7.265</td>
<td>12.083</td>
</tr>
<tr>
<td>dice-based-tri</td>
<td>36.100</td>
<td>7.068</td>
<td>11.821</td>
</tr>
<tr>
<td>tscore-based-tri</td>
<td><strong>36.820</strong></td>
<td><strong>8.031</strong></td>
<td><strong>13.186</strong></td>
</tr>
</tbody>
</table>

Table 2  Coverage and Usage on Total Word Sequences

<table>
<thead>
<tr>
<th>Models</th>
<th>Coverage (%)</th>
<th>Usage (%)</th>
<th>F-score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uni-gram</td>
<td>97.264</td>
<td>96.261</td>
<td>96.76</td>
</tr>
<tr>
<td>frequency-based-uni</td>
<td>98.577</td>
<td>98.039</td>
<td>98.307</td>
</tr>
<tr>
<td>dice-based-uni</td>
<td>98.577</td>
<td>98.039</td>
<td>98.307</td>
</tr>
<tr>
<td>tscore-based-uni</td>
<td>98.577</td>
<td>98.039</td>
<td>98.307</td>
</tr>
<tr>
<td>bi-gram</td>
<td>83.121</td>
<td>76.336</td>
<td>79.584</td>
</tr>
<tr>
<td>frequency-based-bi</td>
<td><strong>90.034</strong></td>
<td>85.619</td>
<td>87.771</td>
</tr>
<tr>
<td>dice-based-bi</td>
<td>89.939</td>
<td>85.296</td>
<td>87.556</td>
</tr>
<tr>
<td>tscore-based-bi</td>
<td>89.969</td>
<td><strong>86.617</strong></td>
<td><strong>88.261</strong></td>
</tr>
<tr>
<td>tri-gram</td>
<td>51.151</td>
<td>40.191</td>
<td>45.013</td>
</tr>
<tr>
<td>frequency-based-tri</td>
<td>72.298</td>
<td>63.321</td>
<td>67.512</td>
</tr>
<tr>
<td>dice-based-tri</td>
<td>72.118</td>
<td>61.613</td>
<td>66.453</td>
</tr>
<tr>
<td>tscore-based-tri</td>
<td><strong>72.548</strong></td>
<td><strong>65.403</strong></td>
<td><strong>68.790</strong></td>
</tr>
</tbody>
</table>

sequence experiment, even in bi-grams, NST-based methods have much better performance than the normal bi-gram model. As for tri-grams, the NST-based methods can even improve around 25%.

We also use different portions of different sizes of BNC corpus. We gradually increase the amount of training data to examine how it affects the F-scores of word sequences. As show in Figure 7, all strategies improve along with the increasing of training data size. Also, all those methods increase at almost the same speed, even we increase the training data size to 100 million words, NST-based methods still have a great advantage over the normal n-gram models.

We also compare HWS to other models. For the purpose of comparing it with gold standard
dependency grammar, we use the English part of CoNLL2007 shared task for the data set.\(^\text{10}\) Dependency Grammar structure is compatible with HWS except it allows one-to-many dependency relations. Thus, given a English dependency structure shown in Figure 8, we can convert it into 3-gram word sequences with directional information as \{ (has-L, date-L, a), (date-L, a-L, ⟨/s⟩), (date-L, a-R, ⟨/s⟩), (has-L, date-L, record), (date-L, record-L, ⟨/s⟩), (date-L, record-R, ⟨/s⟩), ((⟨s⟩, has-L date), (has-L, date-R, ⟨/s⟩)), ((⟨s⟩, has), ((⟨s⟩, has-R n’t), (has-R n’t-L, ⟨/s⟩)), (has-

\[\begin{array}{c}
\text{Fig. 7} \quad \text{The Increasing of F-scores on Total Word Sequences with the Addition of Training Data Size}
\end{array}\]

\[\begin{array}{c}
\text{Fig. 8} \quad \text{An example of English dependency structure (from CoNLL2007 shared task)}
\end{array}\]

\(^{10}\) The types of words of training data is 26,600, while that of test data is 1,373. OOV = 107.
R n’t-R, ⟨s⟩, has-R, been), (has-R, been-L, ⟨s⟩), (has-R, been-R, set), (been-R, set-L, ⟨s⟩), (been-R, set-R, ⟨s⟩), (⟨s⟩, has-R, .), (has-R, -.L, ⟨s⟩), (has-R, -.R, ⟨s⟩)\}. The results of coverage and usage are shown in Table 3. For unique grams, our bi-gram model (db-based-bi) outperforms dependency grammar based bi-gram model, while its trigram model outperforms ours. For total grams, the results swap. We can also find that skip-gram models do improve coverage at the sacrifice of usage, but even we skip 3 words, proposed models still outperform skip-n-gram models on coverage.

Besides the coverage and usage, we also compare our model by evaluating other measures, such like occupied memory, processing time and perplexity. The results are summarized in Table 4.

Occupied memory: Compared to conventional trigram model, the size of skip-3-gram models grow fast along with the increase of skip words. Since proposed model and dependency-based model adopt the directional information, which means almost each word \( w \) are treated as two words \( w\)-L and \( w\)-R, their sizes are nearly the same as that of 1skip-3-gram model. But for

### Table 3 Coverage and Usage on English CoNLL2007 Shared Task Data Set (Unique/Total)

<table>
<thead>
<tr>
<th>Models</th>
<th>Coverage (%)</th>
<th>Usage (%)</th>
<th>F-score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bi-gram</td>
<td>63.260/71.574</td>
<td>1.309/22.899</td>
<td>2.566/34.697</td>
</tr>
<tr>
<td>1skip-bi-gram</td>
<td>67.585/74.890</td>
<td>0.683/14.567</td>
<td>1.353/24.390</td>
</tr>
<tr>
<td>2skip-bi-gram</td>
<td>69.439/76.327</td>
<td>0.481/11.680</td>
<td>0.956/20.260</td>
</tr>
<tr>
<td>3skip-bi-gram</td>
<td>71.087/77.688</td>
<td>0.385/10.514</td>
<td>0.766/18.522</td>
</tr>
<tr>
<td>db-based-bi</td>
<td><strong>72.348</strong>/83.591</td>
<td><strong>1.737</strong>/43.091</td>
<td><strong>3.393</strong>/56.867</td>
</tr>
<tr>
<td>dependency-bi</td>
<td>69.124/84.195</td>
<td>1.645/46.722</td>
<td>3.213/60.096</td>
</tr>
<tr>
<td>trigram</td>
<td>29.805/32.960</td>
<td>0.416/3.968</td>
<td>0.820/7.083</td>
</tr>
<tr>
<td>1skip-tri-gram</td>
<td>31.372/34.459</td>
<td>0.217/2.213</td>
<td>0.432/4.159</td>
</tr>
<tr>
<td>2skip-tri-gram</td>
<td>32.388/35.419</td>
<td>0.152/1.605</td>
<td>0.303/3.071</td>
</tr>
<tr>
<td>3skip-tri-gram</td>
<td>33.235/36.238</td>
<td>0.120/1.311</td>
<td>0.240/2.531</td>
</tr>
<tr>
<td>db-based-tri</td>
<td>43.024/52.658</td>
<td>0.659/17.580</td>
<td>1.298/26.360</td>
</tr>
<tr>
<td>dependency-tri</td>
<td><strong>43.599</strong>/52.159</td>
<td><strong>0.708</strong>/12.738</td>
<td><strong>1.394</strong>/20.476</td>
</tr>
</tbody>
</table>

### Table 4 Other Performance on English CoNLL2007 Shared Task Data Set

<table>
<thead>
<tr>
<th>Models</th>
<th>Memory (mb)</th>
<th>Training + Evaluation Time (ms)</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>trigram</td>
<td><strong>39.6</strong></td>
<td>2,730 + 10,101</td>
<td>208.171</td>
</tr>
<tr>
<td>1skip-tri-gram</td>
<td>68.5</td>
<td>5,866 + 18,668</td>
<td>185.490</td>
</tr>
<tr>
<td>2skip-tri-gram</td>
<td>95.3</td>
<td>7,251 + 26,033</td>
<td>178.078</td>
</tr>
<tr>
<td>3skip-tri-gram</td>
<td>119.5</td>
<td>9,649 + 36,874</td>
<td>86.120</td>
</tr>
<tr>
<td>db-based-tri</td>
<td>4.6 + 67.7</td>
<td>14,754 + 3,287 + 16,351</td>
<td>57.761</td>
</tr>
<tr>
<td>dependency-tri</td>
<td>71.4</td>
<td>dependency parsing + 17,828</td>
<td><strong>36.227</strong></td>
</tr>
</tbody>
</table>
proposed method, we have to use an extra memory to store the NST tree.

Processing time: We use a 3 GHz Intel Core i7, 16 GB 1600 MHz DDR3 computer to perform the experiment. For the time of conversion raw sentences into sequences, conventional n-gram is the fastest method, only costs 2,730 ms. Also, due to its smallest model size, its training time (MKN smoothing) only costs 10,101 ms too. Compared to skip-3-gram, proposed model performs much faster either on sequence conversion or smoothing, but additionally, it needs an extra step to train the NST from the training data, which is time-consuming.

Perplexity: Compare to conventional 3-gram model and skip-3-gram model, proposed model can reduce the perplexity largely. The dependency grammar based model outperforms our model slightly, but we use gold standard parsing results for the dependency grammar based model while using a totally unsupervised method for ours.

To clarify this, we compare our model to the generative dependency n-gram language model, which is proposed by (Ding and Yamamoto 2013). In this paper, they propose an unsupervised method to construct dependency structures (generative dependency n-gram language model), by which parameters are estimated. For the datasets and the performance of their model, we use exactly the same as they described in their paper. The comparison results of perplexities are shown in Table 5. Same as unsupervised methods, our model greatly outperforms the generative dependency n-gram language model. It is also verified that our model performs well on multiple languages.

5 Extrinsic Evaluation

For the purpose of examining how our models work in the real world application, we also performed extrinsic experiments to evaluate our method. In this paper, we use the reranking of n-best translation candidates to examining how language models work in a statistical machine translation task.
5.1 Settings

We use the French-English part of TED talk parallel corpus\textsuperscript{11} for the experiment dataset. The training data contains 139,761 sentence pairs, while the test data contains 1,617 sentence pairs. For training language models, we set English as the target language.

As for statistical machine translation toolkit, we use Moses system\textsuperscript{12} to train the translation model and output 50-best translation candidates for each French sentence of the test data. Then we use 139,761 English sentences to train language models. With these models, 50-best translation candidates are reranked. According to these reranking results, the performance of machine translation system is evaluated, which also means, the language models are evaluated indirectly. In this paper, we use the following measures for evaluating reranking results.

**BLEU** (Papineni, Roukos, Ward, and Zhu 2002): BLEU score measures how many words overlap in a given candidate translation when compared to a reference translation, which provides some insight into how good the fluency of the output from an engine will be.

**METEOR** (Banerjee and Lavie 2005): METEOR score computes a one-to-one alignment between matching words in a candidate translation and a reference.

**TER** (Snover, Dorr, Schwartz, Micciulla, and Makhoul 2006): TER score measures the number of edits required to change a system output into one of the references, which gives an indication as to how much post-editing will be required on the translated output of an engine.

We use open source tool multeval\textsuperscript{13} to perform the evaluation.

5.2 Results

We use various association scores to perform experiments and compared them to the normal n-gram strategy. For estimating the probabilities of translation candidates, we use the modified Kneser-Ney smoothing (MKN\textsuperscript{14}) as the smoothing method of all strategies. As shown in Table 6, NST-based strategies outperform that of n-gram on each score.

To examine our methods on other languages, we also perform the same experiment on Spanish-English, Japanese-English and Chinese-English dataset. As shown in Table 7, for each language pair, NST-based strategies still outperform n-gram strategy on all the three measures, especially with an obvious improvement on TER score.

\textsuperscript{11} Which can also be downloaded at https://github.com/aisophie/HWS.
\textsuperscript{12} http://www.statmt.org/moses/
\textsuperscript{13} https://github.com/jhclark/multeval
\textsuperscript{14} The technical detail of MKN is shown in the Appendix.
Table 6  Performance on French-English SMT Task Using Various Word Arranging Strategies

<table>
<thead>
<tr>
<th>Models</th>
<th>BLEU</th>
<th>METEOR</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>tri-gram</td>
<td>31.3</td>
<td>33.5</td>
<td>49.0</td>
</tr>
<tr>
<td>frequency-based-tri</td>
<td>31.7</td>
<td>33.5</td>
<td>48.5</td>
</tr>
<tr>
<td>dice-based-tri</td>
<td>31.8</td>
<td>33.6</td>
<td>48.3</td>
</tr>
<tr>
<td>tscore-based-tri</td>
<td>31.7</td>
<td>33.6</td>
<td>48.4</td>
</tr>
</tbody>
</table>

Table 7  Performance on Spanish-English, Japanese-English and Chinese-English SMT Task Using Various Word Arranging Strategies

**Spanish-English**

<table>
<thead>
<tr>
<th>Models</th>
<th>BLEU</th>
<th>METEOR</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>tri-gram</td>
<td>31.8</td>
<td>34.9</td>
<td>48.9</td>
</tr>
<tr>
<td>frequency-based-tri</td>
<td>32.1</td>
<td>34.9</td>
<td>48.3</td>
</tr>
<tr>
<td>dice-based-tri</td>
<td>32.1</td>
<td>34.9</td>
<td>48.2</td>
</tr>
<tr>
<td>tscore-based-tri</td>
<td>32.3</td>
<td>34.9</td>
<td>48.1</td>
</tr>
</tbody>
</table>

**Japanese-English**

<table>
<thead>
<tr>
<th>Models</th>
<th>BLEU</th>
<th>METEOR</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>tri-gram</td>
<td>7.4</td>
<td>19.2</td>
<td>87.6</td>
</tr>
<tr>
<td>frequency-based-tri</td>
<td>7.8</td>
<td>19.2</td>
<td>86.2</td>
</tr>
<tr>
<td>dice-based-tri</td>
<td>7.6</td>
<td>19.2</td>
<td>86.1</td>
</tr>
<tr>
<td>tscore-based-tri</td>
<td>7.8</td>
<td>19.2</td>
<td>86.0</td>
</tr>
</tbody>
</table>

**Chinese-English**

<table>
<thead>
<tr>
<th>Models</th>
<th>BLEU</th>
<th>METEOR</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>tri-gram</td>
<td>12.4</td>
<td>22.2</td>
<td>76.8</td>
</tr>
<tr>
<td>frequency-based-tri</td>
<td>12.7</td>
<td>22.2</td>
<td>75.9</td>
</tr>
<tr>
<td>dice-based-tri</td>
<td>12.5</td>
<td>22.1</td>
<td>76.0</td>
</tr>
<tr>
<td>tscore-based-tri</td>
<td>12.6</td>
<td>22.1</td>
<td>76.0</td>
</tr>
</tbody>
</table>

5.3 Analasis

For the purpose of clarifying the effect of the proposed model, we take below example to reveal how reranking task benefits from HWS.

The best candidate outputted by n-gram model is “and he has around 70’s .”, while the result given by HWS model is “and there are about 70 of them .”, which is a better translation.

The reason why n-gram model select “and he has around 70’s .” is that ‘70’ is an OOV and n-gram estimate sentence probabilities in a continuous way. Consequently, even ‘there are about ⟨NUMBER⟩ of them’ is quite a common expression, its probability is still assigned much smaller.

On the other hand, HWS estimates the probability of this sentence by the following sequences.
{(⟨s⟩, .), (⟨s⟩, -L, are), (-L, are-L, there), (are-L, there-L, and), (there-L, and-L, ⟨/s⟩),
(there-L, and-R, ⟨/s⟩), (are-L, there-R, ⟨/s⟩), (-L, are-R, of), (are-R, of-L, about), (of-L, about-L, ⟨/s⟩),
(of-L, about-R, 70), (about-R, 70-L, ⟨/s⟩), (about-R, 70-R, ⟨/s⟩), (are-R, of-R, them),
(of-R, them-L, ⟨/s⟩), (of-R, them-R, ⟨/s⟩), (⟨s⟩, -R, ⟨/s⟩)}

Among these sequences, patterns such like “... there are ... .”, “... are ... of ... .”, which
to repeatedly appear in the training data, are assigned bigger possibilities. And sequences including
OOV ‘70’, such like (of-L, about-R, 70), are actually calculated as how likely an unknown word
generated from pattern “about ... of”. Also, unlike n-gram models, this OOV won’t affect ‘them’
at all because ‘them’ is generated from pattern “are ... of”. ...

Compare to conventional n-gram models, HWS models make it possible to take advantage of
repeated patterns (including long distance ones) for word prediction and probability estimation.
As a result, HWS models select more natural translation candidate as the output.

6 Conclusion

In this paper, inspired by the basic idea and structure of cognitive grammar, we proposed a
generalized hierarchical word sequence framework for language modeling. Under this framework,
we adopt different kinds of association scores for rearranging word sequences.

For evaluation, we compared our rearranged word sequences to conventional n-gram word
sequences and performed intrinsic and extrinsic experiments. The intrinsic experiment proved
that our methods can greatly relieve the data sparseness problem, while the extrinsic experiments
proved that SMT tasks can benefit from our strategies. Both verified that language modeling
can achieve better performance by using our word sequences rearranging strategies.

But on the other hand, our model needs an extra process for training NST, also an extra space
for storing it, which is relatively more time-consuming and memory-consuming than conventional
n-gram models. Also, since HWS structure has to be constructed after reading the whole sentence,
it is not appropriate to apply it in some instant applications, such like speech recognition. Despite
these two disadvantages, we recommend to replace conventional n-grams with HWS-n-grams
under all kinds of NLP applications.

Further, instead of conventional n-gram word sequences, our rearranged word sequences can
also be used as the features of various kinds of machine learning approaches, which is an interesting
future study.
Reference


Appendix

[Modified Kneser-Ney Smoothing]

The state-of-the-art method for smoothing is modified Kneser-Ney smoothing proposed in (Chen and Goodman 1999). Based on normal Kneser-Ney smoothing, MKN uses different discount parameters for non-zero counts, whose calculation is shown as Equation (6).

\[
P_{MKN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{C(w_{i-n+1}^i) - D(C(w_{i-n+1}^i)), 0\}}{C(w_{i-n+1}^i)} + \gamma_{high}(w_{i-n+1}^{i-1})\hat{P}_{MKN}(w_i|w_{i-n+2}^{i-1})
\]

The discount value \( D \) is a discount value calculate by Equation (7).\(^{15}\)

\[
D(c) = \begin{cases} 
0 & \text{if } c = 0 \\
D_1 = 1 - 2\frac{n_1 + n_2}{n_1} & \text{if } c = 1 \\
D_2 = 2 - 3\frac{n_1 + n_2}{n_1 + n_2} & \text{if } c = 2 \\
D_{3+} = 3 - 4\frac{n_1 + n_2}{n_1 + n_3} & \text{if } c > 2 
\end{cases}
\]

And \( \gamma_{high}(w_{i-n+1}^{i-1}) \) is defined as Equation (8).\(^{16}\)

\[
\gamma_{high}(w_{i-n+1}^{i-1}) = \frac{D_1N_1(w_{i-n+1}^{i-1}) + D_2N_2(w_{i-n+1}^{i-1}) + D_{3+}N_{3+}(w_{i-n+1}^{i-1})}{C(w_{i-n+1}^{i-1})}
\]

The lower order models \( \hat{P}_{MKN}(w_i|w_{i-n+1}^{i-1}) \) are interpolated recursively as below.

\[
\hat{P}_{MKN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{N_{1+}(w_{i-n+1}^{i-1}) - D(C(w_{i-n+1}^i)), 0\}}{N_{1+}(w_{i-n+1}^{i-1})} + \gamma_{mid}(w_{i-n+1}^{i-1})\hat{P}_{MKN}(w_i|w_{i-n+2}^{i-1})
\]

where \( \gamma_{mid}(w_{i-n+1}^{i-1}) \) is defined as Equation (10).

\[
\gamma_{mid}(w_{i-n+1}^{i-1}) = \frac{D_1N_1(w_{i-n+1}^{i-1}) + D_2N_2(w_{i-n+1}^{i-1}) + D_{3+}N_{3+}(w_{i-n+1}^{i-1})}{N_{1+}(w_{i-n+1}^{i-1})}
\]

---

\(^{15}\) \( n_i \) is the total number of n-grams which appear exactly \( i \) times in the training data.

\(^{16}\) \( N_1(w_{i-n+1}^{i-1}) = |\{w_i : C(w_{i-n+1}^i) = 1\}| \), and \( N_2(w_{i-n+1}^{i-1}) \) \( N_{3+}(w_{i-n+1}^{i-1}) \) are counted in a similar way.
Xiaoyi Wu: received his B.S. degree from Liaoning University in 2003 and M.S. degree from Beijing Foreign Studies University in 2011. He is currently a doctoral student at NAIST. His research interests include linguistics and natural language processing.

Kevin Duh: received the B.S. degree from Rice University, Houston, TX, USA, in 2003, and Ph.D. from the University of Washington, Seattle, WA, USA, in 2009, both in electrical engineering. He was working as an Assistant Professor at the Nara Institute of Science and Technology (NAIST), Graduate School of Information Science, Ikoma, Japan. Before joining NAIST, from 2009 to 2012, he worked at the NTT Communication Science Laboratories. His research interests include intersection of natural language processing and machine learning, in particular, in areas relating to machine translation and deep learning.

Yuji Matsumoto: received his M.S. and Ph.D. degrees in information science from Kyoto University in 1979 and in 1989. He joined Machine Inference Section of Electrotechnical Laboratory in 1979. He has then experienced an academic visitor at Imperial College of Science and Technology, a deputy chief of First Laboratory at ICOT, and an associate professor at Kyoto University. He is currently a professor at the Graduate School of Information Science, Nara Institute of Science and Technology. His main research interests are natural language understanding and machine learning.

(Received September 17, 2016)
(Revised December 7, 2016)
(Rerevised February 13, 2017)
(Accepted March 31, 2017)