NAIST-IS-DD1361024

Doctoral Dissertation

Hierarchical Word Sequences Structures for Language Modeling

Xiaoyi Wu

June 15, 2017

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Xiaoyi Wu

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Satoshi Nakamura	(Co-supervisor)
Associate Professor Masashi Shimbo	(Co-supervisor)
Assistant Professor Hiroyuki Shindo	(Co-supervisor)
Assistant Professor Hiroshi Noji	(Co-supervisor)
Associate Professor Kevin Duh	(Johns Hopkins University)

Hierarchical Word Sequences Structures for Language Modeling*

Xiaoyi Wu

Abstract

Language modeling is a fundamental research problem that has wide application for many NLP tasks. The most important role of language modeling is estimating the probabilities of sentences of natural language. 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. Thus, even using large-scale corpus and high performance smoothing methods such like MKN, 3-grams cannot be distinguished well, which affects the final performance of language models.

In this dissertation, based on the basic idea of cognitive grammar structure, we propose a hierarchical word sequence structure, where different assumptions can be adopted to rearrange word sequences in a totally unsupervised fashion. We present three different methods (assumption-based, NST-based, and neural network-based) to construct the hierarchical word sequence structures. 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. The intrinsic experiment proved that our methods can greatly

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1361024, June 15, 2017.

relieve the data sparseness problem, while the extrinsic experiments proved that SMT tasks can benefit from our strategies. Both prove that our method can be considered as a better alternative for n-gram language models.

Keywords:

language modeling, word sequence, sparseness problem, cognitive grammar, n-gram model, hierarchical word sequence model, neural network language model

階層的語序列構造に基づく言語モデリング*

呉 暁一

内容梗概

言語モデリングは自然言語処理の基礎研究であり幅広く多くの技術に応用されている。言語モデルの最も重要な役割は自然言語の文の確率を推定することである。文の確率を推定するため，ほとんどの言語モデリングに関する先行研究は n-gram に基づく手法を用いている。しかし，n-gram 言語モデルの仮説はロバスト性が不足しており，データスパースネスの問題が深刻である。たとえ大規模コーパスおよび MKN のような高性能なスムージング手法を用いても，3-gram に基づいたものであっても，言語モデルの性能に大きな悪影響を与えている。

本論文では，認知文法 (Cognitive Grammar) 構造の基本概念に基づき，階層的語序列 (Hierarchical Word Sequence, HWS) 構造を提案する。この構造に基づき，色々な仮説を用い，特定の言語知識に依存せず，完全に教師なしで語序列を再配列することが可能である。更に，この HWS 構造を構築するため,「仮説指向」,「NST 指向」,「ニューラルネットワーク指向」の三つの異なる手法を提案する。n-gram 言語モデルが文の左側から右側へ確率を推定するのに比べて，HWS はより柔軟的な策略を用いる。

評価として再配列した語序列と伝統的な n-gram 語序列を直接実験と間接実験で比較した結果，提案モデルの方はより高い性能が得られ，データスパースネス問題を軽減すると同時に，機械翻訳など自然言語処理応用にもさらなる精度をもたらすことがわかった。これにより，HWS が n-gram 言語モデルの代用になることを示すことができた。

*奈良先端科学技術大学院大学 情報科学研究科 博士論文, NAIST-IS-DD1361024, 2017 年 6 月 15 日.

キーワード

言語モデリング, 語序列, スパース問題, 認知文法, n-gram 言語モデル, 階層的語序列言語モデル、ニューラルネットワーク言語モデル

目 次

図 目 次

表 目 次

1. Introduction

1.1 Background

Probabilistic Language Modeling is a fundamental research direction of Natural Language Processing. It is widely used in various application such as machine translation [7], spelling correction [21], speech recognition [25], word prediction [4] and so on.

Theoretically, the more language model can reflect the real process of sentence generation, the more NLP applications can benefit from it. That's the reason why language modeling plays an important role in all kinds of NLP applications.

The most widely used language model is n-gram language model [26], which assumes that in one sentence, following words depends on their preceding words. However, this assumption is to simple to reflect the true sentence generation process, which causes a severe sparseness problem.

To relieve the sparseness problem caused by n-gram approach, many smoothing methods, such as Katz back-off [15], Kneser-Ney [16], and modified Kneser-Ney [9] has been developed. However, they all take the n-gram approach 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 [1], which cannot be distinguished accurately even using a high-performance 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 [13], skip-gram [14][1] is proposed to overcome the data sparseness problem. For each n-gram word sequence, the skip-gram model enumerates all possible

[1]The k-skip-n-grams for a sentence $w_1, ...w_m$ is defined as the set $\{w_{i_1}, w_{i_2}, ...w_{i_n} | \Sigma_{j=1}^{n} i_j - i_{j-1} < k\}$.

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.

There are also many structured language models, such as class-based language model [6], structured language model [8], factored language model(FLM) [5] and dependency tree language model [27] [10] have been developed. However, most of structured language models depend on specific linguistic knowledge and the precision of other NLP tasks, which also increases the computational complexity.

In [29], 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 [30] [31], 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.

In this dissertation, inspired by the view point of sentence generation process from cognitive grammar, proposed by Langacker [17] [18] [19], we propose a generalized hierarchical word sequence structure, by which various strategies of rearranging word sequences can be used for language modeling. We present three different methods to achieve this structure. We also use both intrinsic and extrinsic experiments to verify the effectiveness of the three methods and these strategies.

1.2 Contributions of This Research

The dissertation makes the following scientific contributions:

1) We present a generalized hierarchical word sequence structure, extending our previous work on specific HWS methods, which makes proposed method more potential for further improving.

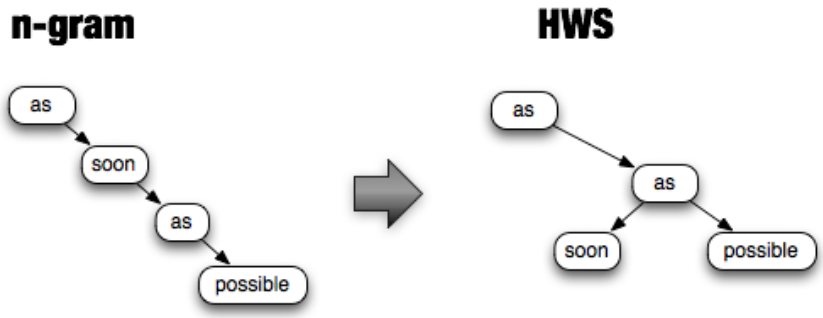

Figure 1. Rearranged Word Sequences Compared to n-gram Word Sequences

2) We adopt the basic idea of cognitive grammar structure as the theoretical linguistic basis of proposed structure, which makes proposed method linguistically supported. Meanwhile, it can also be regarded as an approximate implement of cognitive grammar structure for natural language processing.

3) Under this generalized structure, we present various strategies for rearranging word sequences and empirically verify their better performance in comparison to n-grams in language modeling (Figure 1), which proves that proposed method can be considered as a better alternative for n-gram method.

1.3 Dissertation Outlines

This disseration is organized as follows.

Chapter 2 Cognitive Grammar and Its Advantage First, as the linguistic basis of our structure and methods, we introduce the basic idea of cognitive grammar structure and its advantage in language modeling.

Chapter 3 Hierarchical Word Sequence Structure First, we give a complete review of the original hierarchical word sequence language models, which are inspired by the basic idea of Cognitive Grammar. Then we generalize those models into one unified structure under a formal representation. To achieve this structure, we present three different unsupervised methods in the following three

chapters.

Chapter 4 Method One: Assumption Based Construction of HWS Structure The first method is using certain assumption to achieve HWS structures.

Chapter 5 Method Two: Node Selection Tree Based Construction of HWS Structure The second one is training a Node Selection Tree (NST) from the training data first, then use it to convert sentences into HWS structures.

Chapter 6 Method Three: Neural Network Based Construction of HWS Structure The last one is training a neural network from the training data first, then use it to convert sentences into HWS structures.

Chapter 7 Intrinsic Evaluation For intrinsic evaluation, we use various experiments to verify that HWS-n-grams have advantage on reducing unseen sequences and consequently relieve the data sparseness problem.

Chapter 8 Extrinsic Evaluation For extrinsic evaluation, we use sentence reranking to verify that SMT tasks can also benefit more from HWS-n-grams than conventional n-grams.

Chapter 9 Conclustion Finally, we summarize our findings and describe our possible future directions.

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 ([17] [18] [19]). In cognitive grammar, it is presumed that complex expressions are formed from schematic patterns, such as 'V_s X in the N_b' [2] (e.x. hit him in the belly) and 'a N_1+less N_2' (e.x. moonless), which are called '*schemas*'.

> "They[3] 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." ([19] 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 [19] (p.21) as $\langle V_s\ X$ in the $N_b\rangle \rightarrow \langle$kick X in the shin$\rangle \rightarrow \langle$kick my pet giraffe in the shin$\rangle$[4]. In this example, $\langle V_s\ X$ in the $N_b\rangle$ is the schema with the highest schematicity, while the schema $\langle$kick X in the shin$\rangle$ is a more specific one. The structure of cognitive grammar is shown in Figure 2.

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

[2] V_s represents 'strike motion', while N_b 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 $N_{possession}$' on the principle of cognitive grammar.

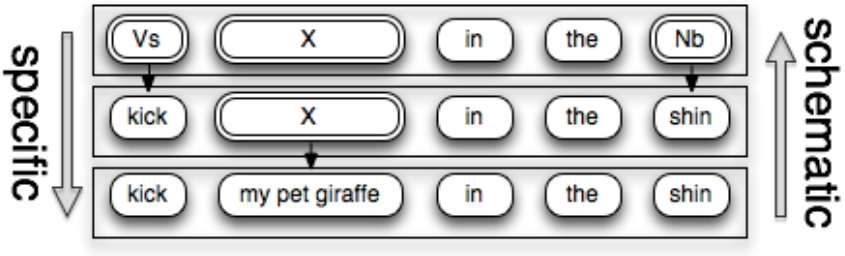

Figure 2. An example of structure of cognitive grammar

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 2 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 2 can be converted into a special dependency structure as shown in Figure 3. 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

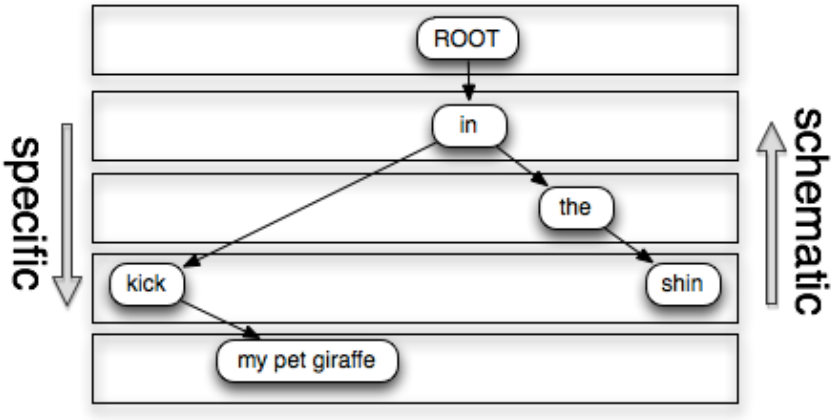

Figure 3. An example of CG-based dependency structure

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 4, 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 4(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

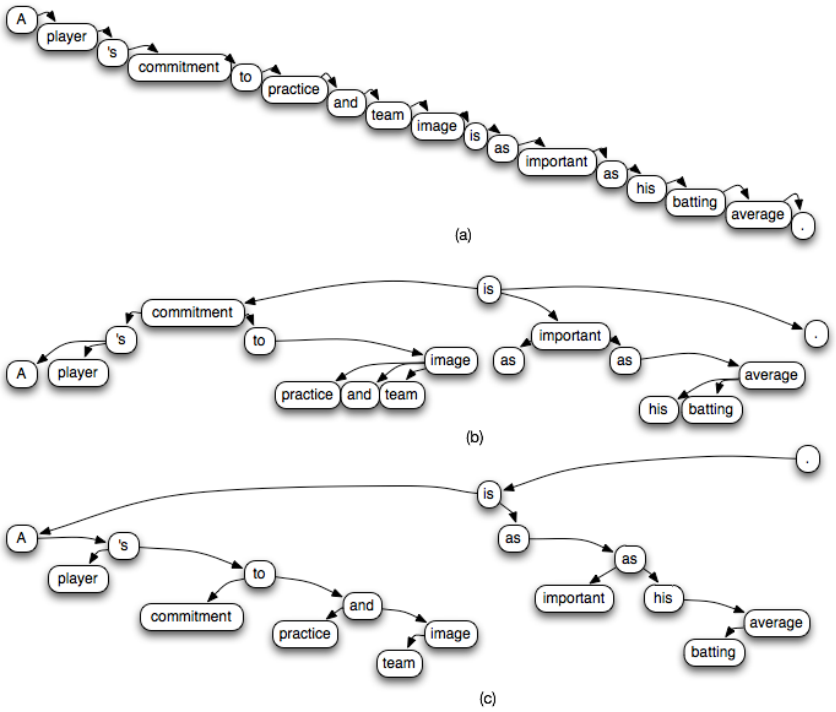

Figure 4. A demonstration of the difference among (a)n-gram, (b)dependency Grammar and (c)CG-based structure on sentence generation process

generated from their following words or even long-distance words. Taking 'as important as' in Figure 4 as an example, the second 'as' coexists with the first 'as' but not generated from 'important'. 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 does not

work accurately in every case. For example, in the sentence 'the more you study the more you know', according to the dependency parsing result, all other words depend on the word 'study' because it's the predicate of this sentence. However, 'the more ... the more ...' appears in almost every sentence which expresses that two things vary together, while not all such kinds of sentences use 'study' as their predicate word, which indicates that 'the more ... the more ...' is not actually generated from predicate 'study'. 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 4(c), 'is as ... 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 in a supervised fashion. Thus, compared to other structure-based language models, such as class-based language model [6], structured language model [8], factored language model(FLM) [5] and dependency tree language model [27] [10], the pattern-oriented language model does not use any specific linguistic knowledge or any abstracted categories, which makes it possible to be trained in a totally unsupervised fashion.

3. Hierarchical Word Sequence Structure

3.1 Hierarchical Word Sequence 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 [29], 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 Step2 and Step3 recursively until all the substrings become empty strings, then we can get a set of word nodes $M = \{w_i, w_j, w_k, ...\}$ and a set of edges $E = \{(w_i, w_j), (w_i, w_k), ...\}$. Finally, a binary tree $T = (M, E)$ 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*.

[5]If w_i appears multiple times in s, then select the first one.

Thus, the HWS language model [29] is essentially a special n-gram model with a different assumption. A conventional 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.

Thus, the HWS language models can inherit the advantage of cognitive grammar structure, which makes it gain more advantage than utterance-oriented structure and predicate-oriented structure, as we described in the previous chapter. However, this frequency-based method is only based on a simple heuristic assumption, for further improvement, in the next chapter, we propose a more generalized structure.

3.2 Generalized Hierarchical Word Sequence Structure

Suppose we are given a sentence $s = w_1, w_2, ..., w_n$ and a schematicity permutation function $f : s \mapsto s^{'}$, where $s^{'} = w_1^{'}, w_2^{'}, ..., w_n^{'}$ is a schematicity ordered permutation of s. For each word index $i(1 \leq i \leq n, w_i \in s)$, there is a corresponding reordered index $j(1 \leq j \leq n, w_j^{'} \in s^{'}, w_j^{'} = w_i)$.

To make the degree of schematicity more clear, we create an $n \times n$ matrix A. With the reordered index j, we fill each w_i into cell $A_{j,i}$ and fill ε into all other blank cells. We call the matrix A the **generalized hierarchical word sequence structure** (abbreviated as **GHWSS**) of the sentence s. An example is shown in Figure 5.

Any GHWSS can be converted to a dependency structure[6]. In a GHWSS, given any word $A_{j,i} \neq \varepsilon$, the words in its higher rows are $X = \{A_{k,m} | k < j, 1 \leq m \leq n, w_k^{'} = w_m\}$, in which the nearest two horizontal neighbor words of w are $\hat{l} = A_{k_l,m_l}$ and $\hat{r} = A_{k_r,m_r}$ respectively[7]. Then we assume that w depends on $\hat{w} = \hat{l}$ if $k_l > k_r$ or $\hat{w} = \hat{r}$ if $k_l < k_r$. For example, in Figure 5, given the word

[6]Not equivalent to dependency grammar structure.

[7]There is no $\hat{l}$ when $i = 1$, while no $\hat{r}$ when $i = n$.

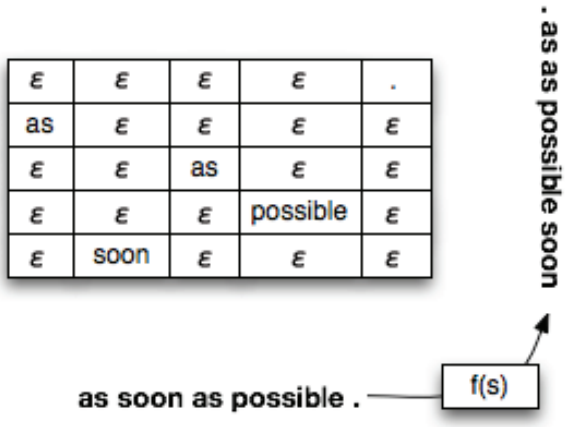

Figure 5. An Example of Generative Hierarchical Word Sequence Structure

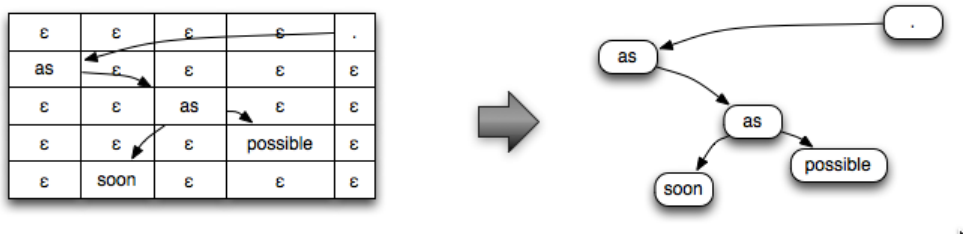

Figure 6. An Example of Conversion from GHWSS to dependency structure

'soon', the words in its higher rows are $X = \{as, as, possible, .\}$, in which the nearest horizontal neighbors of 'soon' are $\hat{l} = as$ and $\hat{r} = as$, since the second 'as' is closer to 'soon' vertically, we assume 'soon' depends the second 'as' in this GHWSS. Finally, the GHWSS in Figure 5 can be converted to a dependency structure as shown in Figure 6.

Since GHWSS is constructed by the degree of schematicity, which is consistent with the pattern-oriented idea of cognitive grammar, the dependency structure converted from GHWSS can be considered as CG-based dependency structure, which we described in Chapter 2.1. We set symbol '$\langle s \rangle$' and '$\langle / s \rangle$' as the beginning and ending of each dependency chain separately.

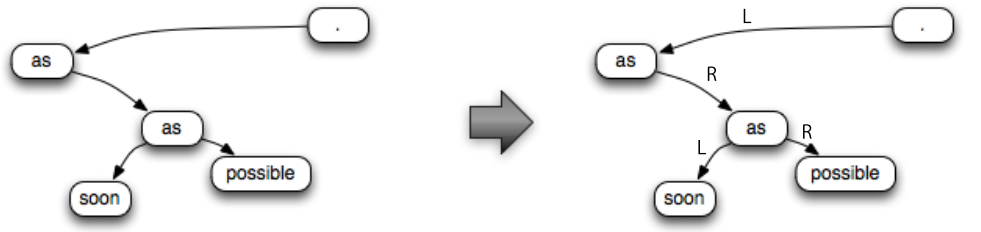

Figure 7. An Example of HWS Structure with Directional Information

For each word $w = A_{j,i}$, if we assume that it only depends on its previous few words in its dependency chain, then we can achieve HWS-n-grams under GHWSS. Taking Figure 6 as the example, we can extract hws-3-grams like {($\langle s\rangle$, $\langle s\rangle$, .), ($\langle s\rangle$, ., as), (., as, as), (as, as, possible), (as, possible, $\langle/s\rangle$), (as, as, soon), (as, soon, $\langle/s\rangle$)}.

Since the word order information is encoded in GHWSS, it is easy to attach directional information to hws-n-grams, so that the contextual constraint of natural language can be reflected more accurately in proposed language model.

The example with directional information is shown in Figure 7. Then the above hws-3-grams should be {($\langle s\rangle$, $\langle s\rangle$, .), ($\langle s\rangle$, .-R, $\langle/s\rangle$), ($\langle s\rangle$, .-L, as), (.-L, as-L, $\langle/s\rangle$), (.-L, as-R, as), (as-R, as-L, soon), (as-L, soon-L, $\langle/s\rangle$), (as-L, soon-R, $\langle/s\rangle$), (as-R, as-R, possible), (as-R, possible-L, $\langle/s\rangle$), (as-R, possible-R, $\langle/s\rangle$)} 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.

In this example, (as-R, as-L, soon) indicates that 'soon' is located between two 'as's, while (as-R, as-R, possible) indicates that 'possible' is located on the right side of the second 'as'. Similarly, if we use 4-grams or higher order ones, the relative position of each word will be more specific. In other words, for each word (node), its position (relative to the whole sentence) can be strictly determined by its preceding parent nodes. The bigger n is, the more syntactical information HWS-n-grams can reflect.

Notice that once a schematicity permutation function f is implemented, the GHWSS of any sentence can be determined and constructed. Thus, the performance of HWS-based language models, which is converted from GHWSS, is totally determined by how to implement the function f for reordering words in schematicity.

In the following three chapters, we introduce three different methods (assumption-based method, NST-based method and neural network based method) to implement the function f.

4. Method One: Assumption Based Construction of HWS Structure

The first method is converting raw sentences into the GHWSS under certain assumption.

In Section 2.2, we introduced the advantages of pattern-oriented assumption for rearranging word sequences. In this chapter, we propose various kinds of strategies for implementing function f based on the pattern-oriented assumption.

4.1 Top Down Strategy: Word Frequency Based Method

Since patterns are frequently used expressions, patterns consisting of frequently used words can be preferred.

Based on this idea, we propose a frequency-based method to achieve HWS structures. We can regard this method as a special case of GHWSS, if the function f is implemented as follows:

Step 1. Calculate word frequencies from training data and sort all words by their frequency. Let $C(v_j)$ represents the frequency of v_j, assume we get a frequency-sorted list $V = \{v_1, v_2, ..., v_m\}$, where $C(v_j) > C(v_{j+1}), 1 \leq j \leq m-1$.

Step 2. According to V, for each sentence $s = w_1, w_2, ..., w_n$, we permute it into $s^{'} = w_1^{'}, w_2^{'}, ..., w_n^{'}(w_k^{'} = v_x, w_{k+1}^{'} = v_y, 1 \leq k \leq n-1, 1 \leq x \leq y \leq m)$.

Then the GHWSS constructed by the permutation $s^{'}$ is equivalent to that of frequency-based HWS method.

4.2 Top Down Strategy: Word Association Based Method

Since in schematic patterns, such as 'too ... to', 'as ... as', the words are always coexist with each other, we can adopt the information of word association to achieve higher quality hierarchical patterns.

Based on this idea, instead of frequency-based method, we use the information of word association to construct high quality HWS structures. We can regard this method as a special case of GHWSS, if the function f is implemented as follows:

Step 1. For each sentence s in corpus D, we convert it into $s^{'}$, in which we remove the second and subsequent occurrence of the same word, so that each word only appear once. Then we have a new corpus $D' = \{s_i^{'} | 1 \leq i \leq |D|\}$ and only use it for word counting in Step 2. We call this process as *unification* on the purpose of making words that truly appears in more sentences (such as '.') be located at relatively higher rows of GHWSS.

Step 2. After the unification preprocessing, for each word w_i in the corpus $D' = \{s_i^{'} | 1 \leq i \leq |D|\}$, we count its frequency $C(w_i)$ and its cooccurrence with another word $C(w_i, w_j)$.

Step 3. For each original sentence $s \in D$, we initiate an empty list X and set the beginning symbol '$\langle s \rangle$' as the initial context c [8].

Step 4. For each word $w \in s$, we calculate its word association score with context c. In this dissertation, we use two different scores as the word association measure:

Dice coefficient [**11**] is a measure used for comparing the similarity of two samples, which is also widely used for collocation extraction. Dice coefficient only uses the relation between c and w to measure their strength of association. The Equation is shown as below.

$$score_{Dice}(c, w) = \frac{2 \times C(c, w)}{C(c) + C(w)} \tag{1}$$

T-score is another useful measure, whose calculation is shown as below [9] .

$$score_{TB}(c, w) = (C(c, w) - \frac{C(c) \times C(w)}{|V|}) \div \sqrt{C(c, w)} \tag{2}$$

[8]Since $\langle s \rangle$ appears only once in each sentence, we set $C(\langle s \rangle)$ as the size of corpus.

[9]V stands for the total number of words in corpus.

Then we add the i-th word $\hat{w}$ with the maximum score to list X[10] and use it to split s into two substrings $s_l = w_1, ..., w_{i-1}$ and $s_r = w_{i+1}, ..., w_n$.

Step 5. We set $\hat{w}$ as the new context $c^{'}$. For each word in s_l, we calculate its word association score with $c^{'}$ and add the word with the maximum score to list X[11] and use it to divide s_l into two smaller substrings. Then we apply the same process to the substring s_r.

Execute Step 4 and Step 5 recursively until anymore substrings cannot be divided, then the original sentence s is permuted as list X, by which GHWSS of s can be constructed.

4.3 Bottom up Strategy: Abstraction Based Method

> "In CG, rules take the form of schemas: they are abstract templates obtained by reinforcing the commonality inherent in a set of instances." ([19] pp.23)

According to this idea of cognitive grammar, we can use a gradual abstraction process to achieve pseudo 'schemas' in a bottom-up way.

Step 1. For each sentence $s = w_1, w_2, ...w_n$ in corpus D, we add $w_0 = \langle s \rangle$ and $w_{n+1} = \langle /s \rangle$ to its beginning and the end. For each word $w_i (1 \leq i \leq n)$, we count its cooccurrence with each of its left-side words and right-side words separately[12].

Step 2. For each sentence $s \in D$, we initiate an empty list X.

Step 3. For each word $w_i (1 \leq i \leq n)$ in the sentence $s \in D$, we calculate its *independence score* with its nearby words. The independence score is defined as Equation (3).

$$score_{independence}(w_i) = \frac{2 \times P(w_i|w_{i-1}^R) \times P(w_i|w_{i+1}^L)}{P(w_i|w_{i-1}^R) + P(w_i|w_{i+1}^L)} \tag{3}$$

[10] If $\hat{w}$ appears multiple times in s, then select the first one.

[11] If the context word $c^{'}$ also appears in s_l, then we regard it as the word with the maximum score and add it to X directly.

[12] which are not necessary to be continuous.

Step 4. We remove the i-th word $\hat{w}$ with the minimum independence score from the sentence s and add $\hat{w}$ to the list X. Then we set the remaining words as the new sentence $s^{'} = \langle s \rangle, w_1, w_2, ...w_{i-1}, w_{i+1}...w_n, \langle /s \rangle$.

Repeat Step 3 and Step 4 recursively until $s^{'}$ becomes an empty string ($\langle s \rangle, \langle /s \rangle$), then the original sentence s is permuted as list X. Since X is constructed in a bottom-up way, we need to reverse it for building GHWSS.

5. Method Two: Node Selection Tree Based Construction of HWS Structure

5.1 Priority List for GHWSS

In this Chapter, we implement the permute function f from a new perspective : priority list. Suppose we are given a priority list, which represents the degree of schematicity of all words, then any sentence can be reordered according to this list, as a result, the function f is implemented.

Taking frequency-based HWS method as an example, we take the frequency-sorted vocabulary list $V = \{v_1, v_2, ..., v_m\}$ as 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. If we take word association into consideration, as we did in Chapter 4.2, then we have to use a tree structure to represent the priority 'list'.

5.2 Generalized Priority List: Node Selection Tree

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 ε as an empty context where no more words generated. Obviously, each priority list starts from $ROOT^R$ and ends with ε. Thus, in original HWS method, priority

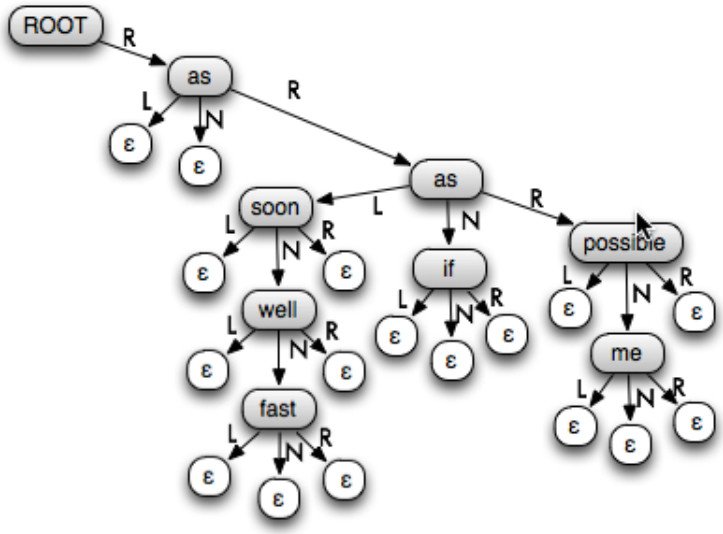

Figure 8. An Example of Node Selection Tree

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[13] 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 8, 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

[13]As the pattern '...as...as...', the same word are allowed to repeat as different nodes.

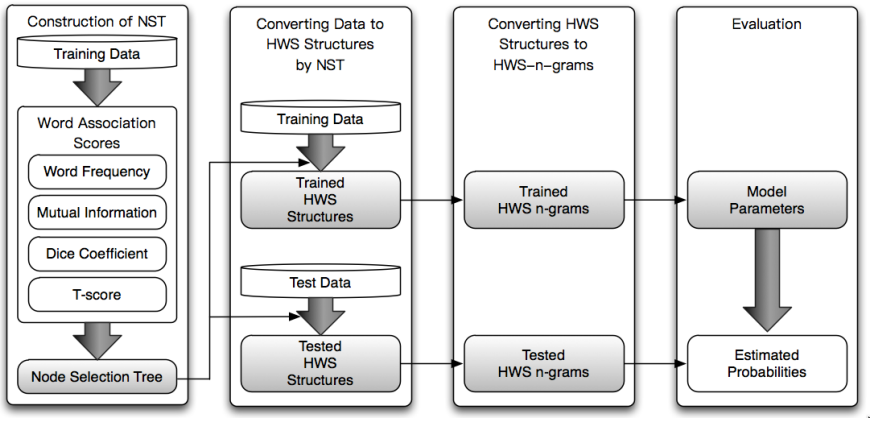

Figure 9. Generalized HWS Framework

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 11):

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 5.3.

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 5.4.

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, as the same way we described in Chapter 3.2.

Step 4: Build HWS language model. Finally, instead of conventional n-grams, we use HWS-n-grams to build language models.

5.3 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 5.2) 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 Ω 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 5.2, we adopt the same association scores we used in Chapter 4.2 as the function $score(c, w)$: Dice coefficient [11] (Equation (4)) and T-score (Equation (5)).

$$score_{DB}(c, w) = \frac{2 \times C(c, w)}{C(c) + C(w)} \tag{4}$$

$$score_{TB}(c, w) = (C(c, w) - \frac{C(c) \times C(w)}{|V|}) \div \sqrt{C(c, w)} \tag{5}$$

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.

Algorithm 1 Constructing an NST from given corpus S

Input: Corpus S (a set of string s)

Output: Node Selection Tree T (a set of triple (parent node, arc, subtree))

function GETNODE(c, S') ▷ return a word $\hat{w}$ from strings S' under context c

initialize a set Ω as the vocabulary of S'

$\hat{w} = \arg\max_{w_i \in \Omega}(score(c, w_i))$

return $\hat{w}$

end function

function GETTREE(c, S') ▷ return a tree T' from strings S' under context c

if $S' = \emptyset$ **then return** $\emptyset$

end if

$\hat{w}$ = 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**

$s_i = w_1, w_2, ..., w_n$ ▷ split string s_i into words

if $\hat{w} \notin s_i$ **then**

$N = N \cup s_i$ ▷ add string s_i to set N

else

j = index of the first $\hat{w}$ in s_i

$l = w_1, w_2, ..., w_{j-1}$

$r = w_{j+1}, ..., w_n$

$L = L \cup l$ ▷ add substring l to set L

$R = R \cup r$ ▷ add substring r to set R

end if

end for

initialize a subtree T' ▷ T' is a set of triple (parent node, arc, subtree)

$T' = T' \cup (\hat{w}$, 'N', GetTree($\hat{w}, N$))

$T' = T' \cup (\hat{w}$, 'L', GetTree($\hat{w}, L$))

$T' = T' \cup (\hat{w}$, 'R', GetTree($\hat{w}, R$))

return T'

end function

initialize a tree T ▷ T is a set of triple (parent node, arc, subtree)

$T = T \cup$ ('ROOT', 'R', GetTree('ROOT', S)) ▷ recursion by 'GetTree'

return T

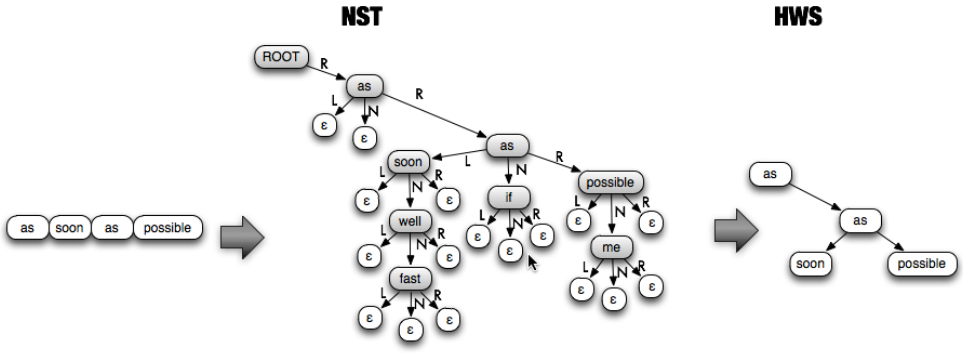

Figure 10. Conversion of raw sentences to HWS structure via NST

5.4 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 8 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 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 10. 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 this

method. This process is shown in Algorithm 2.

Algorithm 2 Converting a sentence s into HWS structure using an NST T

Input: Node Selection Tree T and sentence s

Output: HWS tree (recursively expressed as (left subtree, node, right subtree))

function TOHWS(T', s')

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

  end if

  if $T' = \emptyset$ **then**

    return ConvertByFrequency(s') ▷ frequency-based HWS subtree

  end if

  r = 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]['L'] ▷ the NST subtree under the 'L' arc of node r

    rTree = T'[r]['R'] ▷ the NST subtree under the 'R' arc of node r

    return (toHWS(lTree, lStr), r, toHWS(rTree, rStr)) ▷ Core Structure

  else

    nTree = T'[r]['N'] ▷ the NST subtree under the 'N' arc of node r

    return toHWS(nTree, s')

  end if

end function

$s = w_1, w_2, ..., w_n$ ▷ split sentence s into words

hws = toHWS(T, s) ▷ recursion by 'toHWS'

return hws

6. Method Three: Neural Network Based Construction of HWS Structure

6.1 Neural Network based HWS

In the previous chapters, we introduced two different methods. The two methods have in common that they both use HWS-n-grams instead of conventional n-grams, then use conventional smoothing methods (such as MKN smoothing) to estimate the probabilities. Thus, although both use discontinuous word sequences for probability estimation, the methods are still essentially discrete.

In [3], neural network is used for training distributed representation of each word, with these word embeddings, probabilities can be estimated in a continuous way.

In [22] [23], recurrent neural network is used for language modeling, which outperformed significantly and has been considered as the state-of-art language model.

In this chapter, we present a neural network based HWS language model (NNHWS), which not only takes the advantage of HWS structure, but also estimates probabilities in a continuous way.

6.2 Training of NNHWS

Basically, we use the CBOW architecture as the framework of our NNHWS. But for incorporating the basic idea of HWS, we make a slight change in the original CBOW model.

First, for training data, instead of bag of words, we use sentence as the basic training data unit. We also add border symbol '$\langle s \rangle$' '$\langle /s \rangle$' to each sentence so that grammar constraints can be further enhanced. each line of training data is a triple (left context L, right context R, generated word w_t).

Second, we use variable length words as the contexts L and R. We do this because according to the basic idea of HWS, for each word w_t, not all its previous words are related to its generation process. We filter out these words from its context so that the connection with its true context can be strengthened.

Our improved CBOW method is described as follows:

Step 1. Initiate the weight matrix W, bias matrix b and word embedding matrix C.

Step 2. Given a sentence s $= w_1, w_2, ...w_m$, we add border symbols to make it become s$^{'} = \langle s\rangle, w_1, w_2, ...w_m, \langle /s\rangle$. For each $1 \leq t \leq m$, we can get a training data triple (w_0^{t-1}, w_{t+1}^{m+1}, w_t), where w_0^{t-1} is considered as the left context of w_t, while w_{t+1}^{m+1} as its right context [14].

Step 3. For each word $w_k(1 \leq k \leq t-1)$ in the left context (except the border symbol $w_0 = \langle s\rangle$), we first look up their corresponding word embeddings in C, then use softmax method to estimate the conditional probability $P(w_k|\langle s\rangle, w_t)$. We remove all the words whose probability ≤ 0.001 from left context because they can be considered as unlikely words that play a part in the generation process of w_t. For all the remaining word embeddings in the left context, we calculate their average value as a new vector and take it as the *left context vector*.

Step 4. Execute Step 3 to the right context to achieve the *right context vector*.

Step 5. we concatenate the two context vectors as one long vector u. Then we use softmax to estimate $P(w_t|u)$ and update the word embeddings literately.

The architecture of NNHWS is shown in Figure 11.

6.3 Conversion into HWS structures by using NNHWS

After sufficient iterations, we can store the weight matrix W, bias matrix b and word embedding matrix C as the parameters of our neural network.

With these parameters, any sentence can be converted into a GHWSS.

[14]For actual implement, we set the maximum size for both contexts

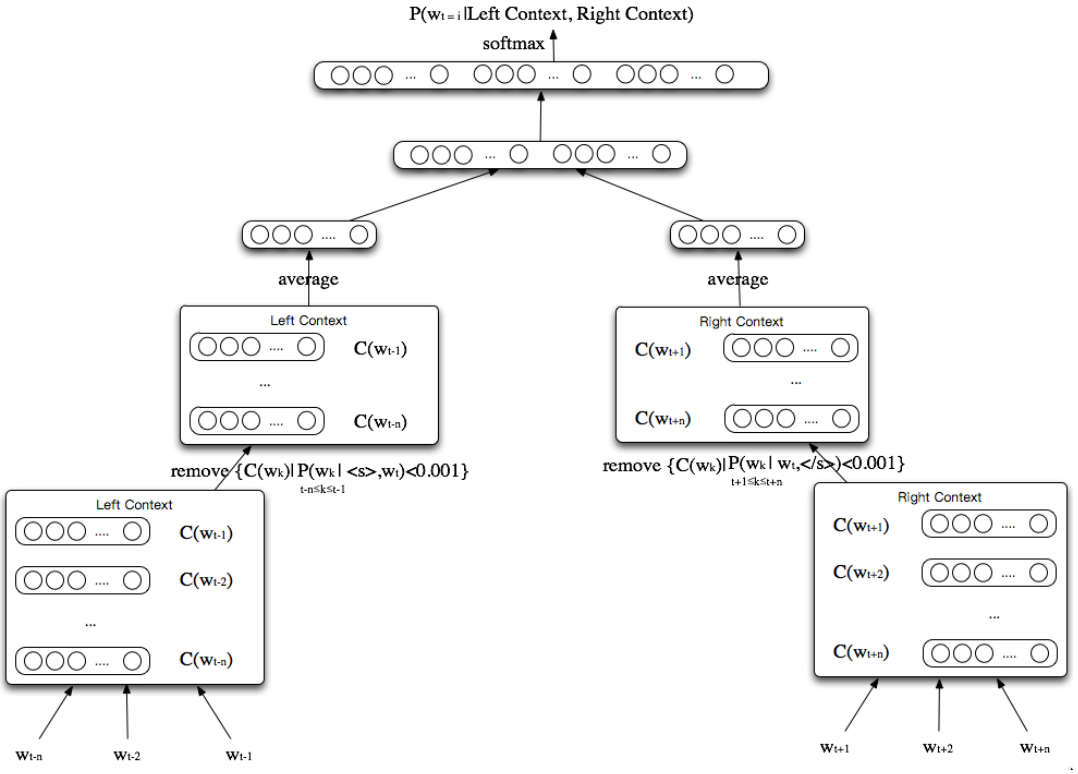

Figure 11. The architecture of NNHWS

Given a sentence s $= w_1, w_2, ...w_m$, we add border symbols to make it become s$' = \langle s\rangle, w_1, w_2, ...w_m, \langle /s\rangle$. For each word $w_t(1 \leq t \leq m) \in s'$, we can calculate its probability using these trained parameters. Then we can achieve a probability list $\{P(w_k|w_0^{k-1}, w_{k+1}^{m+1})|1 \leq k \leq m\}$. We permute the sentence s by sorting this probability list, then a GHWSS can be constructed as Figure 12.

6.4 Estimating probabilities of HWS structures by using NNHWS

After the HWS structure are constructed, the generation process of each word becomes clearer, which means we can filter out more unrelated words from contexts, and consequently, we can estimate probabilities more accurately.

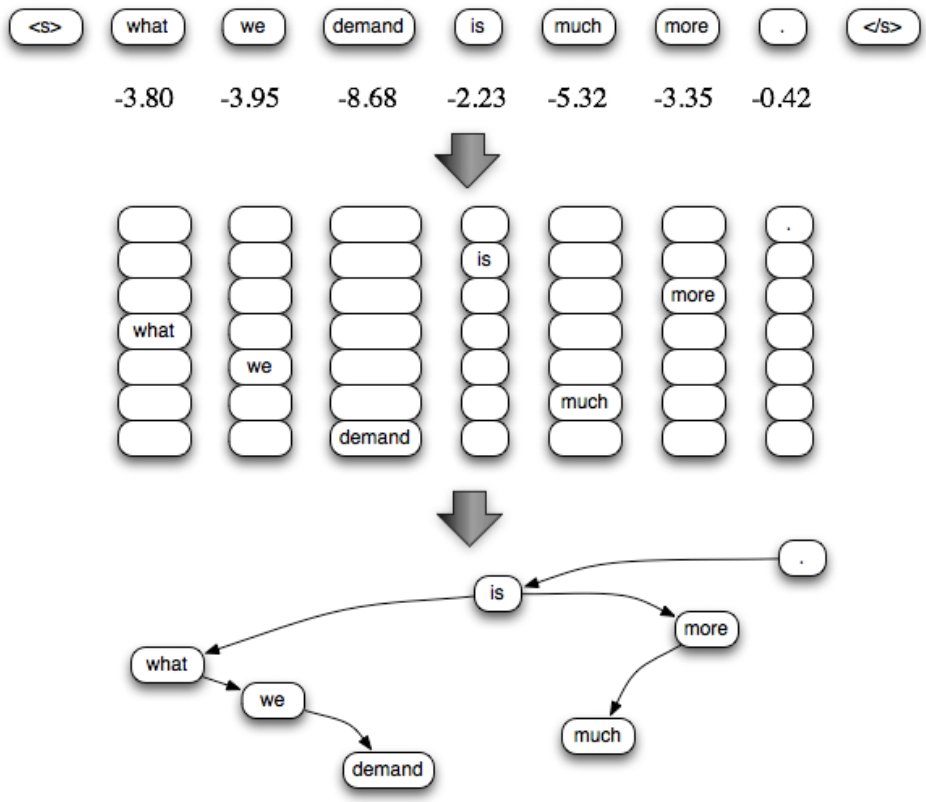

Figure 12. Conversion into HWS structures by using NNHWS

As shown in Figure 13, with the GHWSS constructed in previous section, the probability of sentence 'what we demand is much more .' can be calculated as the product of $\{P(.|(\langle s\rangle), (\langle /s\rangle))$, $P(is|(\langle s\rangle), (., \langle /s\rangle))$, $P(more|(\langle s\rangle, is)$, $(., \langle /s\rangle))$, $P(what|(\langle s\rangle)$, $(is, more, ., \langle /s\rangle))$, $P(we|(\langle s\rangle, what), (is, more, ., \langle /s\rangle))$, $P(much|(\langle s\rangle, what, we), (is, more, ., \langle /s\rangle))$, $P(demand|(\langle s\rangle, what, we), (is, much, more, ., \langle /s\rangle))\}$

In conventional language models, it is difficult for smoothing method to estimate those conditional probabilities, but in neural network based language models, those probabilities can be estimated smoothly and accurately.

NNHWS can take this advantage while retaining the merit of HWS structures.

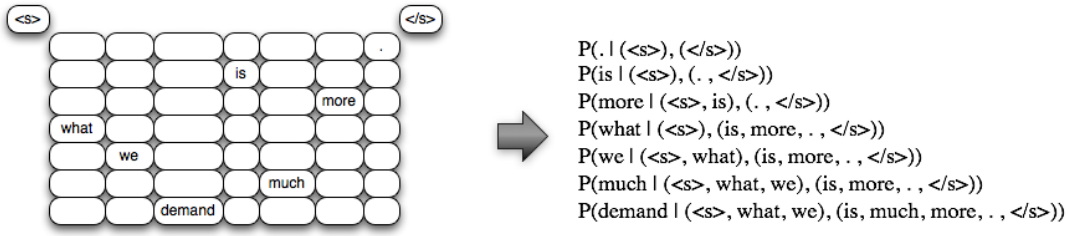

Figure 13. Estimating the probability of a GHWSS by using NNHWS

7. Intrinsic Evaluation

7.1 Settings

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

British National Corpus (BNC) [15] 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 [16] 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) [17] to split raw English sentences into words. We also converted all words to lowercase.

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 (6). Obviously, the higher coverage score

[15]http://www.natcorp.ox.ac.uk
[16]https://catalog.ldc.upenn.edu/LDC2011T07
[17]http://www.nltk.org

a language model can achieve, the more it can relieve the data sparseness problem (reduce the unseen sequences).

$$score_{coverage} = \frac{|TR \bigcap TE|}{|TE|} \tag{6}$$

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 (7)) to estimate how much redundancy is contained in a model.

$$score_{usage} = \frac{|TR \bigcap TE|}{|TR|} \tag{7}$$

A balanced measure between coverage and usage is calculated by Equation (8).

$$F\text{-}Score = \frac{2 \times coverage \times usage}{coverage + usage} \tag{8}$$

As intrinsic evaluation of language modeling, perplexity [20] 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.

7.2 Results

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

The NST-based method is in common with assumption-based method that they both extract HWS-n-grams from GHWSS and use conventional smoothing method (such as MKN) for probability estimation. Although they have different implementation for permutation function f, they are totally comparable for experiments. Thus, we use the same experiment settings for the two kinds of methods.

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

According to the results, the coverage and usage of HWS-bi-grams are nearly the same as that of conventional bi-grams. But as for tri-grams, each HWS-based method improves a lot on both coverage and usage. We can also find that NST-based method have better performance on coverage, but much worse performance on usage. Which means NST-based method brings more redundancy.

We also execute the same experiment on total word sequences and the result is shown in Table 2. The coverage and usage of HWS-uni-grams are a little larger than conventional unigrams because HWS-based methods are not continuous and consequently has more ending symbol '$\langle/s\rangle$'s. Different from unique sequence experiment, even in bi-grams, HWS-based methods have much better performance than the conventional bi-gram model. As for tri-grams, the HWS-based methods can even improve around 25 percentage points.

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 shown in Figure 14, all strategies improve along with the increasing of training data size. Also, all those methods increase at almost the same rate. Even we increase the training data size to 100 million words, HWS-based methods still have a great advantage over the conventional 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 [18]. Dependency Grammar structure is compatible with HWS except it allows one-to-many dependency relations. Thus, given a English dependency structure shown in Figure 15, we can convert it into 3-gram word sequences with directional information as {(has-L, date-L, a), (date-L, a-L, $\langle/s\rangle$), (date-L, a-R, $\langle/s\rangle$), (has-L, date-L, record), (date-L, record-L, $\langle/s\rangle$),

[18]The types of words of training data is 26600, while that of test data is 1373. OOV = 107.

Table 1. Coverage and Usage on Unique Word Sequences

Models	**Coverage(%)**	**Usage(%)**	**F-score(%)**
conventional uni-gram	56.560	16.300	25.307
frequency-based HWS-uni-gram	56.560	16.300	25.307
dice-based HWS-uni-gram	56.560	16.300	25.307
dice-based HWS-uni-gram(NST)	56.560	16.300	25.307
tscore-based HWS-uni-gram	56.560	16.300	25.307
tscore-based HWS-uni-gram(NST)	56.560	16.300	25.307
abstraction-based HWS-uni-gram	56.560	16.300	25.307
conventional bi-gram	**46.471**	12.015	**19.093**
frequency-based HWS-bi-gram	46.066	**12.019**	19.064
dice-based HWS-bi-gram	45.995	11.894	18.900
dice-based HWS-bi-gram(NST)	46.136	9.273	15.442
tscore-based HWS-bi-gram	45.709	11.872	18.848
tscore-based HWS-bi-gram(NST)	46.435	11.387	18.288
abstraction-based HWS-bi-gram	46.015	12.007	19.045
conventional tri-gram	27.164	5.626	9.321
frequency-based HWS-tri-gram	36.512	8.546	13.85
dice-based HWS-tri-gram	35.994	8.359	13.567
dice-based HWS-tri-gram(NST)	36.100	7.068	11.821
tscore-based HWS-tri-gram	36.473	8.501	13.788
tscore-based HWS-tri-gram(NST)	36.820	8.031	13.186
abstraction-based HWS-tri-gram	**36.885**	**8.659**	**14.026**

Table 2. Coverage and Usage on Total Word Sequences

Models	**Coverage(%)**	**Usage(%)**	**F-score(%)**
conventional uni-gram	97.264	96.261	96.760
frequency-based HWS-uni-gram	98.577	98.039	98.307
dice-based HWS-uni-gram	98.577	98.039	98.307
dice-based HWS-uni-gram(NST)	98.577	98.039	98.307
tscore-based HWS-uni-gram	98.577	98.039	98.307
tscore-based HWS-uni-gram(NST)	98.577	98.039	98.307
abstraction-based HWS-uni-gram	98.577	98.039	98.307
conventional bi-gram	83.121	76.336	79.584
frequency-based HWS-bi-gram	89.730	86.937	88.312
dice-based HWS-bi-gram	86.854	87.139	86.997
dice-based HWS-bi-gram(NST)	89.939	85.296	87.556
tscore-based HWS-bi-gram	89.949	**87.252**	**88.580**
tscore-based HWS-bi-gram(NST)	89.969	86.617	88.261
abstraction-based HWS-bi-gram	**90.056**	86.836	88.417
conventional tri-gram	51.151	40.191	45.013
frequency-based HWS-tri-gram	72.432	67.221	69.729
dice-based HWS-tri-gram	64.456	65.625	65.035
dice-based HWS-tri-gram(NST)	72.118	61.613	66.453
tscore-based HWS-tri-gram	**72.926**	**67.382**	**70.045**
tscore-based HWS-tri-gram(NST)	72.548	65.403	68.790
abstraction-based HWS-tri-gram	72.283	65.335	68.633

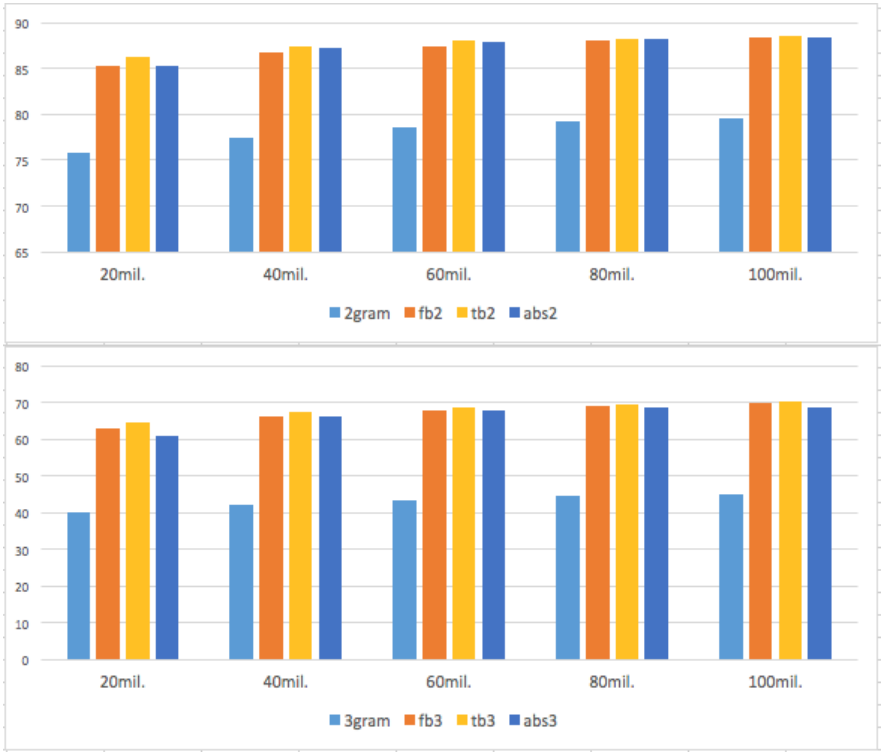

Figure 14. F-scores on Total Word Sequences with different Training Data Sizes

(date-L, record-R, $\langle/s\rangle$), ($\langle s\rangle$, has-L date), (has-L, date-R, $\langle/s\rangle$), ($\langle s\rangle$, has), ($\langle s\rangle$, has-R n't), (has-R n't-L, $\langle/s\rangle$), (has-R n't-R, $\langle/s\rangle$), ($\langle s\rangle$, has-R, been), (has-R, been-L, $\langle/s\rangle$), (has-R, been-R, set), (been-R, set-L, $\langle/s\rangle$), (been-R, set-R, $\langle/s\rangle$), ($\langle s\rangle$, has-R, .), (has-R, .-L, $\langle/s\rangle$), (has-R, .-R, $\langle/s\rangle$)}. The results of coverage and usage are shown in Table 3. For unique grams, our bi-gram model 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-

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

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

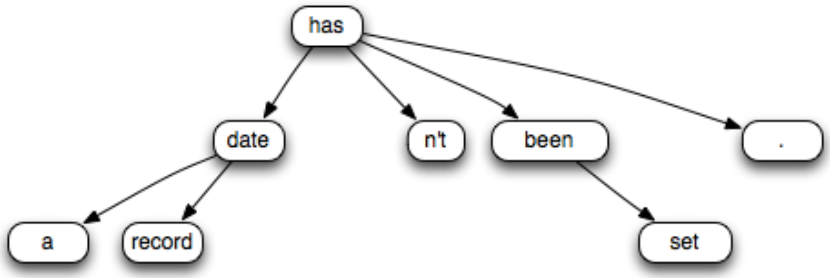

Figure 15. An example of English dependency structure (from CoNLL2007 shared task)

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 proposed method, we have to use an extra memory to store the NST tree.

Processing time: We use a 3GHz 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 2730ms. Also, due to its smallest model size, its training time (MKN smoothing) only costs 10101ms 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 [12]. In this paper, they propose an un-

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

Models	**Memory(mb)**	**Training+Evaluation Time(ms)**	**Perplexity**
conventional trigram	**39.6**	**2730 + 10101**	208.171
1skip-tri-gram	68.5	5866 + 18668	185.490
2skip-tri-gram	95.3	7251 + 26033	178.078
3skip-tri-gram	119.5	9649 + 36874	86.120
db HWS-tri-gram(NST)	4.6 + 67.7	14754 + 3287 + 16351	57.761
dependency-tri-gram	71.4	dependency parsing + 17828	**36.227**

Table 5. The Perplexities of HWS Model and Generative Dependency N-gram Language Model

Models	**en**	**de**	**es**
conventional tri-gram	86	139	86
generative dependency tri-gram	156	261	158
dice based-HWS-trigram(NST)	**40.885**	**54.325**	**46.464**

supervised 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.

8. 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 dissertation, we use the reranking of n-best translation candidates to examining how language models work in a statistical machine translation task.

8.1 Settings

For training language models, we set English as the target language. As for statistical machine translation toolkit, we use **Moses system**[19] to train the translation model and output 50-best translation candidates for each sentence of the test data. Then we use all 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 dissertation, we use the following measures for evaluating reranking results.

BLEU [24]: 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 [2]: METEOR score computes a one-to-one alignment between matching words in a candidate translation and a reference.

TER [28]: 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** [20] to perform the evaluation.

We compare our models with SRILM [21] and RNNLM [22]. For SRILM, we use

[19]http://www.statmt.org/moses/
[20]https://github.com/jhclark/multeval
[21]http://www.speech.sri.com/projects/srilm/download.html
[22]http://www.fit.vutbr.cz/ imikolov/rnnlm/

Kneser-Ney as the smoothing method [23] and defaults for other settings. For RNNLM, we use 30 hidden layers and 100 classes. We set -bptt as 4 and -direct-order as 3.

For NNHWS, we use Tensor Flow framework [24] for the implementation. We set the maximum words of left and right contexts as 3 separately, vocabulary as 50000 and embedding dimensions as 128. For loss function, we use Sampled Softmax Loss, and for optimizer, we use Adagrad algorithm. We perform our algorithm 1000 iterations and for each iteration, we use 128 random batches.

8.2 Results

We first use the French-English part of TED talk parallel corpus for the experiment dataset. The training data contains 139,761 sentence pairs, while the test data contains 1,617 sentence pairs.

We use various word rearranging strategies to perform experiments and compared them to the conventional n-gram strategy. For estimating the probabilities of translation candidates, we use the modified Kneser-Ney smoothing (MKN) as the smoothing method of all strategies. As shown in Table 8, HWS based strategies outperform that of n-gram on each score, and NST-based method generally outperforms their counterparts of assumption-based method.

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, HWS based strategies still outperform n-gram strategy on all the three measures, especially with an obvious improvement on TER score.

The NST-based methods perform better because NST trees encode much more syntactical information inside, which makes the GHWSS constructed by NST trees have more grammatical constraints than that of assumption-based methods.

[23] -interpolate kndiscount.

[24] https://www.tensorflow.org

Table 6. Performance on French-English SMT Task Using Various Word Arranging Strategies

Models	BLEU	METEOR	TER
conventional tri-gram(SRILM)	31.3	33.4	49.0
frequency-based HWS-tri-gram	31.7	**33.6**	48.6
dice-based HWS-tri-gram	31.5	33.5	48.5
dice-based HWS-tri-gram(NST)	**31.8**	**33.6**	**48.3**
tscore-based HWS-tri-gram	31.7	33.5	48.5
tscore-based HWS-tri-gram(NST)	31.7	**33.6**	48.4
abstraction-based HWS-tri-gram	31.5	33.5	48.6

But on the other hand, we should not also neglect that NST-based methods are not as efficient as assumption-based methods. the NST trees require enormous space to store it and make much redundancy as seen from the result of the intrinsic experiment in this chapter. Building NST trees also increases the computational complexity. The NST-based method is both time-consuming and space-consuming.

Thus, it can be a tradeoff between the two methods, either should be selected according with the demands of specific task.

We also use the baseline system described in Moses[25] to evaluate NNHWS. Equally, we use *news-commentary-v8* and *newstest2011* as training data and test data separately [26].

The results is shown in Table 8, we can find that HWS-based strategies outperform conventional n-grams on BLEU and METEOR, and NST-based methods perform better than their counterparts of assumption-based methods.

We also compare those models to RNNLM, which is considered as the state-

[25]http://www.statmt.org/moses/?n=Moses.Baseline

[26]Which can be downloaded at http://www.statmt.org/wmt13/training-parallel-nc-v8.tgz

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

Spanish-English			
Models	**BLEU**	**METEOR**	**TER**
conventional tri-gram(SRILM)	31.9	34.9	48.8
frequency-based-tri-gram	32.0	34.9	48.5
dice-based HWS-tri-gram	32.0	34.9	48.2
dice-based HWS-tri-gram(NST)	32.1	34.9	48.2
tscore-based HWS-tri-gram	32.2	34.9	48.2
tscore-based HWS-tri-gram(NST)	**32.3**	34.9	**48.1**
abstraction-based HWS-tri-gram	31.9	34.9	48.4
Japanese-English			
Models	**BLEU**	**METEOR**	**TER**
conventional tri-gram(SRILM)	7.5	**19.2**	87.4
frequency-based-tri-gram	7.6	**19.2**	86.4
dice-based HWS-tri-gram	7.6	19.1	86.2
dice-based HWS-tri-gram(NST)	7.6	19.2	86.1
tscore-based HWS-tri-gram	7.6	19.1	86.4
tscore-based HWS-tri-gram(NST)	**7.8**	19.2	**86.0**
abstraction-based HWS-tri-gram	7.7	19.1	86.1
Chinese-English			
Models	**BLEU**	**METEOR**	**TER**
conventional tri-gram(SRILM)	12.5	22.1	76.7
frequency-based-tri-gram	**12.6**	**22.2**	76.2
dice-based HWS-tri-gram	12.5	22.1	76.0
dice-based HWS-tri-gram(NST)	12.5	22.1	76.0
tscore-based HWS-tri-gram	**12.6**	22.1	76.0
tscore-based HWS-tri-gram(NST)	**12.6**	22.1	76.0
abstraction-based HWS-tri-gram	**12.6**	**22.2**	**75.9**

Table 8. Performance on French-English SMT Task Using Various Word Arranging Strategies

Models	BLEU	METEOR	TER
conventional tri-gram(SRILM)	20.6	28.4	60.1
frequency-based HWS-trigram	20.6	28.5	60.4
dice-based HWS-tri-gram	20.7	28.6	60.3
dice-based HWS-tri-gram(NST)	20.7	28.6	60.2
tscore-based HWS-tri-gram	20.7	28.6	60.2
tscore-based HWS-tri-gram(NST)	20.7	28.6	60.1
Abstraction-based HWS-trigram	20.7	28.5	60.3
RNNLM	21.4	28.7	58.9
NNHWS	**21.5**	**28.8**	**58.7**

of-art language model. with the power of neural network, RNNLM gains an obvious advantage on all measures, but when we apply neural network on HWS structure, even better performance can be achieved, which proves the effectiveness of proposed structure.

8.3 Analysis

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 $\langle NUMBER \rangle$ 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.

{($\langle s\rangle$, .), ($\langle s\rangle$, .-L ,are), (.-L ,are-L, there), (are-L, there-L, and), (there-L, and-L, $\langle/s\rangle$), (there-L ,and-R ,$\langle/s\rangle$), (are-L, there-R, $\langle/s\rangle$), (.-L ,are-R, of), (are-R ,of-L, about), (of-L, about-L ,$\langle/s\rangle$), (of-L ,about-R, 70), (about-R ,70-L, $\langle/s\rangle$), (about-R ,70-R, $\langle/s\rangle$), (are-R ,of-R, them), (of-R ,them-L, $\langle/s\rangle$), (of-R, them-R ,$\langle/s\rangle$), ($\langle s\rangle$, .-R, $\langle/s\rangle$)}

Among these sequences, patterns such like "... there are ... ." "... are ... of ... ." (correspond to '(.-L ,are-L, there)' and '(.-L ,are-R, of)' respectively), which 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 patten "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.

9. Conclusion

9.1 Summary

In this paper, based on the basic idea and structure of cognitive grammar, we proposed a generalized hierarchical word sequence structure for language modeling. Under this structure, we presented three different kinds of unsupervised strategies 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, which also proves that our strategies can be used as better alternatives for n-gram language models.

But on the other hand, compare to conventional n-gram language models, our models relatively need more processing time, especially for the NST-based method, which also needs an extra space for storing the tree structure. 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.

9.2 Future Work

9.2.1 Supervised HWS

For future work, we also plan to train HWS models in a supervise fashion. According to the principles of Cognitive Grammar structure, we can build an annotated corpus by converting sentences into CG-based dependency structures, then use it for 'parsing' sentences into HWS structures. In this supervised fashion,

the 'parsing' of CG-based dependency structure can be actually converted as the classification of schemas. It is expected that the HWS models trained in this way can achieve better performance than the unsupervised methods presented in this thesis.

9.2.2 Pattern Embeddings

For the improvement of NNHWS, we also plan to treat patterns (or schemas in CG) as an whole unit to train pattern embeddings, just as what we do to the words in all kinds of neural network language models. As proposed structure is constructed by hierarchical patterns, instead of using word embeddings, pattern embeddings may reflect the mechanism of nature language more precisely.

9.2.3 Function-driven NLP

Once the goals described in Section 9.2.1 and Section 9.2.2 are completed, it becomes possible to perform classical natural language processing applications in a new function-driven fashion.

With those hierarchical pattern embeddings, the language comprehension process can be considered as recognizing schemas from raw sentences hierarchically. After mapping each schema to its corresponding function, the meaning of the sentence can be hierarchically comprehended, which can be considered as the process of semantic analysis.

On the other hand, the language generation process can be considered as specifying certain function hierarchically. After mapping each specified function to its corresponding schema, a CG grammar structure can be established. Combining all the schema hierarchies into one line, a sentence can be generated.

With these two basic ideas, we have new solutions for some NLP applications.

For example, in the machine translation task, since we treat the whole pattern as a unit and assign it with an embedding, after the training process, a pattern

embedding space can be established . Furthermore, just as word embeddings, for two different languages, the patterns sharing the similar functions may have similar geometric arrangements in both language embeddings space too, which makes it possible to perform function-driven machine translation. Given a sentence of source language, we first hierarchically recognize the schemas and convert it into the hierarchical pattern structure, then for each pattern in this structure, we map it to the embedding space of the target language. Finally, we combine the mapped hierarchical structure into one line, then the target translation can be generated.

It is also possible to apply it into conversation system, given a sentence, we first convert it into the hierarchical pattern structure, since a pattern with the highest schematicity in this structure always represents the main function of this sentence (e.x. 'the more ... the more ...', '... would rather ... than ...'), the main purpose of this sentence can be estimated. Then we treat the specific parts of this pattern as the details of the sentence, processing with the knowledge database, we can choose the most appropriate answer function. Then we map this function to its corresponding schema, and hierarchically specify it, the appropriate reply to the input sentence can be finally generated.

In summary, the function-driven method provides new solutions to many classical NLP tasks, which is a promising direction in Natural Language processing.

Acknowledgements

First I want to thank my mentor Professor Yuji Matsumoto. He has always been very kind to every student. Since I majored in linguistics, I knew few things about natural language processing before I came to Japan. Even so, he still accepted me as his student and taught me from the basic. He taught me how to write computer science paper and how to do research. He taught me how to write code and how to use latex. Professor Matsumoto totally changed my life, and I shall never forgot his kindness and endless support.

I am also grateful to Associate Professor Mamoru Komachi, Kevin Duh and Hiroyuki Shindo, they all gave me many help and precious ideas.

I would also like to thank the staff in International Student Affairs and Ms. Yuko Kitagawa. They all helped me too much in my private life these years. I also want to thank the MEXT for providing me scholarship to complete my Ph.D course.

Finally I want to thank my parents and my fiancee, they always support me unconditionally, and give me courage when I encounter difficulties. Thanks for their accompany. I dedicate this thesis to them.

References

[1] B. Allison, D. Guthrie, L. Guthrie, W. Liu, and Y Wilks. *Quantifying the Likelihood of Unseen Events: A further look at the data Sparsity problem.* Awaiting publication, 2005.

[2] S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. In *Journal of machine learning research*, volume 3, pages 1137–1155, 2003.

[4] S. Bickel, P. Haider, and T. Scheffer. Predicting sentences using n-gram language models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 193–200, 2005.

[5] J. A. Bilmes and K. Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 2, pages 4–6, 2003.

[6] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. La. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[7] P.F Brown, J Cocke, S.A Pietra, V.J Pietra, F Jelinek, J.D Lafferty, R.L Mercer, and P.S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[8] C. Chelba. A structured language model. In *Proceedings of ACL-EACL*, pages 498–500, 1997.

[9] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393, 1999.

[10] W. Chen, M. Zhang, and H Li. Utilizing dependency language models for graph-based dependency parsing models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 213–222, 2012.

[11] L. R. Dice. Measures of the amount of ecologic association between species. In *Ecology*, volume 26, pages 297–302, 1945.

[12] C. Ding and M. Yamamoto. An unsupervised parameter estimation algorithm for a generative dependency n-gram language model. In *IJCNLP*, pages 516–524, 2013.

[13] D. Guthrie, B. Allison, W. Liu, and L. Guthrie. A closer look at skip-gram modeling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation*, pages 1–4, 2006.

[14] X. Huang, F. Alleva, H.W. Hon, M.Y. Hwang, and K. F. Lee. The sphinx-ii speech recognition system: an overview. 7(2):137–148, 1993.

[15] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[16] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. *Acoustics, Speech, and Signal Processing*, 1:181–184, 1995.

[17] R.W. Langacker. An introduction to cognitive grammar. 10:1–40, 1986.

[18] R.W. Langacker. *Grammar and Conceptualization.* Mouton de Gruyter, DE, 1999.

[19] R.W. Langacker. *cognitive grammar - A Basic Introduction.* Oxford, 2008.

[20] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing.* MIT Press, 1999.

[21] E. Mays, F. J. Damerau, and R. L. Mercer. Context based spelling correction. *Information Processing and Management*, 27(5):517–522, 1990.

[22] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[23] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky. Rnnlm-recurrent neural network language modeling toolkit. In *In Proc. of the 2011 ASRU Workshop*, pages 196–201, 2011.

[24] K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318, 2002.

[25] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[26] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[27] L. Shen, J. Xu, and R.M. Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *ACL*, pages 577–585, 2008.

[28] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231, 2006.

[29] X. Wu and Y. Matsumoto. A hierarchical word sequence language model. In *Proceedings of The 28th Pacific Asia Conference on Language*, pages 489–494, 2014.

[30] X. Wu and Y. Matsumoto. An improved hierarchical word sequence language model using directional information. In *Proceedings of The 29th Pacific Asia Conference on Language*, pages 453–458, 2015.

[31] X. Wu, Y. Matsumoto, K. Duh, and S. Hiroyuki. An improved hierarchical word sequence language model using word association. In *Statistical Language and Speech Processing*, pages 275–287, 2015.

List of Publications

Journal Paper

[1] X.Wu, K. Duh and Y. Matsumoto. Generalized HierarchicalWord Sequence Framework for Language Modeling. **自然言語処理**, 24(3), 2017.

International Conference Papers

[1] X. Wu and Y. Matsumoto. A hierarchical word sequence language model. In *Proceedings of The 28th Pacific Asia Conference on Language*, pages 489-494, 2014.

[2] X. Wu and Y. Matsumoto. An improved hierarchical word sequence language model using directional information. In *Proceedings of The 29th Pacific Asia Conference on Language*, pages 453-458, 2015.

[3] X. Wu, K. Duh, S. Hiroyuki and Y. Matsumoto. An improved hierarchical word sequence language model using word association. In *Statistical Language and Speech Processing*, pages 275-287, 2015.

[4] X. Wu, K. Duh and Y. Matsumoto. A generalized framework for hierarchical word sequence language model. In *Proceedings of The 30th Pacific Asia Conference on Language*, 2016.

Appendix

A. Modified Kneser-Ney Smoothing

The state-of-the-art method for smoothing is modified Kneser-Ney smoothing proposed in [9]. Based on normal Kneser-Ney smoothing, MKN uses different discount parameters for non-zero counts, whose calculation is shown as Equation (9).

$$P_{MKN}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{C(w_{i-n+1}^{i}) - D(C(w_{i-n+1}^{i})), 0\}}{C(w_{i-n+1}^{i-1})} + \gamma_{high}(w_{i-n+1}^{i-1})\hat{P}_{MKN}(w_i|w_{i-n+2}^{i-1}) \tag{9}$$

The discount value D is a discount value calculate by Equation (10).[27]

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 = 1 - 2\frac{n_1}{n_1+n_2}\frac{n_2}{n_1} & \text{if } c = 1 \\ D_2 = 2 - 3\frac{n_1}{n_1+n_2}\frac{n_3}{n_2} & \text{if } c = 2 \\ D_{3+} = 3 - 4\frac{n_1}{n_1+n_2}\frac{n_4}{n_3} & \text{if } c > 2 \end{cases} \tag{10}$$

And $\gamma_{high}(w_{i-n+1}^{i-1})$ is defined as Equation (11).[28]

$$\gamma_{high}(w_{i-n+1}^{i-1}) = \frac{D_1N_1(w_{i-n+1}^{i-1}\bullet) + D_2N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+}N_{3+}(w_{i-n+1}^{i-1}\bullet)}{C(w_{i-n+1}^{i-1})} \tag{11}$$

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

[27] n_i is the total number of n-grams which appear exactly i times in the training data.

[28] $N_1(w_{i-n+1}^{i-1}\bullet) = |\{w_i : C(w_{i-n+1}^{i}) = 1\}|$, and $N_2(w_{i-n+1}^{i-1}\bullet)$ $N_{3+}(w_{i-n+1}^{i-1}\bullet)$ are counted in a similar way.

$$\begin{aligned}
&\hat{P}_{MKN}(w_i|w_{i-n+1}^{i-1}) \\
&= \frac{max\{N_{1+}(\bullet w_{i-n+1}^{i}) - D(C(w_{i-n+1}^{i})), 0\}}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} \\
&+\gamma_{mid}(w_{i-n+1}^{i-1})\hat{P}_{MKN}(w_i|w_{i-n+2}^{i-1})
\end{aligned} \tag{12}$$

where $\gamma_{mid}(w_{i-n+1}^{i-1})$ is defined as Equation (13).

$$\begin{aligned}
&\gamma_{mid}(w_{i-n+1}^{i-1}) = \\
&\frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)}
\end{aligned} \tag{13}$$