An Empirical Comparison Between *N*-gram and Syntactic Language Models for Word Ordering

Jiangming Liu and **Yue Zhang**
Singapore University of Technology and Design,
8 Somapah Road, Singapore, 487372
{jiangming_liu, yue_zhang}@sutd.edu.sg

Abstract

Syntactic language models and *N*-gram language models have both been used in word ordering. In this paper, we give an empirical comparison between *N*-gram and syntactic language models on word order task. Our results show that the quality of automatically-parsed training data has a relatively small impact on syntactic models. Both of syntactic and *N*-gram models can benefit from large-scale raw text. Compared with *N*-gram models, syntactic models give overall better performance, but they require much more training time. In addition, the two models lead to different error distributions in word ordering. A combination of the two models integrates the advantages of each model, achieving the best result in a standard benchmark.

1 Introduction

N-gram language models have been used in a wide range of the generation tasks, such as machine translation (Koehn et al., 2003; Chiang, 2007; Galley et al., 2004), text summarization (Barzilay and McKeown, 2005) and realization (Guo et al., 2011). Such models are trained from large-scale raw text, capturing distributions of local word *N*-grams, which can be used to improve the fluency of synthesized text.

More recently, syntactic language models have been used as a complement or alternative to *N*-gram language models for machine translation (Charniak et al., 2003; Shen et al., 2008; Schwartz et al., 2011), syntactic analysis (Chen et al., 2012) and tree linearization (Song et al., 2014). Compared with *N*-gram models, syntactic models capture rich structural information, and can be more effective in improving the fluency of large constituents, long-range dependencies and overall sentential grammaticality. However, Syntactic models require annotated syntactic structures for training, which are expensive to obtain manually. In addition, they can be slower compared to *N*-gram models.

In this paper, we make an empirical comparison between syntactic and *N*-gram language models on the task of word ordering (Wan et al., 2009; Zhang and Clark, 2011a; De Gispert et al., 2014), which is to order a set of input words into a grammatical and fluent sentence. The task can be regarded as an abstract language modeling problem, although methods have been explored extending it for tree linearization (Zhang, 2013), broader text generation (Song et al., 2014) and machine translation (Zhang et al., 2014).

We choose the model of Liu et al.(2015) as the syntactic language model. There has been two main types of syntactic language models in the literature, the first being relatively more oriented to syntactic structure, without an explicit emphasis on word orders (Shen et al., 2008; Chen et al., 2012). As a result, this type of syntactic language models are typically used jointly with *N*-gram model for text-to-text tasks. The second type models syntactic structures incrementally, thereby can be used to directly score surface orders (Schwartz et al., 2011; Liu et al., 2015). We choose the discriminative model of Liu et al. (2015), which gives state-of-the-art results for word ordering.

We try to answer the following research questions by comparing the syntactic model and the *N*-gram model using the same search algorithm.

• **What is the influence of automatically-parsed training data on the performance of syntactic models.** Because manual syntactic annotations are relatively limited and highly expensive, it is necessary to use large-scale automatically-parsed sentences for training syntactic language models. As a result, the syntactic structures that a word ordering system learns can be inaccurate. However, this might not affect

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 369–378,
Lisbon, Portugal, 17-21 September 2015.

$$\text{Initial State} \quad ([], set(1...n), \emptyset)$$

$$\text{Final State} \quad ([], \emptyset, A)$$

Induction Rules:

$$\text{SHIFT} \quad \frac{(\sigma,\rho,A)}{([\sigma|i],\rho-\{i\},A)}$$

$$\text{L-ARC} \quad \frac{([\sigma|j\ i],\rho,A)}{([\sigma|i],\rho,A\cup\{j\leftarrow i\})}$$

$$\text{R-ARC} \quad \frac{([\sigma|j\ i],\rho,A)}{([\sigma|i],\rho,A\cup\{j\rightarrow i\})}$$

Figure 1: Deduction system for transition-based linearization.

step	action	σ	ρ	A
init		[]	(0 1 2)	$\emptyset$
0	shift	[1]	(0 2)	
1	shift	[1 2]	(0)	
2	L-arc	[2]	(0)	$A \cup \{1 \leftarrow 2\}$
3	shift	[2 0]	()	
4	R-arc	[2]	()	$A \cup \{2 \rightarrow 0\}$

Figure 2: Transition-based process for ordering {"*potatoes*$_0$", "*Tom*$_1$", "*likes*$_2$"}.

the quality of the synthesized output, which is a string only. We quantitatively study the influence of parsing accuracy of syntactic training data on word ordering output.

• **What is the influence of data scale on the performance.** *N*-gram language models can be trained efficiently over large numbers of raw sentences. In contrast, syntactic language models can be much slower to train due to rich features. We compare the output quality of the two models on different scales of training data, and also on different amounts of training time.

• **What are the errors characteristics of each model.** Syntactic language models can potentially be better in capturing larger constituents and overall sentence structures. However, compared with *N*-gram models, little work has been done to quantify the difference between the two models. We characterise the outputs using a set of different measures, and show empirically the relative strength and weakness of each model.

• **What is the effect of model combination.** Finally, because the two models make different types of errors, they can be combined to give better outputs. We develop a combined model by discretizing probability from *N*-gram model, and using them as features in the syntactic model. The combined model gives the best results in a standard benchmark.

2 Systems

2.1 Syntactic word ordering

Syntactic word ordering algorithms take a multi-set of input words constructing an output sentence and its syntactic derivation simultaneously. Transition-based syntactic word ordering can be modelled as an extension to transition-based parsing (Liu et al., 2015), with the main difference being that the order of words is not given in the input, which leads to a much larger search space.

We take the system of Liu, et al.[1], which gives state-of-the-art performance and efficiencies in standard word ordering benchmark. It maintains outputs in stack σ, and orders the unprocessed incoming words in a set ρ. Given an input bag of words, ρ is initialized to the input and σ is initialized as empty. The system repeatedly applies transition actions to consume words from ρ and construct output on σ.

Figure 1 shows the deduction system, where ρ is unordered and any word in ρ can be shifted onto the stack σ. The set of actions are SHIFT, L-ARC and R-ARC. The SHIFT actions add a word to the stack. For the L-ARC and R-ARC actions, new arcs $\{j \leftarrow i\}$ and $\{j \rightarrow i\}$ are constructed respectively. Under these possible actions, the unordered word set "*potatoes*$_0$ *Tom*$_1$ *likes*$_2$" is generated as shown in Figure 2, and the result is "*Tom*$_1$ ←*likes*$_2$→*potatoes*$_0$".

We apply the learning and search framework of Zhang and Clark (2011a). Pseudocode of the search algorithm is shown in Algorithm 1. [] refers to an empty stack, and $set(1...n)$ represents the full set of input words W and n is the number of distinct words. $candidates$ stores possible states, and $agenda$ stores temporary states transited from possible actions. GETACTIONS generates a set of possible actions depending on the current state s. APPLY generates a new state by applying action on the current state s. N-BEST produces the top k candidates in $agenda$. Finally, the algorithm returns the highest-score state $best$ in the $agenda$.

A global linear model is used to score search hypotheses. Given a hypothesis h, its score is calculated by:

$$Score(h) = \Phi(h) \cdot \vec{\theta},$$

[1] http://sourceforge.net/projects/zgen/

Algorithm 1 Transition-based linearisation

Input: *W*, a set of input word
Output: the highest-scored final state

$candidates \leftarrow ([], set(1..n), \emptyset)$
$agenda \leftarrow \emptyset$
$N \leftarrow 2n$
for $i \leftarrow 1..N$ **do**
  for s **in** $candidates$ **do**
    for $action$ **in** GETACTIONS(s) **do**
      $agenda \leftarrow$ APPLY($s, action$)
    end for
  end for
  $candidates \leftarrow$ N-BEST($agenda$)
  $agenda \leftarrow \emptyset$
end for
$best \leftarrow$ BEST($candidates$)
return $best$

where $\Phi(h)$ is the feature vector of h, extracted by using the same feature templates as Liu et al.(2015), which are shown in Table 1 and $\vec{\theta}$ is the parameter vector of the model. The feature templates essentially represents a syntactic language model. As shown in Figure 2, from the hypotheses produced in steps 2 and 4, the features "$Tom_1 \leftarrow likes_2$" and "$likes_2 \rightarrow potatoes_0$" are extracted, which corresponds to $P(Tom_1|likes_2)$ and $P(potatoes_0|likes_2)$ respectively in the dependency language model of Chen et al.,(2012).

Training. We apply perceptron with early-update (Collins and Roark, 2004), and iteratively tune related parameters on a set of development data. For each iteration, we measure the performance on the development data, and choose best parameters for final tests.

2.2 *N*-gram word ordering

We build an *N*-gram word ordering system under the same beam-search framework as the syntactic word ordering system. In particular, search is performed incrementally, from left to right, adding one word at each step. The decoding process can be regarded as a simplified version of Algorithm 1, with only SHIFT being returned by GETACTIONS, and the score of each transition is given by a standard *N*-gram language model. We use the same beam size for both *N*-gram and the syntactic word ordering. Compared with the syntactic model, the *N*-gram model has less information for disambiguation, but also has less structural ambiguities, and therefore a smaller search space.

Unigram
S_0w; S_0p; $S_{0,l}w$; $S_{0,l}p$; $S_{0,r}w$; $S_{0,r}p$;
$S_{0,l2}w$; $S_{0,l2}p$; $S_{0,r2}w$; $S_{0,r2}p$;
S_1w; S_1p; $S_{1,l}w$; $S_{1,l}p$; $S_{1,r}w$; $S_{1,r}p$;
$S_{1,l2}w$; $S_{1,l2}p$; $S_{1,r2}w$; $S_{1,r2}p$;
Bigram
$S_0wS_{0,l}w$; $S_0wS_{0,l}p$; $S_0pS_{0,l}w$; $S_0pS_{0,l}pS_{0,l}p$;
$S_0wS_{0,r}w$; $S_0wS_{0,r}p$; $S_0pS_{0,r}w$; $S_0pS_{0,r}pS_{0,r}p$;
$S_1wS_{1,l}w$; $S_1wS_{1,l}p$; $S_1pS_{1,l}w$; $S_1pS_{1,l}pS_{1,l}p$;
$S_1wS_{1,r}w$; $S_1wS_{1,r}p$; $S_1pS_{1,r}w$; $S_1pS_{1,r}pS_{1,r}p$;
S_0wS_1w; S_0wS_1p; S_0pS_1w; S_0pS_1p;
Trigram
$S_0wS_0pS_{0,l}w$; $S_0wS_{0,l}wS_{0,l}p$; $S_0wS_0pS_{0,l}p$;
$S_0pS_{0,l}wS_{0,l}p$; $S_0wS_0pS_{0,r}w$; $S_0wS_{0,l}wS_{0,r}p$;
$S_0wS_0pS_{0,r}p$; $S_0pS_{0,r}wS_{0,r}p$;
$S_1wS_1pS_{1,l}w$; $S_1wS_{1,l}wS_{1,l}p$; $S_1wS_1pS_{1,l}p$;
$S_1pS_{1,l}wS_{1,l}p$; $S_1wS_1pS_{1,r}w$; $S_1wS_{1,l}wS_{1,r}p$;
$S_1wS_1pS_{1,r}p$; $S_1pS_{1,r}wS_{1,r}p$;
Linearization
w_0; p_0; $w_{-1}w_0$; $p_{-1}p_0$; $w_{-2}w_{-1}w_0$; $p_{-2}p_{-1}p_0$;
$S_{0,l}S_{0,l2}w$; $S_{0,l}pS_{0,l2}p$; $S_{0,r2}wS_{0,r}w$; $S_{0,r2}pS_{0,r}p$;
$S_{1,l}S_{1,l2}w$; $S_{1,l}pS_{1,l2}p$; $S_{1,r2}wS_{1,r}w$; $S_{1,r2}pS_{1,r}p$;

Table 1: Feature templates.

name	domain	# of sents	# of tokens
training data			
AFP	News	35,390,025	844,395,322
XIN	News	18,095,371	401,769,616
WSJ	Finance	39,832	950,028
testing data			
WSJ	Finance	2,416	56,684
WPB	News	2,000	43,712
SANCL	Blog	1,015	20,356

Table 2: Data.

Training. We train *N*-gram language models from raw text using modified Kneser-Ney smoothing without pruning. The text is true-case tokenized, and we train 4-gram language modes using KenLM[2], which gives high efficiencies in standard *N*-gram language model construction.

3 Experimental settings

3.1 Data

For training data, we use the Wall Street Journal (WSJ) sections 1-22 of the Penn Treebank (Mar-

[2]https://kheafield.com/code/kenlm/

domain	sentence example
Finance	The $ 409 million bid includes the assumption of an estimated $ 300 million in secured liabilities on those properties , according to those making the bid.
News	But after rising steadily during the quarter-century following World War II , wages have stagnated since the manufacturing sector began to contract .
Blog	The freaky thing here is that these bozos are seriously claiming the moral high ground ?

Table 3: Domain examples.

cus et al., 1993), and the Agence France-Presse (AFP) and Xinhua News Agency (XIN) subsets of the English Giga Word Fifth Edition (Parker et al., 2011). As the development data, we use WSJ section 0 for parameter tuning. For testing, we use data from various domain, which consist of WSJ section 23, Washington Post/Bloomberg(WPB) subsets of the English Giga Word Fifth Edition and SANCL blog data, as shown in Table 2. Example sentence in various test domains are shown in Table 3.

3.2 Evaluation metrics

We follow previous work and use the BLEU metric (Papineni et al., 2002) for evaluation. Since BLEU only scores *N*-gram precisions, it can be in favour of *N*-gram language models. We additionally use METEOR[3](Denkowski and Lavie, 2010) to evaluate the system performances. The BLEU metric measures the fluency of generated sentence without considering long range ordering. The METEOR metric can potentially fix this problem using a set of mapping between generated sentences and references to evaluate distortion. The following example illustrates the difference between BLEU and METEOR on long range reordering, where the reference is

(1) *[The document is necessary for developer ,]$_0$ [so you can not follow this document to get right options .]$_1$*

and the generated output sentence is

(2) *[so you can not follow this document to get right options .]$_1$ [The document is necessary for developer ,]$_0$.*

There is a big distortion in the output. The BLEU metric gives a score of 90.09 out of 100, while the METEOR gives a score of 61.34 out of 100. This is because that METEOR is based on explicit word-to-word matches over the whole sentence. For word ordering, word-to-word matches are unique, which facilitates METEOR evaluation between generated sentences and references. As can bee seen from the example, long range distortion can highly influence the METEOR scores making the METEOR metric more suitable for evaluating word ordering distortions.

[3]http://www.cs.cmu.edu/∼alavie/METEOR/

ID	# training sent	# iter	Avg F1
set57	900	1	57.31
set66	1800	1	66.82
set78	9000	1	78.73
set83	all	1	83.93
set88	all	30	88.10

Table 4: Parsing accuracy settings.

3.3 Data preparation

For all the experiments, we assume that the input is a bag of words without order, and the output is a fully ordered sentence. Following previous work (Wan et al., 2009; Zhang, 2013; Liu et al., 2015), we treat base noun phrases (i.e. noun phrases do not contains other noun phrases, such as '*Pierre Vinken*' and '*a big cat*') as a single word. This avoids unnecessary ambiguities in combination between their subcomponents.

The syntactic model requires that the training sentences have syntactic dependency structure. However, only the WSJ data contains gold-standard annotations. In order to obtain automatically annotated dependency trees, we train a constituent parser using the gold-standard bracketed sentences from WSJ, and automatically parse the Giga Word data. The results are turned into dependency trees using Penn2Malt[4], after base noun phrases are extracted. In our experiments, we use ZPar[5] (Zhu et al., 2013) for automatic constituent parsing.

In order to study the influence of parsing accuracy of the training data, we also use ten-fold jackknifing to construct WSJ training data with different accuracies. The data is randomly split into ten equal-size subsets, and each subset is automatically parsed with a parser trained on the other

[4]http://stp.lingfil.uu.se/∼nivre/research/Penn2Malt.html

[5]http://people.sutd.edu.sg/∼yue_zhang/doc/doc/conparser.html

	in-domain on WSJ test		cross-domain on WPB test		cross-domain on SANCL test	
	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)
syntax-set57	48.76	48.98	37.31	46.78	37.60	46.79
syntax-set66	48.79	48.98	37.52	46.81	38.28	46.90
syntax-set78	49.27	49.08	38.10	46.89	38.76	46.96
syntax-set83	49.74	49.16	37.68	46.84	38.67	46.93
syntax-set88	49.73	49.17	38.27	46.92	38.52	46.93
syntax-gold	50.82	49.33	37.76	46.84	39.97	47.26

Table 5: Influence result of parsing accuracy.

nine subset. In order to obtain datasets with different parsing accuracies, we randomly sample a small number of sentences from each training subset, as shown in Table 4. The dependency trees of each set are derived from these bracketed sentences using Penn2Malt after base noun phrase are extracted as a single word.

4 Influence of parsing accuracy

4.1 In-domain word ordering

We train the syntactic models on the WSJ training parsing data with different accuracies. The WSJ development data are used to find out the optimal number of training iterations for each experiments, and the WSJ test results are shown in Table 5.

Table 5 shows that the parsing accuracy can affect the performance of the syntactic model. A higher parsing accuracy can lead to a better syntactic language model. It conforms to the intuition that syntactic quality affects the fluency of surface texts. On the other hand, the influence is not huge, the BLEU scores decrease by 1.0 points as the parsing accuracy decreases from 88.10% to 57.31%

4.2 Cross-domain word ordering

The influence of parsing accuracy of the training data on cross-domain word ordering is measured by using the same training settings, but testing on the WPB and SANCL test sets. Table 5 shows that the performance on cross-domain word ordering cannot reach that of in-domain word ordering using the syntactic models. Compared with the cross-domain experiments, the influence of parsing accuracy becomes smaller. In the WPB test, the fluctuation of performance decline to about 0.9 BLEU points, and in the SANCL test, the fluctuation is about 1.1 BLEU points.

In conclusion, the experiments show that parsing accuracies have a relatively small influence on the syntactic models. This suggests that it is possible to use large automatically-parsed data to train syntactic models. On the other hand, when the training data scale increases, syntactic models can become much slower to train compared with *N*-gram models. The influence on data scale, which includes output quality and training time, is further studied in the next section.

5 Influence of data scale

We use the AFP news data as the training data for the experiments of this section. The syntactic models are trained using automatically-parsed trees derived from ZPar, as described in Section 3.3. The WPB test data is used to measure in-domain performance, and the SANCL blog data is used to measure cross-domain performance.

5.1 Influence on BLEU and METEOR

The Figure 3 and 4 shows that using both the BLEU and the METEOR metrics, the performance of the syntactic model is better than that of the *N*-gram models. It suggests that sentences generated by the syntactic model have both better fluency and better ordering. The performance of the syntactic models is not highly weakened in cross-domain tests.

The grey dot in each figure shows the performance of the syntactic model trained on the gold WSJ training data, and evaluated on the same WPB and SANCL test data sets. A comparison between the grey dots and the dashed lines shows that the syntactic model trained on the WSJ data perform better than the syntactic model trained on similar amounts of AFP data. This again shows the effect of syntactic quality of the training data.

On the other hand, as the scale of automatically-parsed AFP data increases, the performance of the

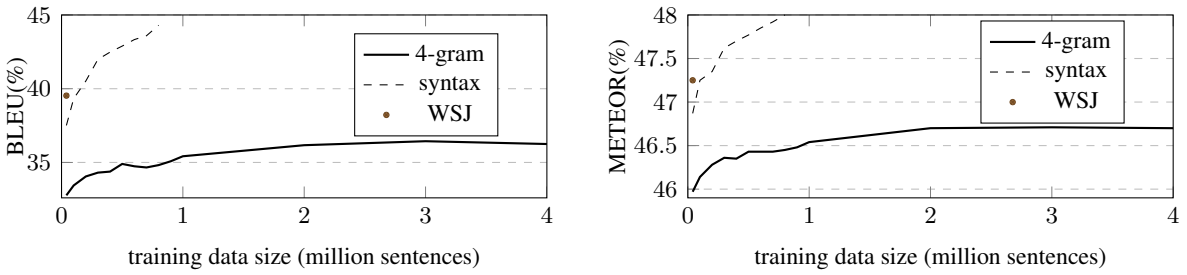

Figure 3: In-domain results on different training data size.

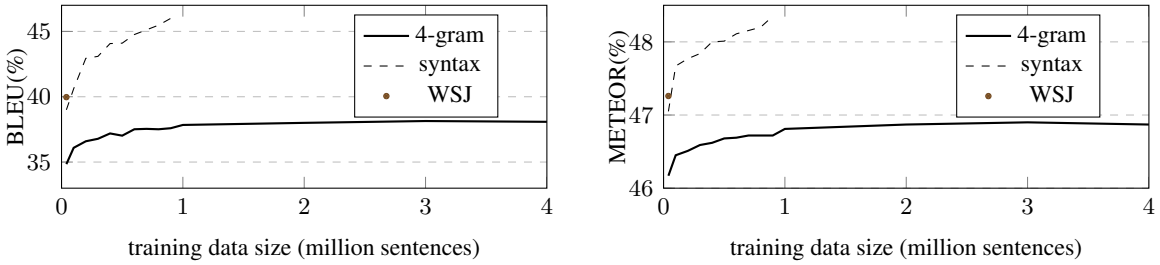

Figure 4: Cross-domain results on different training data sizes.

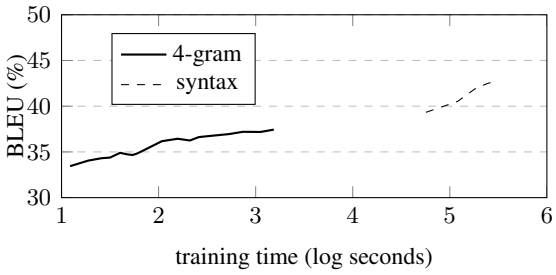

Figure 5: BLEU on different training times.

syntactic model rapidly increases, surpassing the syntactic model trained on the high-quality WSJ data. This observation is important, showing that large-scale data can be used to alleviate the problem of lower syntactic quality in automatically-parsed data, which can be leveraged to address the scarcity issue of manually annotated data in both in-domain and cross-domain settings.

5.2 Influence on training time

The training time of both syntactic models and *N*-gram models increases as the size of training data increases. Figure 5 shows the BLEU of the two systems under different amounts of training time. There is no result reported for the syntactic model beyond 1 million training sentences, because training becomes infeasibly slow [6]. On the other hand, the *N*-gram model can be trained using all the WSJ, AFP, XIN training sentences, which are 53 millions, within $10^{3.2}$ seconds. As a result, there is no overlap between the syntactic model and the *N*-gram model curves.

As can be seen from the figure, the syntactic model is much slower to train. However, it benefits more from the scale of the training data, with the slope of the dashed curve being steeper than that of the solid curve. The *N*-gram model can be trained with more data thanks to the fast training speed. However, the performance of the *N*-gram model flattens when the training data size reaches beyond 3 million. Projection of the solid curve suggests that the performance of the *N*-gram model may not surpass that of the syntactic model even if sufficiently large data is available for training the *N*-gram model in more time.

6 Error analysis

Although giving overall better performance, the syntactic model does not perform better than the *N*-gram model in all cases. Here we analyze the strength of each model via more fine-grained comparison.

In this set of experiments, the syntactic model is trained using gold-standard annotated WSJ training parse trees, and the *N*-gram model is trained using the data containing WSJ training data, AFP and XIN. The WSJ test data, which contains

[6] Our experiments are carried on a single thread of 3.60GHz CPU. If the training time is over 90 hours for a model, we consider it infeasible.

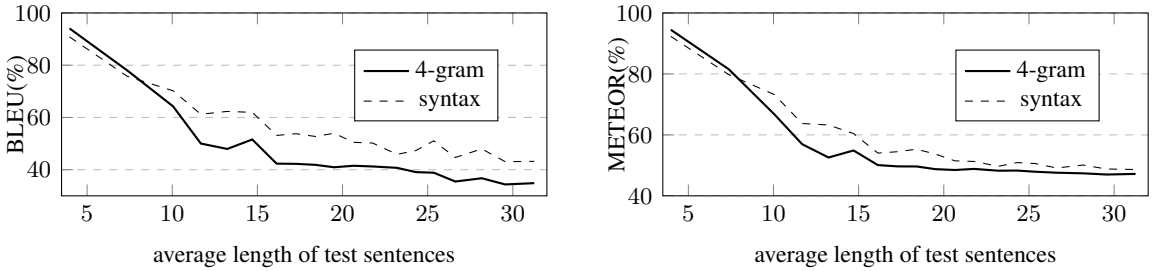

Figure 6: Performance on sentences with different length.

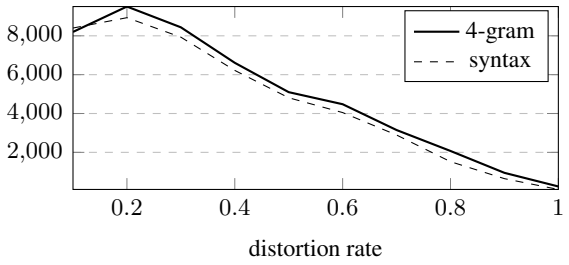

Figure 7: The distribution of distortion.

Template	distribution
NLM-LOW	set 1 if $p < e^{-12.5}$, else 0
NLM-20	use 20 bins to scatter probability
NLM-10	use 10 bins to scatter probability
NLM-5	use 5 bins to scatter probability
NLM-2	use 2 bins to scatter probability

Table 6: NLM feature templates.

golden constituent trees, is used to analyze errors in different aspects.

6.1 Sentence length

The BLEU and METEOR scores of the two systems on various sentence lengths are shown in Figure 6. The results are measured by binning sentences according to their lengths, so that each bin contains about the same number of sentences. As shown by the figure, the *N*-gram model performs better on short sentences (less than 8 tokens), and the syntactic model performs better on longer sentences. This can be explained by the fact that longer sentences have richer underlying syntactic structures, which can better captured by the syntactic model. In contrast, for shorter sentences, the syntactic structure is relatively simple, and therefore the *N*-gram model can give better performance based on string patterns, which form smaller search spaces.

6.2 Distortion range

We measure the average distortion rate of output word w using the following metric:

$$distortion(w) = \frac{|i_w - i'_w|}{len(S_w)},$$

where i_w is index of word w in the output sentence S_w, i'_w is the index of the word w in the reference sentence. $len(S_w)$ is the number of tokens in sentence S_w. Figure 7 shows distributions of distortion respectively by the syntactic and *N*-gram model. The *N*-gram model makes relatively fewer short-range distortions, but more long-range distortions. This can be explained by the local scoring nature of the *N*-gram model. In contrast, the syntactic model makes less long-range distortions, which can suggest better sentence structure.

6.3 Constituent span

We further evaluate sentence structure correctness by evaluating the recalls of discovered constituent span in output two systems, respectively. As shown in Figure 8. The syntactic model performs better in most constituent labels. However, the *N*-gram model performs better in WHPP, SBARQ and WHNP.

In the test data, WHPP, SBARQ and WHNP are much less than PP, NP, VP, ADJP, ADVP and CONJP, on which the syntactic model gives better recalls. WHNP spans are small and most of them consist of a question word (WP$) and one or two nouns (e.g. “whose (WP$) parents (NNS)”). WHPP spans are also small and usually consist of a preposition (IN) and a WHNP span (e.g “at (IN) what level (WHNP)”). The *N*-gram model performs better on these small spans. The syntactic model also performs better on S, which covers the whole sentence structure. This verifies the hypothesis introduce that syntactic language models better capture overall sentential grammaticality.

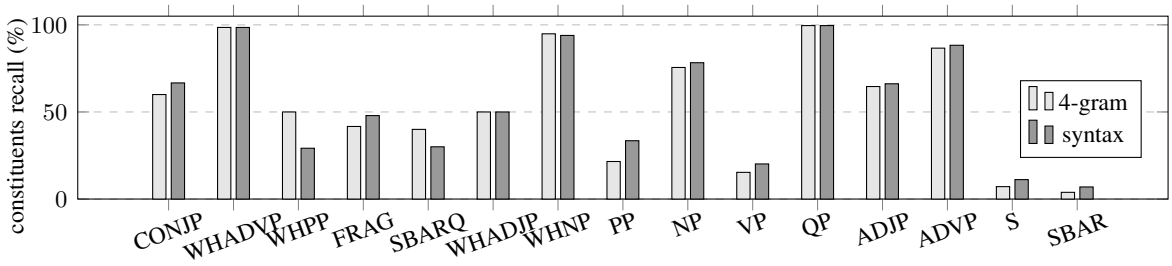

Figure 8: Recalls of different constituents.

	in-domain on WSJ test		cross-domain on WPB test		cross-domain on SANCL test		# of
	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)	sent/s
syntax	50.82	49.33	37.76	46.84	39.97	47.26	17.9
4-gram	42.26	48.00	37.71	46.90	39.72	47.08	**177.0**
combined	**52.38**	**49.66**	**39.12**	**47.07**	**40.60**	**47.38**	15.4

Table 7: Final results on various domains.

7 Combining the syntactic and *N*-gram models

The results above show the respective error characteristics of each model, which are complimentary. This suggests that better results can be achieved by model combination.

7.1 *N*-gram language model feature

We integrate the two types of models by using *N*-gram language model probabilities as features in the syntactic model. *N*-gram language model probabilities, which ranges from 0 to 1. Direct use of real value probabilities as features does not work well in our experiments, and we use discretized features instead. For the L-ARC and R-ARC actions, because no words are pushed onto the stack, The NLM feature is set to NULL by default. For the SHIFT action, different feature values are extracted depending on the NLM from 0 to 1.

In order to measure the *N*-gram probabilities on our data, we train the 4-gram language model WSJ, AFP and XIN data, and randomly sample 4-gram probabilities from the syntactic model output on the WSJ development data, finding that most of 4-gram probabilities p are larger than $10^{-12.5}$. In this way, if p lower than $10^{-12.5}$, NLM feature value is set to LOW. As for p larger than $10^{-12.5}$, we extract the discrete features by assigning them into different bins. We bin the 4-gram probabilities with different granularities without overlap features. As shown in Table 6, NLM-20, NLM-10, NLM-5 and NLM-2 respectively use 20, 10, 5 and 2 bins to capture NLM feature values.

	BLEU (%) on WSJ test
Wan et al. (2009)	33.70*
Zhang and Clark (2011b)	40.10*
Zhang et al. (2012)	43.80*
Zhang (2013)	44.70
syntax (Liu et al., 2015)	50.82
4-gram	42.26
combined	**52.38**

Table 8: Final results of all systems, where "*" means that the system uses extra POS input.

7.2 Final results

We use the WSJ, AFP and XIN for training the *N*-gram model[7]. The same WSJ, WPB and SANCL test data are used to measure performances on different domains.

The experimental results are shown in Tables 7 and 8. In both in-domain and cross-domain test data, the combined system outperforms all other systems, with a BLEU score of 52.38 been achieved in the WSJ domain. It would be overly expensive to obtain a human oracle on discusses. However, according to Papineni (2002), a BLEU

[7]For the combined model, we used the WSJ training data for training, because the syntactic model is slower to train using large data. However, we did a set of experiments to scale up the training data by sampling 900k sentences from AFP. Results show that the combined model gives BLEU scores of 42.86 and 44.44 on the WPB and SANCL tests, respectively. Cross-domain BLEU on WSJ, however falls to 49.84.

BLEU	sentences
ref	For weeks , the market had been nervous about takeovers , after Campeau Corp. 's cash crunch spurred concern about the prospects for future highly leveraged takeovers .
41.37	For weeks , Campeau Corp. 's cash had the prospects for takeovers after the market crunch spurred concern about future highly leveraged takeovers , nervous been about .
ref	Now , at 3:07 , one of the market 's post-crash " reforms " took hold as the S&P 500 futures contract had plunged 12 points , equivalent to around a 100-point drop in the Dow industrials .
51.39	Now , one of the market 's reforms plunged 12 points in the Dow industrials as " post-crash , the S&P 500 futures contract , equivalent to 3:07 took hold at around a 100-point drop had . "
ref	Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .
64.38	Canadian Utilities , Alberta , where the company had 1988 revenue of C$ 1.16 billion in its natural gas and electric utility businesses serves mainly from about 800,000 customers .

Table 9: Output samples.

score of over 52.38 indicate an easily understood sentence. Some sample outputs with different BLEU scores are shown in Table 9

In addition, Table 7 shows that the *N*-gram model is the fastest among the models due to its small search space. The running time of the combined system is larger than the pure syntactic system, because of *N*-gram probability computation. Table 8 compare our results with different previous methods on word ordering. Our combined model gives the best reported performance on this standard benchmarks.

8 Conclusion

We empirically compared the strengths and error distributions of syntactic and *N*-gram language models on word ordering, showing that both can benefit from large-scale raw text. The influence of parsing accuracies has relatively small impact on the syntactic language model trained on automatically-parsed data, which enables scaling up of training data for syntactic language models. However, as the size of training data increases, syntactic language models can become intolerantly slow to train, making them benefit less from the scale of training data, as compared with *N*-gram models.

Syntactic models give better performance compared with *N*-gram models, despite trained with less data. On the other hand, the two models lead to different error distributions in word ordering. As a result, we combined the advantages of both systems by integrating a syntactic model trained with relatively small data and an *N*-gram model trained with relatively large data. The resulting model gives better performance than both single models and achieves the best reported scores in a standard benchmark for word ordering.

We release our code under GPL at `https://github.com/SUTDNLP/ZGen`. Future work includes application of the system on text-to-text problem such as machine translation.

Acknowledgments

The research is funded by the Singapore ministry of education (MOE) ACRF Tier 2 project T2MOE201301. We thank the anonymous reviewers for their detailed comments.

References

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*, pages 40–46. Citeseer.

Wenliang Chen, Min Zhang, and Haizhou Li. 2012. Utilizing dependency language models for graph-based dependency parsing models. In *Proceedings of ACL*, pages 213–222.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, page 111.

A De Gispert, M Tomalin, and W Byrne. 2014. Word ordering with phrase-based grammars. In *Proceedings of EACL*, pages 259–268.

Michael Denkowski and Alon Lavie. 2010. Extending the meteor machine translation evaluation metric to the phrase level. In *HLT/NAACL*, pages 250–253.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule. Technical report, DTIC Document.

Yuqing Guo, Haifeng Wang, and Josef Van Genabith. 2011. Dependency-based n-gram models for general purpose sentence realisation. *Natural Language Engineering*, 17(04):455–483.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 48–54.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *Proceedings of NAACL/HLT*, pages 113–122, Denver, Colorado, May–June.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07*.

Lane Schwartz, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental syntactic language models for phrase-based translation. In *Proceedings of ACL/HLT*, pages 620–631.

Libin Shen, Jinxi Xu, and Ralph M Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*, pages 577–585.

Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of EACL*, pages 852–860.

Yue Zhang and Stephen Clark. 2011a. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Yue Zhang and Stephen Clark. 2011b. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of EMNLP*, pages 1147–1157.

Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of EACL*, pages 736–746. Association for Computational Linguistics.

Yue Zhang, Kai Song, Linfeng Song, Jingbo Zhu, and Qun Liu. 2014. Syntactic smt using a discriminative text generation model. In *Proceedings of EMNLP*, pages 177–182, Doha, Qatar, October.

Yue Zhang. 2013. Partial-tree linearization: generalized word ordering for text synthesis. In *Proceedings of IJCAI*, pages 2232–2238. AAAI Press.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL*, pages 434–443.