Joint Morphological Generation and Syntactic Linearization

Linfeng Song[1*], **Yue Zhang**[2], **Kai Song**[4] **and Qun Liu**[3,1]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Science
[2]Singapore University of Design and Technology
[3]CNGL, School of Computing, Dublin City University
[4]Natrual Language Processing Lab., Northeastern University
songlinfeng@ict.ac.cn; yue_zhang@sutd.edu.sg; songkai2012@yeah.net; qliu@computing.dcu.ie

Abstract

There has been growing interest in stochastic methods to natural language generation (NLG). While most NLG pipelines separate morphological generation and syntactic linearization, the two tasks are closely related. In this paper, we study joint morphological generation and linearization, making use of word order and inflections information for both tasks and reducing error propagation. Experiments show that the joint method significantly outperforms a strong pipelined baseline (by 1.1 BLEU points). It also achieves the best reported result on the Generation Challenge 2011 shared task.

Introduction

There has been a significant growth of interest in stochastic (corpus-based) natural language generation (NLG) (Bohnet et al. 2010; Wan et al. 2009; Bangalore, Rambow, and Whittaker 2000; Oh and Rudnicky 2000; Langkilde and Knight 1998). Such methods often treat the problem of NLG as a pipeline of several independent steps. For example, shown in Figure 1(a), a pipeline based on the meaning text theory (MTT) splits NLG into three independent steps (Bohnet et al. 2010): 1. *syntactic generation*: generating an unordered and lemma-formed syntactic tree from a semantic graph; 2. *syntactic linearization*: linearizing the unordered syntactic tree; 3. *morphological generation*: generating the inflection for each lemma in the string.

Although treated as separated steps, the tasks of *syntactic linearization* and *morphological generation* strongly interact with each other. Take the bag-of-lemmas *[John, honest, nice, and, a, friend, be]* for example. On the one hand, if we know that the correct word order is *"John be a honest and nice friend"*, we can easily generate the correct inflection *"an"* for the lemma *"a"*. On the other hand, if we know that the correct inflection for the lemma *"a"* is *"an"*, we can easily infer that the correct word ordering of the NP-phrase is *"an honest and nice friend"* rather than *"an nice and honest friend"*. Thus jointly performing the two tasks in one single step can lead to better use of information and reduction of error propagation.

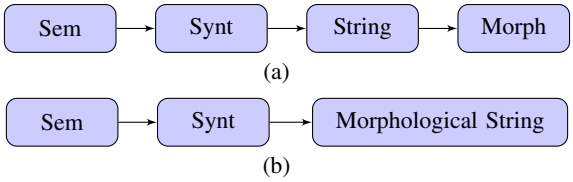

Figure 1: (a) NLG pipeline based on the meaning text theory (Bohnet et al. 2010), (b) pipeline of this paper. *Sem* stands for Semantics, *Synt* stands for syntax and *Morph* stands for morphology.

Recent years have witnessed a great success of joint methods on natural language processing problems, such as joint chunking and part-of-speech tagging (Sutton, McCallum, and Rohanimanesh 2007), joint Chinese word segmentation and part-of-speech tagging (Zhang and Clark 2008; Jiang et al. 2008; Sun 2011), joint named entity recognition and parsing (Finkel and Manning 2009), joint Chinese word segmentation, part-of-speech tagging and parsing (Zhang et al. 2013; Hatori et al. 2012) and joint morphological tagging and parsing (Bohnet et al. 2013). These methods tackle larger combined search spaces, while reaping benefits from richer sources of information.

In this paper, we study the task of joint syntactic linearization and morphological generation for NLG, taking the meaning text theory pipeline of Figure 1(a) as a baseline. We adapt the method of Zhang (2013) to develop a linearization system. As for morphological generation, we first adopt a small set of rules to get candidate inflections for each lemma, and then use a statistical model to select the best inflection as the result. We develop two pipelined baselines: one treating morphological generation as a postprocessing module to linearization, as the MTT pipeline of Bohnet et al. (2010) does, the other treating it as a preprocessing module to linearization. The two baseline systems make different uses of information. While the former benefits morphological generation by word order information, the latter benefits linearization by readily available inflections. Both, however, suffer from the error propagation problem.

Based on the learning and search algorithms of Zhang and Clark (2011), we develop a joint method for morphological generation and linearization (Figure 1(b)). The joint search algorithm takes every candidate inflection as a leaf hypoth-

[*]Work done when the first author was a visiting student at Singapore University of Technology and Design.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

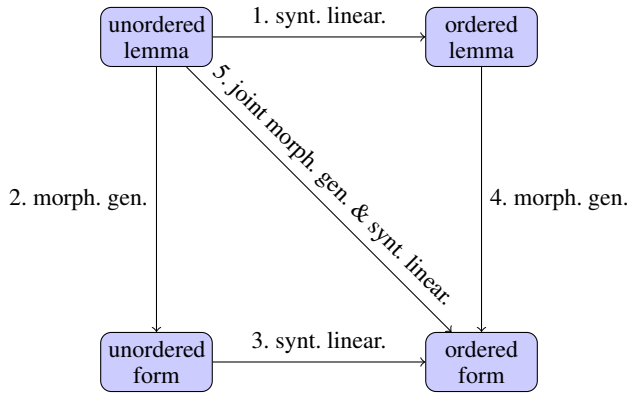

Figure 2: Several pipelines for syntactic linearization (*synt. linear.*) and morphological generation (*morph. gen.*).

esis, and builds an ordered and inflected tree. To ensure that the algorithm selects only one inflection for each lemma, we make candidate inflections for each lemma mutually exclusive. The disambiguation problem is left to the search algorithm, which is designed to tackle NP-complete tasks.

Our experiments are conducted on the dataset from the Generation Challenges 2011 Surface Realisation Shared Task, which focuses on the syntactic realization and morphological generation tasks in the MTT NLG pipeline. We find that performing morphological generation *after* syntactic linearization gives slightly better outputs than performing morphological generation *before* linearization, while the joint method significantly outperforms the pipelined baselines, achieving the best reported result on the shared task.

Our contributions can be summarized as:

- we show that joint syntactic linearization and morphological generation can lead to significant improvement over pipelined methods for natural language generation (by more than 1.1 BLEU points, from 88.48 → 89.62).
- we investigate the order of morphological generation and linearization in a pipeline, systematically analyzing the merits and shortcomings of both pipelined orders.
- we evaluate the dependency-based search architecture of Zhang (2013) on the Generation Challenge 2011 shared task, and report the best results on this task.

Baseline

Figure 2 shows three different ways to perform syntactic linearization and morphological generation. The path 1→4, used by Bohnet et al. (2010), treats morphological generation as a postprocessing step to syntactic linearization. On the other hand, the path 2→3 takes morphological generation as a preprocessing step to linearization. The path 5 joins morphological generation and syntactic linearization.

Bohnet et al. (2010) have shown the effectiveness of the path 1→4, while intuitively the path 2→3 can also be effective. Although the path 2→3 may lose some accuracy during morphological generation since it can not use word order information, it can utilize morphological information

Rules for *be*
pos==VB → be
pos==VBZ → is
pos==VBG or attr[‘partic’] == ‘pres’ → being
pos==VBN or attr[‘partic’] == ‘past’ → been
pos==VBD or attr[‘tense’] == ‘past’
sbj.attr[‘num’] == ‘sg’ → was
sbj.attr[‘num’] == ‘pl’ → were
other → [was,were]
pos==VBP or attr[‘tense’] == ‘pres’
sbj.attr[‘num’] == ‘sg’ → is
sbj.attr[‘num’] == ‘pl’ → are
other → [am,is,are]
Rules for *other verbs*
pos==VB → lemma
pos==VBZ → Dict.get(lemma,VBZ)
pos==VBG or attr[‘partic’] == ‘pres’ → Dict.get(lemma,VBG)
pos==VBN or attr[‘partic’] == ‘past’ → Dict.get(lemma,VBN)
pos==VBD or attr[‘tense’] == ‘past’ → Dict.get(lemma,VBD)
pos==VBP or attr[‘tense’] == ‘pres’
sbj.attr[‘num’] == ‘sg’ → Dict.get(lemma,VBZ)
other → Dict.getall(lemma)
Rules for other types
lemma==a → [a,an]
lemma==not → [not,n’t]
others → Dict.get(lemma,pos)

Table 1: Lemma rules. All rules are in the format: *conditions* → *candidate inflections*. Nested conditions are listed in multi-lines with indentation. *Dict* is an inflection dictionary. For detailed formatting information, refer to Table 4.

and ngram language models during syntactic linearization, which the path 1→4 can not utilize. In this paper, we treat both methods as baseline systems.

Morphological Generation

Four major types of lemmas need morphological generation, including: *nouns*, *verbs*, *articles* and *adjectives*. We use a statistical method to transform these types of lemmas in a lemma-based dependency tree into morphological forms. Given a lemma lem, the model selects the inflection with the highest score from a set of candidates.

$$y = \underset{m \in Morph(lem)}{\arg\max} \overrightarrow{\Theta} \cdot f(m)$$

Here $\overrightarrow{\Theta}$ is the vector of weights and $f(m)$ is the feature vector of the inflection m. For generating a set of candidate morphological forms for each lemma ($Morph(lem)$), we use a dictionary and write a small set of rules, which are listed in Table 1. Since the rules are based on syntactic annotations of the training corpus, we can safely prune impossible inflections for each lemma in the data. For example, if we know the part-of-speech of the lemma *be* is *VBD* (past tense verb), then inflections such as *am*, *is* and *are* can be safely pruned. The averaged perceptron (Collins 2002) is used to train the classifier for each lemma *lem*.

We adopt the syntactic feature templates of Zhang (2013) for morphological generation of the path 1→4 in Figure 2. However, since the system of 2→3 cannot use word order

features for preprocessing candidate inflections generation
WORD(h) · POS(h); WORD(h) · LABEL(h);
WORD(h) · POS(h) · LABEL(h);
WORD(h) · WORD(p); WORD(h) · WORD(p) · WORD(pp);
WORD(h) · POS(p); WORD(h) · POS(p) · POS(pp);
For each modifier m of h:
WORD(h) · WORD(m); WORD(h) · POS(m);

Table 2: Feature templates for the morphological generation system of the path 2→3 in Figure 2. Indices on the surface string: h – head; p – parent of h; pp – grandparent of h; m – modifier of h. Functions: WORD – word at index; POS – part-of-speech at index; LABEL – dependency label at index.

information in morphological generation, we delete the features that are sensitive to word order and add additional features, resulting in the templates shown in Table 2.

Bohnet et al. (2010) use an edit-distance-script-based method for morphological generation. They treat the edit script (such as "insert *ing*" or "remove *e* and insert *ing*") as the output classes for each lemma, in order to reduce the influence of unseen words. However, they have to consider hundreds of classes for each lemma, which is unnecessary since some transformation rules can not be applied to certain lemmas (such as "inserting *ing*" does not apply to nouns). We choose to use a dictionary and hand-written rules to find a small set of candidates for each lemma as its classes.

Syntactic Linearization

Given an unordered dependency tree d, the task of linearization is to find the highest-scored ordered tree y,

$$y = \underset{y' \in Order(d)}{\arg\max} \overrightarrow{\Theta} \cdot \phi(y')$$

We adapt the best first search framework of Zhang (2013) to perform syntactic linearization. Our algorithm uses an *agenda* (a priority queue) and a *chart* (a large beam) to perform best first search. Both structures order hypotheses by the score. Shown in Algorithm 1, the search algorithm initializes the agenda (Line 4) by building a hypothesis h for each input word and pushing h onto *agenda* (Lines 21–24). Here a hypothesis h is a projective sub dependency tree, which can be a leaf node. Then at each step, it pops out the hypothesis h with the highest model score from *agenda*, trying to combine h with each hypothesis h_c on *chart* as long as there is no collision and the head of h_c depends on the head of h. All the combinations are put back onto *agenda*, and h is added onto *chart* (Lines 5–16). Following the previous work on best first generation (White 2004), we set a timeout threshold after which the search process is terminated, and the highest-scored chart hypothesis is used for output.

We adopt the feature templates of Zhang (2013). Shown in Table 3, we use additional features, including 4-gram words and 4-gram part-of-speech tags. Inspired by Shen et al. (2008), we also adopt 4-gram dependency features for words, part-of-speech tags and dependency labels.

Our linearization system uses the online learning algorithm of Zhang and Clark (2011), which is based on the decoding process. At each time, the algorithm examines the newly dequeued hypothesis h from *agenda*. If h is not a gold-standard hypothesis (a gold sub tree), it is treated as a negative example, the lowest-scored gold-standard hypothesis g_{min} on the agenda is taken as a positive example, and the rest of the decoding step is replaced with a parameter update (Crammer et al. 2006). Similarly, if a gold hypothesis is pruned from *chart* due to beam search, it is treated as a positive example, and the highest-scored non-gold hypothesis on *chart* is treated as a negative example, so that the parameters are adjusted. The gold hypothesis is put back onto *chart* after the update, to ensure that the gold output can be recovered. Intuitively, these updates ensure that gold hypotheses have higher scores than non-gold ones, and can survive pruning while being expanded early.

Algorithm 1: Decoding algorithm for linearization.

Input : words – the input words. If morphological generation serves as postprocessing, then the inputs are lemmas.
links – dependency links between words
Output: An ordered sentence with syntax tree.

```
1  agenda ← CreatePriorityQueue();
2  chart ← CreateBeam();
3  goals ← CreateVector();
4  InitAgenda (words,agenda);
5  while not TimeOut() and Empty(agenda) == False
   do
6      h ← Pop(agenda);
7      if Finished(h) then
8          AddTo(goals, h);
9      else
10         for h_c ∈ chart do
11             if NoCollision(h, h_c) and
               h_c.IsDependentOn(h,links) then
12                 h_r ← Combine(h, h_c, DepLeft);
13                 Push(agenda, h_r);
14                 h_r ← Combine(h, h_c, DepRight);
15                 Push(agenda, h_r);
16         AddTo(chart, h);
17 if Empty(goals) then
18     return HighestScore(chart);
19 else
20     return HighestScore(goals);
21 Function InitAgenda (words, agenda)
22     for word ∈ words do
23         h ← BuildLeafHypothesis(word);
24         Push(agenda, h);
```

The Joint Method

There are disadvantages for both baseline methods. On the one hand, the path 1→4 in Figure 2 cannot utilize morphological information of ngram language models during syntactic linearization. In addition, errors made by syntactic linearization will propagate to morphological generation. On

Dependency N-gram Features
$\text{WORD}(h) \cdot \text{WORD}(m) \cdot \text{WORD}(s) \cdot dir$,
$\text{WORD}(h) \cdot \text{WORD}(m) \cdot \text{WORD}(s) \cdot \text{WORD}(ss) \cdot dir$,
$\text{WORD}(h) \cdot \text{WORD}(m) \cdot \text{WORD}(ml)$,
$\text{WORD}(h) \cdot \text{WORD}(m) \cdot \text{WORD}(mr)$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(s) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(s) \cdot \text{POS}(ss) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(ml)$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(mr)$,
$\text{DEP}(h) \cdot \text{DEP}(m) \cdot \text{DEP}(s) \cdot dir$,
$\text{DEP}(h) \cdot \text{DEP}(m) \cdot \text{DEP}(s) \cdot \text{DEP}(ss) \cdot dir$,
$\text{DEP}(h) \cdot \text{DEP}(m) \cdot \text{DEP}(ml)$,
$\text{DEP}(h) \cdot \text{DEP}(m) \cdot \text{DEP}(mr)$,
4-gram Boundary Word & POS-tag features
$\text{WORD}(B-3) \cdot \text{WORD}(B-2)$, $\text{WORD}(B-1) \cdot \text{WORD}(B)$,
$\text{WORD}(B-2) \cdot \text{WORD}(B-1)$, $\text{WORD}(B) \cdot \text{WORD}(B+1)$,
$\text{WORD}(B-1) \cdot \text{WORD}(B)$, $\text{WORD}(B+1) \cdot \text{WORD}(B+2)$,
$\text{POS}(B-3) \cdot \text{POS}(B-2)$, $\text{POS}(B-1) \cdot \text{POS}(B)$,
$\text{POS}(B-2) \cdot \text{POS}(B-1)$, $\text{POS}(B) \cdot \text{POS}(B+1)$,
$\text{POS}(B-1) \cdot \text{POS}(B)$, $\text{POS}(B+1) \cdot \text{POS}(B+2)$,

Table 3: Feature templates. Indices on the surface string: h – head on newly added arc; m – modifier on arc; s – nearest sibling of m; ss – nearest sibling of s; ml, mr – left/rightmost modifier of m; B – boundary between the conjoined phrases (index of the first word of the right phrase). Variables: *dir* – direction of the arc. Functions: WORD – word at index; POS – part-of-speech at index; DEP – dependency label at index.

Algorithm 2: The Agenda Initiation Function for the Joint algorithm.

```
1  Function InitAgendaJoint (lemmas, agenda)
2      for lemma ∈ lemmas do
3          ID = GetID (lemma);
4          for cand in lemma.cands do
5              h ← BuildLeafHypothesis(cand,ID);
6              Push(agenda, h);
```

the other hand, the path 2→3 is limited by lack of word order information during morphological generation. The case of *"a"* in the introduction is one example.

Algorithm

Given a lemma-formed unordered dependency tree as input, the joint algorithm generates inflected and ordered candidates and selects the one with the highest score. Similar to the baselines, we use the rules in Table 1 to generate candidate inflections for each lemma for the joint system. For joint morphological generation and syntactic linearization, we replace the function *InitAgenda* (Lines 21–24 of Algorithm 1) with the function *InitAgendaJoint*, shown in Algorithm 2. We treat each candidate inflection as a leaf hypothesis and impose mutual exclusion by letting all candidate inflections of one lemma share the same ID. The IDs are different among different lemmas. If two hypotheses contain overlapping IDs (a hypothesis contains a set of IDs), they can not be combined into a larger hypothesis.

Input(unordered lemma-formed tree):

Lemma	POS	PID	Lemma	POS	Attributes
SROOT	1	0	play	VB	tense=pres
OBJ	2	1	Elianti	NNP	num=sg
P	3	1	.	.	
SBJ	4	1	Haag	NNP	num=sg
TITLE	5	4	Ms.	NNP	num=sg

Reference(ordered inflected sentence):
Ms. Haag plays Elianti .

Table 4: One training instance from Generation Challenge 2011. The column *DEP* represents the dependency label for each node, *ID* represents the node's unique ID within the dependency tree, *PID* represents the ID of the node's parent in the dependency tree, *Lemma* represents the lemma of the node, *POS* represents the part-of-speech of the node, and *Attributes* represents some attributes (such as partic, tense or number) of the node.

The search framework of Algorithm 1 is capable for joint decoding, because the learning-guided search can be used to tackle the large joint search space, and the baseline linearization features which cover ngram, part-of-speech, and dependency relations between lemmas, are capable for inflection disambiguation.

In addition, search space explosion, which is common for joint tasks, is not severe for joint syntactic linearization and morphological generation. One reason is that many ambiguities are local problems. Consider again the case in the introduction. There are 5 candidates *[am, is, are, was, were]* for the lemma *be*, 2 candidates *[a, an]* for the lemma *a* and 2 candidates *[friends, friend]* for the lemma *friend*. Although there are 20 possible ways of inflecting the three lemmas, most of them can be pruned out in hypothesis combinations. For example, by capturing that the lemma *a* depends on the lemma *friend*, the system knows that *friend* is singular and chooses its candidate inflection *friend*. Capturing that the only subject of *be* being single-3rd person *John* and the part-of-speech of the lemma *be* being *VBP* (verb, present tense), the system can get the correct morphological form *is* for the lemma *be*. Hence no hypothesis can contain the combination *John were a nice and honest friends*. The ambiguities of *[a, an]* can be solved by using ngram features.

Experiments

We perform experiments on the corpus for the Generation Challenges 2011 Surface Realisation Shared Task[1], which provides training and test data for the shallow syntax linearization to morphological generation pipeline. The corpus consists of unordered lemma sequences in the CoNLL format and the corresponding ordered morphological outputs. One example is shown in Table 4.

We obtain our training set from the official training data of this shared task, using this training set to train the morphological generation, syntactic linearization and joint systems. We use the training set and MulText[2] (Ide and Véronis 1994) to get the inflection dictionary mentioned in Table 1. We

[1]http://www.nltg.brighton.ac.uk/research/sr-task/
[2]http://catalog.elra.info/

test the three paths in Figure 2 on the official test set of this shared task. For development, we extract 1 out of every 20 sentences from the original training set as the development test set, and take the remaining training data as the development training set. Our development test set contains 1809 sentences and the development training set contains 34400 sentences. Following Bohnet et al. (2010), we use the gold annotations to train all the systems in the pipelined methods. We perform 10 training rounds for the morphological generation system, 20 iterations for the linearization system and 20 iterations for the joint system, according to development experiments. We set the timeout threshold of both the linearization system and the joint system to 8s for all tests and use the BLEU metric (Papineni et al. 2002) to evaluate the results. All the systems were implemented in Python, and all the experiments were performed on an Intel Xeon E5 2660 CPU with Centos 6.3 and Python 2.6.

Generating Reference Trees

Because our linearization system outputs entire dependency trees rather than surface strings, the training references must be ordered dependency trees. Shown in Table 4, since the training references of the shared task data contain only ordered surface strings, we obtain reference trees by generating the 1-to-1 correspondences between the input dependency trees and reference ordered strings. As the input trees are lemma-formed while the reference sentences are inflected, we first use a morphological transformation dictionary[3] to obtain possible morphological forms for each lemma in the training input, and then find all its possible correspondences in the reference (such as be $\rightarrow$ is, chase $\rightarrow$ chasing). Then we refine the correspondences in a bottom-up manner: assuming all the sub trees are projective, every continuous partial tree should be projected to a continuous phrase in the reference. We use this assumption to prune violating correspondence links between training input and reference and obtain a final 1-to-1 correspondence. If there are multiple mappings after pruning, we choose the first one. This is because all the mappings are identical, and whatever we choose has no influence for the training of the linearization system. Out of the total 39K training instances, 36.2K remain. We discard the 2.8K conflicting instances, which are either non-projective or illegal.

Development Results

Table 5 shows the detailed performance of each method. The column *Morph.* shows the precision of the morphological generation system on the development set, given an unordered tree (for the *Morph. Gen.+Synt. Linear.* and the *joint* systems) or syntactic linearization results (for the *Synt. Linear.+Morph. Gen.* system). It reflects the quality of inflection. The column *Linear.* shows the BLEU score of the syntactic linearization system on the development set, given a lemma-formed input (for the *Synt. Linear.+Morph. Gen.* and the *joint* systems) or morphological generation results (for the *Morph. Gen.+Synt. Linear.* system). It reflects the quality of ordering. The column *Overall* shows the BLEU score of the whole pipeline for each method.

[3]Download from http://www.lexically.net/downloads/BNC_wordlists/e_lemma.txt, it contains transformations for 14760 lemmas and has 1.75 morphological forms for each lemma on average.

Methods	Morph.	Linear.	Overall
Synt. Linear.+Morph. Gen.	97.27	92.31	90.44
Morph. Gen.+Synt. Linear.	96.22	92.90	90.36
the joint method	97.68	93.20	91.67

Table 5: Development test results. *Morph.* stands for morphology, *Gen.* stands for generation, *Synt.* stands for syntactic and *Linear.* stands for linearization.

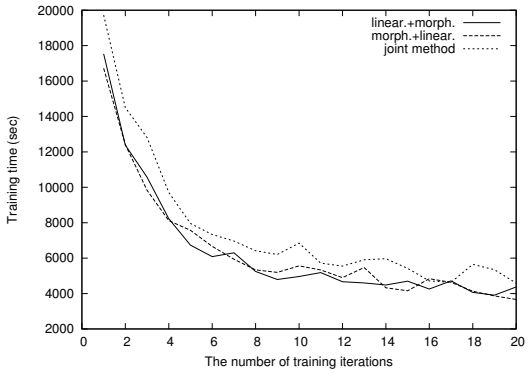

Figure 3: Training time (in seconds) by training iteration.

The results in Table 5 indicate the merits and shortcomings of each method. On the one hand, we can see that taking morphological generation as a postprocessing step benefits from word order information for morphological generation (97.27 vs 96.22). However, it suffers the lack of inflections during tree linearization (92.31 vs 92.90). On the other hand, taking morphological generation as a preprocessing step benefits from inflection information during linearization, while the lack of word order knowledge hampers morphological generation. Finally, the joint system outperforms both pipelined methods on both ordering and inflection. It proves that the joint method can improve the performance of both tasks by reducing error propagation.

Figure 3 shows the time taken for each training iteration of each method, which we use for measuring the search space. First of all, as the number of training iterations increases, the training time decreases. Because the trained model improves by the iteration, the best-first search algorithm expands fewer incorrect hypotheses before the gold goal is found. In addition, the joint method has a larger search space and its training time at each iteration is longer than the pipelined methods. Nevertheless, its decreasing tendency is similar to the two baseline methods, and its training time at the end is not significantly longer than the two baseline methods, both of which proves that the joint method does not suffer from the search space explosion problem. At the last iterations, the training time of the joint method becomes comparable with the pipelined methods.

Methods	BLEU
STUMABA-S	89.11
DCU	85.75
Morph. Generation+Synt. Linearization	88.22
Synt. Linearization+Morph. Generation	88.48
the Joint method	**89.62**

Table 6: Final results of various methods. *Morph.* is short for Morphological and *Synt.* is short for Syntactic.

Inflection Candidates Num	1	2	>=3
Count	136003	58843	5582
Percent	67.86%	29.36%	2.78%

Table 7: Numbers of lemma instances with different range of inflection candidates.

Final Results

Table 6 presents the final results of various methods on the official test set of the Generation Challenge 2011 shared task. We refer to Belz et al. (2011) to obtain the results of *STUMABA-S* and *DCU*, which are the two best results for this shared task. From the results we can see that the two pipelined methods show nearly comparable performance, which proves that taking morphological generation as preprocessing is also a reasonable option for the meaning text theory. The joint method significantly outperforms the pipelined methods, with $p<0.01$ on statistical significance test *sign-test* (Collins, Koehn, and Kucerova 2005), and exceeds the previous best system *STUMABA-S*, becoming the best reported result in this shared task.

Analysis

Inflection Candidates Generation Table 7 shows the percentage of lemma instances with different number of inflection candidates. From the result we can see that most lemmas (about 68%) have only one inflection candidate, meaning their inflection can be easily determined by simple rules and dictionary. Out of the 5582 lemma instances who have more than 3 inflection candidates, we find that 5385 of them (more than 96%) are either '*be*', '*do*' or '*have*'. Since their inflections are usually hard to determine, this give our joint method the space to improve over the pipelined baseline methods.

Unseen Words We do not have suffix features, and the inflection of unseen words are not captured by lexicalized features. However their part-of-speech can be a useful indicator for inflection disambiguation. Out of the total 57646 lemma instances in the test set, there are 644 unseen nouns and verbs, which count for about 1 percent of the total instances. Most lemma instances can be covered by our training data.

Related Work

Existing approaches of syntactic realization can be divided into several major categories: rule-based, memory-based (or example based) (Varges and Mellish 2001) and corpus-based (Langkilde and Knight 1998; Filippova and Strube 2009; Wan et al. 2009; Bohnet et al. 2010). In recent years, corpora with rich annotations (such as the CoNLL corpus) have become available, leading to a great success in corpus-based methods. However, since most of the annotations are shallow syntax and semantics, up to now most corpus-based work focuses on semantics-based realization (White 2004; Espinosa, White, and Mehay 2008) and shallow syntax-based linearization (Filippova and Strube 2007; 2009).

Two grammars are typically used by syntactic linearization systems: CCG (Zhang and Clark 2011; Zhang, Blackwood, and Clark 2012) and the dependency grammar (He et al. 2009; Wan et al. 2009; Filippova and Strube 2009; Bohnet et al. 2010; Zhang 2013). Our work follows the dependency-based methods, because the lexicalized dependency grammar is effective for disambiguation in joint morphological generation and syntactic linearization. Another reason is that the dependency grammar is commonly used in statistical machine translation (SMT) (Quirk, Menezes, and Cherry 2005; Ding and Palmer 2005; Xiong, Liu, and Lin 2007; Shen, Xu, and Weischedel 2008; Xie, Mi, and Liu 2011), a major application of natural language generation.

In recent years, there is a growing interest in generation-based methods (Bangalore, Haffner, and Kanthak 2007; Gali and Venkatapathy 2009) for SMT. However, generating from deep structure requires the recovering of inflection of each lemma. The meaning text theory pipeline of Bohnet et al. (2010) treats morphological generation as the postprocessing step of linearization, leading to a error propagation problem. Jones et al. (2012) avoids this problem by integrating morphological generation into the generation rules, resulting a high risk of data sparseness and larger ambiguity for SMT. Our work alleviates this problem by joint morphological generation and syntactic linearization. Our method can be used by dependency-based SMT models.

It has been shown that joint morphological and syntactic parsing can improve over the syntactic analysis pipeline (Bohnet et al. 2013). Our work demonstrates that its counterpart in natural language generation is also true, reinforcing the finding that syntax and morphology are closely related.

Conclusion and Future Work

In this paper, we adopt a simple but effective method for jointly performing two tasks in natural language generation: morphological generation and syntactic linearization. Our experiments show that the joint method gives significant improvement over the pipelined baselines. In addition, jointly performing the two tasks does not result in efficiency issues due to search space explosion. As a result, it should always be preferable to pipelined methods. By achieving the best reported results on this task, we also show the effectiveness of the learning-guided search framework of Zhang and Clark (2011). We release our code at https://sourceforge.net/projects/zgen/

For future work, we will investigate joint methods on partial tree tasks (Zhang 2013) and for deep syntax realization and morphological generation.

Acknowledge

The authors were supported by Singapore Ministry of Education Tier2 Grant T2MOE201301, SRG ISTD 2012 038 from SUTD, and Science Foundation Ireland (Grant No. 07/CE/I1142). We thank Anja Belz and Michael White for kindly providing the shared task data, and the anonymous reviewers for their insightful comments.

References

Bangalore, S.; Haffner, P.; and Kanthak, S. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proceedings of ACL*, 152.

Bangalore, S.; Rambow, O.; and Whittaker, S. 2000. Evaluation metrics for generation. In *Proceedings of INLG*, 1–8.

Belz, A.; White, M.; Espinosa, D.; Kow, E.; Hogan, D.; and Stent, A. 2011. The first surface realisation shared task: Overview and evaluation results. In *ENLG*, 217–226.

Bohnet, B.; Wanner, L.; Mille, S.; and Burga, A. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Coling*, 98–106.

Bohnet, B.; Nivre, J.; Boguslavsky, I.; Farkas, R.; Ginter, F.; and Hajic, J. 2013. Joint morphological and syntactic analysis for richly inflected languages. In *TACL*.

Collins, M.; Koehn, P.; and Kucerova, I. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 1–8.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. In *JMLR (Volume 7)*, 551–585.

Ding, Y., and Palmer, M. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*, 541–548.

Espinosa, D.; White, M.; and Mehay, D. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, 183–191.

Filippova, K., and Strube, M. 2007. Generating constituent order in german clauses. In *Proceedings of ACL*, 320–327.

Filippova, K., and Strube, M. 2009. Tree linearization in english: Improving language model based approaches. In *Proceedings of HLT/ACL: Short Papers*, 225–228.

Finkel, J. R., and Manning, C. D. 2009. Joint parsing and named entity recognition. In *NAACL*, 326–334.

Gali, K., and Venkatapathy, S. 2009. Sentence realisation from bag of words with dependency constraints. In *Proceedings of NAACL, Workshop*, 19–24.

Hatori, J.; Matsuzaki, T.; Miyao, Y.; and Tsujii, J. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *ACL*, 1045–1053.

He, W.; Wang, H.; Guo, Y.; and Liu, T. 2009. Dependency based chinese sentence realization. In *ACL*.

Ide, N., and Véronis, J. 1994. Multext: Multilingual text tools and corpora. In *Proceedings of Coling*, 588–592.

Jiang, W.; Huang, L.; Liu, Q.; and Lü, Y. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL/HLT*, 897–904.

Jones, B.; Andreas, J.; Bauer, D.; Hermann, K. M.; and Knight, K. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Coling*, 1359–1376.

Langkilde, I., and Knight, K. 1998. Generation that exploits corpus-based statistical knowledge. In *Coling*, 704–710.

Oh, A. H., and Rudnicky, A. I. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of ANLP/NAACL Workshop*, 27–32.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 311–318.

Quirk, C.; Menezes, A.; and Cherry, C. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of ACL*, 271–279.

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

Sun, W. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *ACL*.

Sutton, C.; McCallum, A.; and Rohanimanesh, K. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *JMLR (Volume 8)*, 693–723.

Varges, S., and Mellish, C. 2001. Instance-based natural language generation. In *Proceedings of NAACL*, 1–8.

Wan, S.; Dras, M.; Dale, R.; and Paris, C. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of EACL*, 852–860.

White, M. 2004. Reining in CCG chart realization. In *Proceedings of INLG-04*, 182–191.

Xie, J.; Mi, H.; and Liu, Q. 2011. A novel dependency-to-string model for statistical machine translation. In *EMNLP*.

Xiong, D.; Liu, Q.; and Lin, S. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of the WMT*, 40–47.

Zhang, Y., and Clark, S. 2008. Joint word segmentation and POS tagging using a single perceptron. In *ACL*, 888–896.

Zhang, Y., and Clark, S. 2011. Syntax-based grammaticality improvement using CCG and guided search. In *EMNLP*.

Zhang, M.; Zhang, Y.; Che, W.; and Liu, T. 2013. Chinese parsing exploiting characters. In *ACL*, 125–134.

Zhang, Y.; Blackwood, G.; and Clark, S. 2012. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of EACL*, 736–746.

Zhang, Y. 2013. Partial-tree linearization: generalized word ordering for text synthesis. In *IJCAI*, 2232–2238.