Sentence Realisation from Bag of Words with dependency constraints

Karthik Gali, Sriram Venkatapathy
Language Technologies Research Centre,
IIIT-Hyderabad, Hyderabad, India
{karthikg@students,sriram@research}.iiit.ac.in

Abstract

In this paper, we present five models for sentence realisation from a bag-of-words containing minimal syntactic information. It has a large variety of applications ranging from Machine Translation to Dialogue systems. Our models employ simple and efficient techniques based on n-gram Language modeling.

We evaluated the models by comparing the synthesized sentences with reference sentences using the standard BLEU metric(Papineni et al., 2001). We obtained higher results (BLEU score of 0.8156) when compared to the state-of-art results. In future, we plan to incorporate our sentence realiser in Machine Translation and observe its effect on the translation accuracies.

1 Introduction

In applications such as Machine Translation (MT) and Dialogue Systems, sentence realisation is a major step. Sentence realisation involves generating a well-formed sentence from a bag of lexical items. These lexical items may be syntactically related to each other. The level of syntactic information attached to the lexical items might vary with application. In order to appeal to the wide range of applications that use sentence realisation, our experiments assume only basic syntactic information, such as unlabeled dependency relationships between the lexical items.

In this paper, we present different models for sentence realisation. These models consider a bag of words with unlabelled dependency relations as input and apply simple n-gram language modeling techniques to get a well-formed sentence.

We now present the role of a sentence realiser in the task of MT. In transfer-based approaches for MT[1] (Lavie et al., 2003), the source sentence is first analyzed by a parser (a phrase-structure or a dependency-based parser). Then the source lexical items are transferred to the target language using a bi-lingual dictionary. The target language sentence is finally realised by applying transfer-rules that map the grammar of both the languages. Generally, these transfer rules make use of rich analysis on the source side such as dependency labels etc. The accuracy of having such rich analysis (dependency labeling) is low and hence, might affect the performance of the sentence realiser. Also, the approach of manually constructing transfer rules is costly, especially for divergent language pairs such as English and Hindi or English and Japanese. Our models can be used in this scenario, providing a robust alternative to the transfer rules.

A sentence realiser can also be used in the framework of a two-step statistical machine translation. In the two-step framework, the semantic transfer and sentence realisation are decoupled into independent modules. This provides an opporunity to develop simple and efficient modules for each of the steps. The model for *Global Lexical Selection and Sentence Re-construction* (Bangalore et al., 2007) is one such approach. In this approach, discriminative techniques are used to first transfer semantic information of the source sentence by looking at the source sentence globally, this obtaining a accurate bag-of-words in the target language. The words in the bag might be attached with mild syntactic information (ie., the words they modify) (Venkatapathy and Bangalore, 2007). We propose models that take

[1] http://www.isi.edu/natural-language/mteval/html/412.html

Proceedings of the NAACL HLT Student Research Workshop and Doctoral Consortium, pages 19–24,
Boulder, Colorado, June 2009.

this information as input and produce the target sentence. We can also use our sentence realiser as an ordering module in other approaches such as (Quirk et al., 2005), where the goal is to order an unordered bag (of treelets in this case) with dependency links.

In Natural Language Generation applications such as Dialogue systems etc, the set of concepts and the dependencies between the concepts is obtained first which is known as text planning. These concepts are then realized into words resulting in a bag of words with syntactic relations (Bangalore and Rambow, 2000). This is known as sentence planning. In the end, the surface string can be obtained by our models.

In this paper, we do not test our models with any of the applications mentioned above. However, we plan to test our models with these applications, especially on the two-stage statistical MT approach using the bag-of-words obtained by Global Lexical Selection (Bangalore et al., 2007),(Venkatapathy and Bangalore, 2007). Here, we test our models independent of any application, by beginning with a given bag-of-words (with dependency links).

The structure of the paper is as follows. We give an overview of the related work in section 2. In section 3, we talk about the effect of dependency constraints and gives details of the experimental setup in section 4. In section 5, we describe about the experiments that have been conducted. In section 6, our experimental results are presented. In section 7, we talk about the possible future work and we conclude with section 8.

2 Related Work

There have been approaches for sentence realisation such as FUF/SURGE (Elhadad, 1991), OpenCCG (White, 2004) and XLE (Crouch et al., 2007) that apply hand-crafted grammars based on particular linguistic theories. These approaches expect rich syntactic information as input in order to realise the sentence. There are other approaches in which the generation grammars are extracted semi-automatically (Belz, 2007) or automatically (such as HPSG (Nakanishi and Miyao, 2005), LFG (Cahill and van Genabith, 2006; Hogan et al., 2007) and CCG (White et al., 2007)). The limitation of these approaches is that these cannot be incorporated into a wide range of applications as they rely on rich syntactic information for generation. On the contrary, we use simple n-gram models to realise (or linearize) a bag-of-words where the only information available is the presence of various links between the words.

Our work is similar to a recently published work by Guo (Guo et al., 2008). They use n-gram models to realise sentences from the f-structures of HPSG (equivalent to labeled dependency structure). Their models rely heavily on the dependency relation labels (also called grammatical roles) available in HPSG. However, the dependency role information (of any dependency formalism) is either not readily available in a variety of applications in NLP. We propose to explore the realisation of a sentence using minimal syntactic information. Apart from dependency links, we also make use of part-of-speech tags which are easily available and hence, our sentence realiser can be plugged much easily into various applications. Guo (Guo et al., 2008) conduct their experiments by considering gold data as input. Apart from using gold data as input, we also conduct experiments by assuming noisy input data to test the robustness of our models. The search algorithm used by both Guo and us is locally greedy i.e., we compute the best string at every node. Guo uses the Viterbi algorithm to get best string whereas we consider and score all permutations to obtain the best string.

There has been burgeoning interest in the probabilistic models for sentence realisation, especially for realisation ranking in a two stage sentence realisation architecture where in the first stage a set of sentence realisations are generated and then a realisation ranker will choose the best of them (Bangalore and Rambow, 2000).

One major observation in our experiments was that the POS tags held immensely in the task of sentence realisation.

3 Effect of Dependency Constraints

There is a major advantage in using dependency constraints for sentence realisation. The search space reduces drastically when the constraints are applied. These constraints state that the realised sentences should be projective with respect to the de-

pendency structure (unordered) of the input bag-of-words ie.., any word and its children in the dependency tree should project as a contiguous unit in the realised sentence. This is a safe assumption to make as the non-projectivity in English is only used to account for Long-Distance Dependencies and such cases are low in number (Guo et al., 2008).

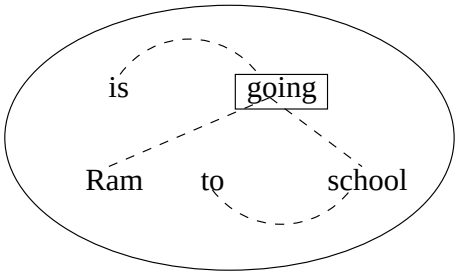

Figure 1: Bag of words with dependency constraints and head marked

We now present an example to show how the dependency constraints reduce the search space. For example, consider an unordered dependency tree in Figure 1, which has five words. If we don't use the constraints provided by the dependency tree then the search space is 5! (120). But, if we use the constraints provided by the dependency tree then the search space is 2! + 4! = 28. There is a huge reduction in the search space if we use the constraints provided by the dependency tree. Further, it has been shown in (Chang and Toutanova, 2007) that applying the constraints also aids for the synthesis of better constructed sentences.

4 Experimental Set-up

For the experiments, we use the WSJ portion of the Penn tree bank (Marcus et al., 1993), using the standard train/development/test splits, viz 39,832 sentences from 2-21 sections, 2416 sentences from section 23 for testing and 1,700 sentences from section 22 for development. The input to our sentence realiser are bag of words with dependency constraints which are automatically extracted from the Penn treebank using head percolation rules used in (Magerman, 1995), which do not contain any order information. We also use the provided part-of-speech tags in some experiments.

In a typical application, the input to the sentence realiser is noisy. To test the robustness of our models in such scenarios, we also conduct experiments with noisy input data. We parse the test data with an unlabelled projective dependency parser (Nivre et al., 2006) and drop the order information to obtain the input to our sentence realiser. However we still use the correct bag of words. We propose to test this aspect in future by plugging our sentence realiser in Machine Translation.

Table 1 shows the number of nodes having a particular number of children in the test data.

Children	countNodes		Children	countNodes
0	30219		5	1017
1	13649		6	685
2	5887		7	269
3	3207		8	106
4	1526		> 8	119

Table 1: The number of nodes having a particular number of children in the test data

From Table 1, we can see that more than 96% of the internal nodes of the trees contain five or less children. It means that for almost all the nodes, the reordering complexity is minimal. This makes this approach very feasible if the order of a sub-tree is computed after the order of the sub-trees of its children is fixed. Hence, the approaches that we present in the next section use bottom-up traversal of the tree. During the traversal, the appropriate order of every sub-tree is fixed.

5 Experiments

The task here is to realise a well formed sentence from a bag of words with dependency constraints (unordered dependency tree) for which we propose five models using n-gram based Language modeling techinque. We train the language models of order 3 using Good-Turning smoothing on the training data of Penn Treebank.

5.1 Model 1 : Sentential Language Model

We traverse the tree in bottom up manner and find the best phrase at each subtree. The best phrase corresponding to the subtree is assigned to the root node of the sub-tree during the traversal.

Let the node n have N children represented as c_i ($1 < i < N$). During the bottom up traversal, the

children c_i are assigned best phrases before processing node n. Let the best phrases corresponding to the children be $p(c_i)$. The best phrase corresponding to the node n is computed by exploring the permutations of n and the best phrases $p(c_i)$ corresponding to the children c_i. The total number of permutations that are explored are $(N+1)!$. A sentential language model is applied on each of the candidate phrases to select the best phrase.

$$p(n) = bestPhrase\ (\ perm\ (n,\ \forall\ i\ p(c_i))\ o\ LM\ ) \quad (1)$$

In Sentential Language Model, we used a LM that is trained on complete sentences of the training corpus to score the permutations.

5.2 Model 2 : Subtree-type based Language Models(STLM)

The major problem with model 1 is that we are using a common sentential language model (trained on complete sentences) to score phrases corresponding to various sub-tree types. In this model, we build different LMs for phrases corresponding to different subtree-types.

To build STLMs, the training data is parsed first. Each subtree in the parse structure is represented by the part-of-speech tag of its head. Different language models are created for each of the POS tags. We have 44 different language models each corresponding to a particular POS tag. For example, a *IN* language model contains phrases like *in hour, of chaos, after crash, in futures, etc* and VBD language model contains phrases like *were criticized, never resumed* while training.

So, in this model we realise a sentence from a unordered dependency tree by traversing the dependency tree in bottom-up manner as we did in model 1; but while scoring the permuted phrases we use different language models for subtrees headed by words of various pos tags.

$$p(n) = bestPhrase\ (\ perm\ (n,\ \forall\ i\ p(c_i))\ o\ LM_{POS(n)}\ ) \quad (2)$$

Here, $LM_{POS(n)}$ represents the language model associated with the part-of-speech of the node n.

5.3 Model 3 : Head-word STLM

In the models presented earlier, a node and its children are ordered using the best phrases of the children. For example, the best phrase assigned to the node 'was' is computed by taking of the permutation of 'was' and its children 'The equity market', 'illiquid' and '.' and then applying the language model. In model 3, instead of considering best phrases while ordering, the heads of the the children c_i are considered. For example, the best phrase assigned to the node 'was' is computed by first permuting the nodes 'was', 'market', 'illiquid' and '.' and then applying the language models trained on the treelets (head and children) and not on entire sub-trees.

The major advantage of using this model is that order at a node is independent of the best phrases of its descendants and also any mistakes in computation of best phrases of descendants doesn't effect the choice of reordering decision at a particular node.

5.4 Model 4 : POS based STLM

We now experiment by using Part-Of-Speech (POS) tags of words for ordering the nodes. In the previous approaches, the language models were trained on the words which were then used to compute the best strings associated with various nodes. Here, we order the node and its children using a language model trained on POS tag sequences. The motivation behind buliding such kind of Language models is that it deals with unseen words effectively. Hence, in this model, the best phrase corresponding to the node 'was' is obtained by permuting the POS tags of the words 'was', 'market', 'illiquid and '.' which are 'VBZ', 'NN', 'NN' and '.' respectively. As the best POS tag sequence might correspond to several orderings of the treelet, a word based STLM is applied to choose the correct ordering.

The major advantages of this model is that it is more general and it deals with unseen words effectively. Also, it is much faster than earlier models as this model is a POS tag based model.

5.5 Model 5: Head-marked POS based STLM

In POS based STLM, the head of a particular node isn't marked while applying the language model. Hence, all the nodes of the treelet are treated equally while applying the LM. For example, in Figure 2, the structures of treelets is not taken into account while applying the head-POS based language model. Both are treated in the same manner while applying TLM. In this model, we experiment by marking the head

information for the POS of the head word which treats the treelets in Figure 2 in a different manner to obtain the best phrase. As the best POS tag sequence might correspond to several orderings of the treelet, we test various word-based approaches to choose the best ordering among the many possibilities. The best approach was the one where head-word of the treelet had the POS tag attached to it.

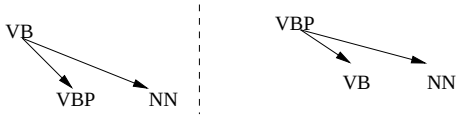

Figure 2: Two different treelets which would have same best POS tag sequence

6 Results and Discussion

To evaluate our models, we compare the system generated sentences with reference sentences and get the BLEU score. As mentioned in section 4, We evaluate our models on two different types of input. In the first input type, we have bag of words with dependency constraints extracted from treebank and in the second input type, the dependency constraints among the bag of words are extracted from the parser which are noisy. Table 2 shows the results of model 1-5.

Model	Treebank(gold)	Parser(noisy)
Model 1	0.5472	0.5514
Model 2	0.6886	0.6870
Model 3	0.7284	0.7227
Model 4	0.7890	0.7783
Model 5	0.8156	0.8027

Table 2: The results of Model 1-5

We can observe that in model 1, BLEU score of the parser input is high when compared to Treebank input. This might be because, the parser input is projective (as we used projective parsing) whereas the treebank input might contain some non-projective cases. In general, for all the models, the results with noisy dependency links are comparable to the cases where gold dependency links are used which is encouraging.

We have taken the Table-3 from (Guo et al., 2008), which shows the BLEU scores of different

Paper	BLEU score
Langkilde(2002)	0.757
Nakanishi(2005)	0.705
Cahill(2006)	0.6651
Hogan(2007)	0.6882
White(2007)	0.5768
Guo(2008)	0.7440
Our Model	0.8156

Table 3: Comparsion of results for English WSJ section 23

systems on section 23 of PTB. Its really difficult to compare sentence realisers as the information contained in the input vaires greatly between systems. But, we can clearly see that the our system performs better than all the systems. The main observations from the results are, (1) Searching the entire space of O(n!) helps, (2) Treelet LM capture characteristics of phrases headed by various POS tags, in contrast to sentential LM which is a general LM, (3) POS tags can play an important role in ordering nodes of a dependency structure, (4) The head models performed better than the models that used all the nodes of the sub-tree, and (5) Marking the head of a treelet provides vital clues to the language model for reordering.

7 Future Experiments

Although the results of the proposed models are much higher when compared to other methods, the major constraint with our models is the computational complexity, which is O(n!). However, our approach is still tractable because of the low values of n. We plan to reduce the search space complexity by using Viterbi search (Guo et al., 2008), and examine the drop in results because of that.

The models proposed in paper, consider only the locally best phrases (local to the sub-tree) at every step. In order to retain the globally best possibilities at every step, we plan to use beam search, where we retain K-best best phrases for every sub-tree.

Also, the goal is to test the approach for morphologically-rich languages such as Hindi. Also, it would require us to expand our features set. We also plan to test the factored models.

The most important experiment that we plan to

perform is to test our system in the context of MT, where the input is more real and noisy.

To train more robust language models, we plan to use the much larger data on a web scale.

8 Conclusion

In this paper, we had experimented with five ngram based models for sentence realisation from bag of words with dependency constraints. We have evaluated our models on two different types of input(gold and noisy). From the results, we can conclude that the model 'Marked Head-POS based LM' works best in both cases.

Acknowledgments

The authors of this work were supported by ILMT grant 11(10)/2006-HCC(TDIL) and EILMT grant 11(9)/2006HCC(TDIL). We would also like to thank the four reviewers for their valuable reviews.

References

S. Bangalore and O. Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. *Proceedings of the 18th conference on Computational linguistics*.

S. Bangalore, P. Haffner, and S. Kanthak. 2007. Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction. In *Annual Meeting - ACL*, volume 45.

A. Belz. 2007. Probabilistic Generation of Weather Forecast Texts. In *Proceedings of NAACL HLT*.

A. Cahill and J. van Genabith. 2006. Robust PCFG-Based Generation Using Automatically Acquired LFG Approximations. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 44.

P.C. Chang and K. Toutanova. 2007. A Discriminative Syntactic Word Order Model for Machine Translation. *Proceedings of the 45th Annual Meeting of the ACL*.

D. Crouch, M. Dalrymple, R. Kaplan, T. King, J. Maxwell, and P. Newman. 2007. XLE documentation. *Available on-line*.

M. Elhadad. 1991. FUF: The universal unifier user manual version 5.0. *Department of Computer Science, Columbia University. New York*.

Y. Guo, J. van Genabith, and H. Wang. 2008. Dependency-Based N-Gram Models for General Purpose Sentence Realisation. *Proceedings of the 22nd conference on Computational linguistics*.

D. Hogan, C. Cafferkey, A. Cahill, and J. van Genabith. 2007. Exploiting Multi-Word Units in History-Based Probabilistic Generation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

A. Lavie, S. Vogel, L. Levin, E. Peterson, K. Probst, A.F. Llitjós, R. Reynolds, J. Carbonell, and R. Cohen. 2003. Experiments with a Hindi-to-English transfer-based MT system under a miserly data scenario. *ACM-TALIP*, 2(2).

D.M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on ACL*. ACL Morristown, NJ, USA.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*, 19(2).

H. Nakanishi and Y. Miyao. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the International Workshop on Parsing Technology*.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth CoNLL*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on ACL*.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. *Proceedings of the 43rd Annual Meeting of ACL*.

S. Venkatapathy and S. Bangalore. 2007. Three models for discriminative machine translation using Global Lexical Selection and Sentence Reconstruction. In *Proceedings of SSST, NAACLHLT/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 152–159.

M. White, R. Rajkumar, and S. Martin. 2007. Towards Broad Coverage Surface Realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)*.

M. White. 2004. Reining in CCG Chart Realization. *LECTURE NOTES IN COMPUTER SCIENCE*.