From Factorial to Quadratic Time Complexity for Sentence Realization using Nearest Neighbour Algorithm

Karthik Gali, Sriram Venkatapathy, Taraka Rama

[1] Language Technologies Research Centre,
IIIT-Hyderabad, Hyderabad, India
{karthikg@students,sriram@research,taraka@students}.iiit.ac.in

Abstract. *Sentence Realization is the task of generating a well-formed sentence from a bag of words. Sentence Realization is a major step in many Natural Language Processing applications like Machine Translation (MT), Summarization and Dialogue Systems. In this paper, we explore a graph based Nearest Neighbour Algorithm for the task of Sentence Realization.*

1. Introduction

Sentence Realization is a major step in many Natural Language Processing applications like Machine Translation (MT), Summarization and Dialogue Systems. The task of Sentence Realization involves formation of a well-formed sentence from a bag of lexical items. These lexical items may be attached syntactically with one another. The level of syntactic information varies from application to application. Our aim consists of achieving quality sentence realiser using as much as minimum syntactic information and of minimal computational complexity. As such our experiments assume only basic syntactic information, such as unlabeled dependency relationships between the lexical items.

Graph based algorithms for Natural Language applications such as Parsing [McDonald et al. 2005], Summarization [Mihalcea and Tarau 2005] and Word sense disambiguation [Mihalcea 2005] have been well explored. For the task of Sentence Realization, graph based algorithms have yet to be explored. This paper is a novel effort in that direction.

In this paper, we explore the graph based Nearest Neighbour Algorithm[1] for the task of Sentence Realization from a bag of words with dependency constraints. In a similar work, Gali and Venkatapathy(2009) have suggested syntax based language models for the task of sentence realization from bag of words. For our case, we adopt their experimental setup and try to address the limitations of their approach using our approach.

The paper is organised as follows. Section 2 discusses the models given in [Gali and Venkatapathy 2009]. The details of the experimental setup is given in Section 3 and the algorithm itself is described in detail in Section 4. The outcome of our experiments are presented and discussed in Section 5. We talk about the possible future directions of the work in Section 6 and conclude in Section 7.

2. Syntax based Language Models

Gali and Venkatapathy(2009) has suggested syntax based language models for the task of sentence realization. In their approach, they travel the unordered dependency tree(bag

[1]http://en.wikipedia.org/wiki/Nearest_neighbour_algorithm

of words with dependency constraints) in a bottom up fashion. They compute the best relative order at every node in the unordered dependency tree using five different types of language models like *sentential language model, subtree-type based language models(STLM), head-word STLM, POS based STLM and head-marked POS based STLM*. The best order obtained at the head of the dependency tree is the sentence realized from the unordered dependency tree.

Head-marked POS based STLM model performs the best among all. One of the problems, with this model is that it does a exhaustive search which is in the order of $N!$. In this paper, we present a graph based model of quadratic complexity for finding the best relative order at every node instead of doing an exhaustive search, as in the case of head-marked POS based STLM, albeit at the cost of accuracy.

3. Experimental Setup

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. The same experimental setup described in [Gali and Venkatapathy 2009] is kept in place for comparing our model with theirs.

4. Nearest Neighbour Algorithm

A major bottleneck of the models presented in [Gali and Venkatapathy 2009] is that their worst case complexity is $O(N!)$, where N is the number of nodes in the subtree. In order to overcome this bottleneck we propose a faster algorithm based on Nearest Neighbour algorithm. The problem is formulated as follows. We represent the nodes and ”<s>” which represents the beginning of a phrase as vertices of a graph. Then a directed complete graph is constructed from the above vertices.

A complete directed graph is a simple graph in which every pair of distinct vertices is connected by an directed edge. The complete directed graph on n vertices has $n*(n-1)$ edges. Every edge in complete directed graph is associated with a score $Score(x, y)$ that maps edge between x and y to a real number. These scores are a negative of the conditional probability $p(y/x)$ which is given by $-(count(x, y)/count(x))$. We take the negative of the conditional probability in order to apply the algorithm directly. We can see from the above formulation that the path which visits each vertex exactly once and has the least score is the required phrase of the subtree. Such a path which visits each vertex exactly once is called Hamiltonian path. Now, our task of finding the best sentence is reduced to building a complete directed graph with nodes of subtree as vertices and find the least costly Hamiltonian path.

But finding the Hamiltonian path is a NP-complete problem i.e., the search space is $N!$ for N nodes. But there are some approximate algorithms[2] which give the approxi-

[2]http://en.wikipedia.org/wiki/Approximation_algorithm

mate solution with less computational complexity. The solution might not be always optimal but the primary advantage is the reduction of search space from factorial to quadratic. One of the familiar approximate algorithm is Nearest Neighbour algorithm.

In Nearest Neighbour Algorithm, we start with <s> node and mark it has visited. Then we find and visit the lightest edge going from the current vertex and not visited and mark it as visited. We continue till all the nodes of the graph are visited which gives us the Hamiltonian path.

Since there are (N^2) possible bigram probabilities, the run time complexity of the Nearest Neighbour algorithm is in the order of N^2. This will decrease the search space and in turn effect the system output since the best output might not be possibly explored. The method described takes the local best at each step. So, the output might not be a global best. In order to get the global best, we store K-best instead of top one at each stage. Then in the end when we get the K-best phrases for the subtree. Then we chose the best phrase having highest global probability. Higher value of K allows for more phrases to be considered. Hence the search space for the K-best Nearest Neighbour algorithm will be in the order of $K * N^2$.

5. Results

BLEU score [Papineni et al. 2001] used in evaluation of the performance of a MT system is used to evaluate our system's performance. We observe from figure 1 that BLEU score increases at a faster rate upto the value of $K = 10$ and stabilises after that.

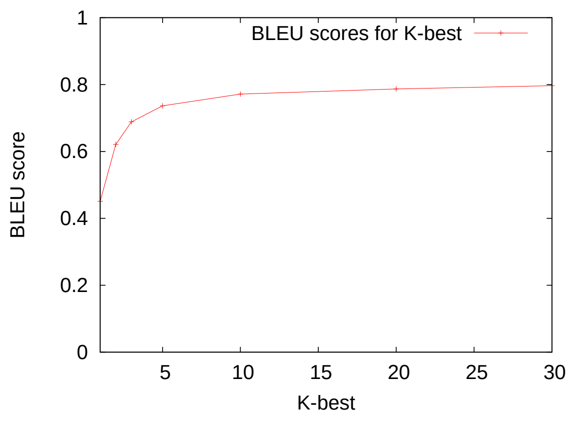

Figure 1. Graph showing the BLEU scores of Nearest Neighbour Algorithm for different values of K

We can see from the figure that for $K = 30$ the BLEU score achieved for the standard test set is 0.7968. Gali and Venkatapathy(2009) who does a exhaustive search of $N!$ has achieved a BLEU score of 0.8156 on the same test data. The results show that there is a decrease of 0.0188 BLEU score with the decrease of computational complexity from $N!$ to $K * N^2$ (K=30).

6. Future Experiments

In this paper, we have explored the Nearest Nearest Algorithm for finding the Hamiltonian path in a complete graph. In future, we would like to explore different approximation algorithms for finding the Hamiltonian path in a complete graph. We would also like to test our approach for morphologically-rich languages such as Hindi. Another possible direction for the future work lies in verifying how the BLEU score is affected for larger values of K. We also like to evaluate our system with various other sentence evaluation measures such as METEOR, WER, PER as a part of our future work.

7. Conclusion

In this paper, we have successfully tested the graph based Nearest Neighbour Algorithm for the task of Sentence Realization. We have shown that using the graph-based algorithm can reduce the computational complexity from *factorial* to *quadratic* at the cost of 2% reduction in the overall BLEU score. This method of decreasing the computational complexity at a very low cost makes our module suitable for employment in practical applications. We have also checked the importance of storing K-best solutions at each stage and chose the best sentence with higher global probability at the end. The BLEU score improved from 0.4513 with $K = 1$ to 0.7968 with $K = 30$.

Acknowledgements

We would like to thank all the three anonymous authors for their valuable suggestions which helped in improving the paper significantly.

References

Gali, K. and Venkatapathy, S. (2009). Sentence Realisation from Bag of Words with dependency constraints. In *Proceedings of the HLT NAACL 2009 Student Research Workshop*.

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

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

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT and EMNLP*, pages 523–530.

Mihalcea, R. (2005). Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the conference on HLT and EMNLP*, pages 411–418. ACL Morristown, NJ, USA.

Mihalcea, R. and Tarau, P. (2005). Multi-document Summarization with iterative graph-based algorithms. In *Proceedings of the First International Conference on Intelligent Analysis Methods and Tools (IA 2005), McLean, VA.*

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2001). BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting on ACL.*