A Graph-Based Approach to String Regeneration

Matic Horvat
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue, CB3 0FD, U.K.
mh693@cam.ac.uk

William Byrne
Department of Engineering
University of Cambridge
Trumpington Street, CB2 1PZ, U.K.
wjb31@cam.ac.uk

Abstract

The string regeneration problem is the problem of generating a fluent sentence from a bag of words. We explore the N-gram language model approach to string regeneration. The approach computes the highest probability permutation of the input bag of words under an N-gram language model. We describe a graph-based approach for finding the optimal permutation. The evaluation of the approach on a number of datasets yielded promising results, which were confirmed by conducting a manual evaluation study.

1 Introduction

The string regeneration problem can be stated as: given a bag of words taken from a fluent grammatical sentence, recover the original sentence. As it is often difficult to recover the exact original sentence based solely on a bag of words, the problem is relaxed to generating a fluent version of the original sentence (Zhang and Clark, 2011).

The string regeneration problem can generally be considered a difficult problem even for humans. Consider the following bag of words:

> { *Iraq, list, in, a, third, joins, the, ., of, Bush's, of, critics, policy, senator, republican* }

and try to recover the original sentence or at least a fluent grammatical sentence. The original sentence was:

> *a third republican senator joins the list of critics of Bush's policy in Iraq.*

The purpose of investigating and developing approaches to solving the string regeneration problem is grammaticality and fluency improvement of machine generated text. The output of systems generating text, including SMT, abstract-like text summarisation, question answering, and dialogue systems, often lacks grammaticality and fluency (Knight, 2007; Soricut and Marcu, 2005). The string regeneration problem is used as an application-independent method of evaluating approaches for improving grammaticality and fluency of such systems.

The string regeneration can also be viewed as a natural language realization problem. The basic task of all realization approaches is to take a meaning representation as input and generate human-readable output. The approaches differ on how much information is required from the meaning representation, ranging from semantically annotated dependency graphs to shallow syntactic dependency trees. A simple bag of words can then be considered as the least constrained input provided to a natural language realization system. The bag of words can be combined with partial constraints to form a more realistic meaning representation.

Wan et al. (2009) proposed an algorithm for grammaticality improvement based on dependency spanning trees and evaluated it on the string regeneration task. They compared its performance against a baseline N-gram language model generator. They found that their approach performs better with regards to BLEU score. The latter approach does well at a local level but nonetheless often produces ungrammatical sentences.

We argue that the authors have not fully explored the N-gram language model approach to string regeneration. They used a Viterbi-like generator with a 4-gram language model and beam pruning to find approximate solutions. Additionally, the 4-gram language model was trained on a relatively small dataset of around 20 million words.

The N-gram language model approach finds the highest probability permutation of the input bag

Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 85–95,
Gothenburg, Sweden, April 26-30 2014. ©2014 Association for Computational Linguistics

of words under an N-gram language model as the solution to the string regeneration problem. In this paper we describe a graph-based approach to computing the highest probability permutation of a bag of words. The graph-based approach models the problem as a set of vertices containing words and a set of edges between the vertices, whose cost equals language model probabilities. Finding the permutation with the highest probability in the graph formulation is equal to finding the shortest tour in the graph or, equally, solving the Travelling Salesman Problem (TSP). Despite the TSP being an NP-hard problem, state-of-the-art approaches exist to solving large problem instances. An introduction to TSP and its variants discussed in this paper can be found in Applegate et al. (2006b).

In contrast to the baseline N-gram approach by Wan et al. (2009), our approach finds optimal solutions. We built several models based on 2-gram, 3-gram, and 4-gram language models. We experimentally evaluated the graph-based approach on several datasets. The BLEU scores and example output indicated that our approach is successful in constructing a fairly fluent version of the original sentence. We confirmed the results of automatic evaluation by conducting a manual evaluation. The human judges were asked to compare the outputs of two systems and decide which is more fluent. The results are statistically significant and confirm the ranking of the systems obtained using the BLEU scores. Additionally, we explored computing approximate solutions with time constraints. We found that approximate solutions significantly decrease the quality of the output compared to optimal ones.

This paper describes work conducted in the MPhil thesis by Horvat (2013).

2 Graph-Based Approach to String Regeneration

The underlying idea of the approach discussed in this paper is to use an N-gram language model to compute the probabilities of permutations of a bag of words and pick the permutation with the highest probability as our solution.

The probability of a sequence of words under an N-gram language model is computed as:

$$\log P(w_1^n) = \sum_{k=1}^{n} \log P(w_k | w_{k-N+1}^{k-1}) \quad (1)$$

2.1 Naive Approach

A naive approach to finding the permutation with the highest probability is to enumerate all permutations, compute their probabilities using Equation 1, and choose the permutation with the highest probability as the solution.

The time complexity of the naive approach is $\mathcal{O}(n \cdot n!)$ as we are enumerating all permutations of n words and multiplying n conditional probabilities for each permutation. This means that the naive approach is not viable for sentences of even moderate length. For example, there are 3,628,800 permutations of 10 words and 355,687,428,096,000 of 17 words.

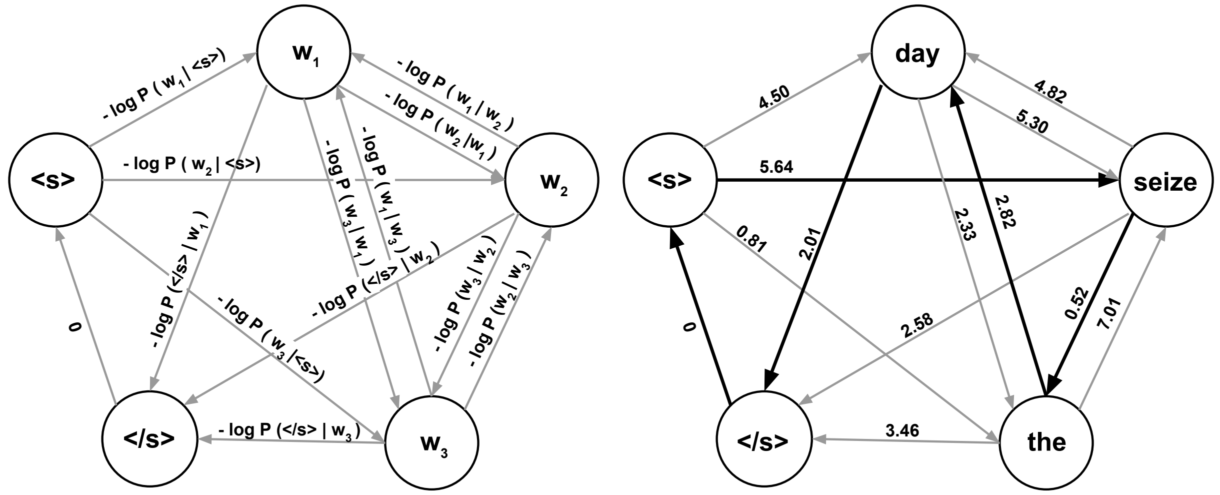
2.2 Bigram Graph-Based Approach

In this section we define the graph-based approach to finding the highest probability permutation and consequently our solution to the string regeneration problem. For a bag of words S we define a set of symbols X, $X = S \cup \{\texttt{<s>}, \texttt{</s>}\}$, which contains all the words in S (with indexes appended to distinguish repeated words) and the start and end of sentence symbols. For a bigram language model, $N = 2$, we define a directed weighted graph $G = (V, E)$, with the set of vertices defined as $V = \{w_i | w_i \in X\}$. Therefore, each symbol in X is represented by a single vertex. Let the set of edges E be a set of ordered pairs of vertices (w_i, w_j), such that $E = \{(w_i, w_j) | w_i, w_j \in V\}$. The edge cost is then defined as:

$$c_{ij} = \begin{cases} 0 & \text{if } w_i = \texttt{</s>} \\ & \text{and } w_j = \texttt{<s>}, \\ -\log P(w_j | w_i) & \text{if } w_i \neq w_j, \\ \infty & \text{otherwise.} \end{cases} \quad (2)$$

The conditional log probabilities of the form $\log P(w_j | w_i)$ are computed by a bigram language model. Consequently, finding the sentence permutation with the highest probability under the bigram language model equals finding the shortest tour in graph G or, equally, solving the Asymmetric Travelling Salesman Problem (ATSP). A general example graph for a sentence of length 3 is shown in Figure 1a.

The individual cases of the edge cost function presented in Equation 2 ensure that the solution tour is a valid sentence permutation. The negation of log probabilities transforms the problem of



(a) A general graph. The edge cost equals the negated bigram conditional log probability of the destination vertex given the origin vertex. Only edges with non-infinite edge cost are shown in the graph. Finding the shortest tour in the graph equals finding the sentence permutation with the highest probability.

(b) An example graph for the bag of words { *day, seize, the* }. The shortest tour is shown in bold and represents the word sequence *<s> seize the day </s>* with the log probability of -10.98. It is necessary to include the (</s> <s>) edge in order to complete the tour.

Figure 1: Graphs modelling a general (Figure 1a) and an example (Figure 1b) bag of words of size three under a bigram language model.

finding the longest tour in graph G to the common problem of finding the shortest tour.

Figure 1b shows a graph for the example bag of words { *day, seize, the* }. The shortest tour is shown in bold and represents the word sequence *<s> seize the day </s>*. The shortest tour equals the sentence permutation with the highest probability under the bigram language model.

The number of vertices and edges in the graph grows with the size n of the bag of words S represented by the graph $G = (V, E)$. The size of the set of vertices V in the graph is $|V| = n + 2$ and the size of the set of edges E is $|E| = |V|^2 = n^2 + 4n + 4$.

We can draw several conclusions about the graph-based approach from its equality to the TSP. Firstly, we can observe that the problem of finding the highest probability permutation is an NP-hard problem. Secondly, modelling the problem as a TSP still presents a large improvement on the naive approach described in Section 2.1. The time complexity of the naive approach for a bag of words of size n equals $\mathcal{O}(n \cdot n!)$. However, the algorithm for solving the TSP with the best-known running time guarantee has the time complexity of $\mathcal{O}(n^2 2^n)$ (Held and Karp, 1962; Applegate et al., 2006b). Although the required time grows exponentially with the length of the sentence, it grows significantly slower than with the factorial time complexity. This is illustrated in Table 1.

n	5	10	15
$n^2 2^n$	800	102,400	7,372,800
$n \cdot n!$	600	36,288,000	19,615,115,520,000

Table 1: Illustration of problem size growth at increasing values of n for algorithms with time complexity of $\mathcal{O}(n^2 2^n)$ and $\mathcal{O}(n \cdot n!)$.

Finally, by modelling the problem as a TSP we are able to take advantage of the extensive research into the TSP and choose between hundreds of algorithms for solving it. Even though no algorithm with lower running time guarantee than $\mathcal{O}(n^2 2^n)$ has been discovered since the dynamic programming algorithm described by Held and Karp (1962), many algorithms that have no guarantees but perform significantly better with most graph instances have been developed since. The size of the largest optimally solved instance has increased considerably over the years, reaching 85,900 vertices in 2006 (Applegate et al., 2009). For a more complete overview of the history and current state-of-the-art computational approaches to solving the TSP we refer the reader to Applegate et al. (2006b).

2.3 Higher N-gram Order Graph-Based Approach

Higher order N-gram language models use longer context compared to bigram language models when computing the conditional probability of the next word. This usually results in improved probability estimates for sequences of words. Therefore, to improve our initial approach using bigrams, we extend it to higher order N-grams. We first explain the intuition behind the approach and then continue with the formal definition.

The higher N-gram Order Graph-Based Approach can be modelled as a Generalized Asymmetric Travelling Salesman Problem (GATSP). GATSP is the directed (asymmetric) version of the Generalized Travelling Salesman Problem (GTSP). GTSP generalizes the TSP by grouping cities into sets called districts. GTSP can then be described as finding the shortest tour of length s, visiting exactly one city in each of the s districts. In our formulation of the graph G, each vertex has a word sequence associated with it and the districts are defined by the first word in the sequence. This means that each word appears exactly once in the solution to the GATSP, ensuring that the solution is a valid permutation. A general example graph with districts for $N = 3$ and a bag of words of size 3 is shown in Figure 2.

This is formally defined as follows. For a bag of words S we define a set of symbols X as before. For a general N-gram language model, $N > 2$, and a set of n symbols X, $|X| = n$, we define an n-partite directed weighted graph $G = (V, E)$, with the set of vertices defined as:

$$\begin{aligned} V = \{w_i | w_i[j] \in X \text{ for } 1 \leq j \leq N\text{-}1, \\ w_i[j] \neq w_i[k] \text{ for } 1 \leq j < k < N\} \end{aligned} \tag{3}$$

Each vertex therefore represents a sequence of symbols $w_i[1..N\text{-}1]$ of length $N - 1$ from the set X, and the symbols occurring in the sequence do not repeat themselves. The set of vertices V is partitioned into n disjoint independent subsets, $V_i = \{w_j | w_j \in V, w_j[1] = i\}$, based on the first word in the word sequence, $w_j[1]$.

Let the set of edges E be a set of ordered pairs of vertices (w_i, w_j), such that:

$$\begin{aligned} E = \{(w_i, w_j) | w_i \in V_k, w_j \in V_l, k \neq l, \\ w_i[2..N\text{-}1] = w_j[1..N\text{-}2], \\ w_i[1] \neq w_j[N\text{-}1]\} \end{aligned} \tag{4}$$

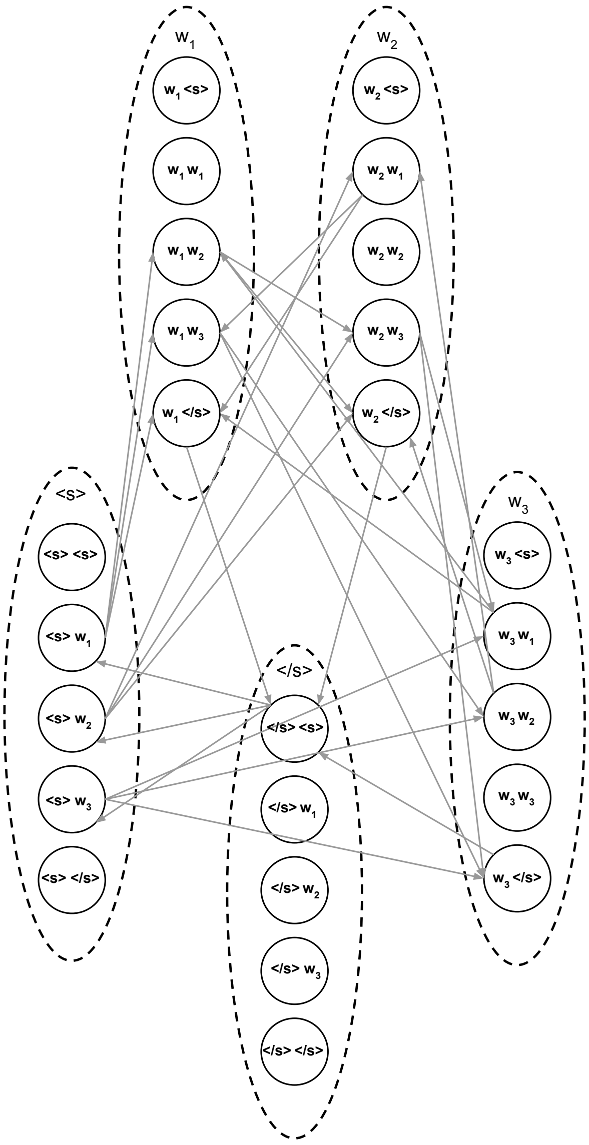


Figure 2: A general example graph for a bag of words of size 3 using a trigram language model. The graph consists of $s = 5$ districts. The vertices are assigned to a district based on the first word of the word sequence associated with the vertex. Two vertices together form a word sequence of three words. The cost of the edge between them equals the conditional probability of the final word given the context of the first two words and is provided by the trigram language model. Only edges with non-infinite cost are shown in the graph. Finding the shortest tour of length s, visiting each district exactly once, equals finding the sentence permutation with the highest probability.

$$c_{ij} = \begin{cases} 0 & \text{if } w_i[N\text{-}1] = \texttt{</s>} \text{ and } w_j[N\text{-}1] = \texttt{<s>}, \\ -\log P(w_j[N\text{-}1] | w_i[1..N\text{-}1]) & \text{if } w_i[k] \in S, 2 \leq k \leq N\text{-}2 \\ & \text{ and } w_i[1] \neq \texttt{</s>} \text{ and } w_j[N\text{-}1] \neq \texttt{<s>}, \\ -\log P(w_j[N\text{-}1] | w_i[x..N\text{-}1]) & \text{if } x \geq 2 \text{ and } w_i[x] = \texttt{<s>} \\ & \text{ and } w_i[x\text{-}1] = \texttt{</s>} \\ \infty & \text{otherwise.} \end{cases} \tag{5}$$

An edge therefore exists between two vertices if they are parts of two different subsets of V (have different first word in the sequence), have a matching subsequence, and the words outside the matching subsequence do not repeat between the two vertices. The edge cost is defined in Equation 5.

The conditional log probabilities of the form $\log P(w_j[N\text{-}1]|w_i[1..N\text{-}1])$ are computed by an N-gram language model. Consequently, finding the highest probability permutation under an N-gram language model equals solving the Generalized Asymmetric Travelling Salesman Problem (GATSP) modelled by graph G.

An important condition for an edge to exist between two vertices is that the word subsequences associated with the vertices match. If two vertices match, they form a word sequence of length N. The conditional log probability of the last word in the sequence given the previous $N - 1$ words equals the cost of the edge between the vertices.

An additional condition for an edge to exist between two vertices is that the words outside of the required matching subsequence do not repeat between the vertices. For example, two sequences 1 2 3 4 and 2 3 4 1 match according to the condition described above, but outside the required matching subsequence (2 3 4), word 1 appears twice which produces an invalid permutation.

The size of the vertex and edge set of the graph $G = (V, E)$ grows with the size n of the bag of words S and the order N of the N-gram language model. The size of the set of vertices V in the graph is $|V| = (n+2)^{N-1}$ for all values of N. The size of the set of edges E (including the infinite cost edges between the full set of vertices) is $|E| = |V|^2 = (n + 2)^{2N-2}$.

3 Implementation

The graph-based approach represents the problem of finding the sentence permutation with the highest probability as an instance of the TSP. Using a bigram language model, the problem equals solving the Asymmetric TSP. Using a higher order N-gram language model, the problem equals solving the Generalized Asymmetric TSP.

Both variations of the TSP are not as widely studied as the basic TSP and fewer algorithms exist for solving them. Transforming the variations of TSP to basic TSP is a solved problem that enables us to use state-of-the-art algorithms for solving large problem instances of the TSP. We decided to use the Concorde TSP Solver (Applegate et al., 2006a), which is prominent for continuously increasing the size of the largest optimally solved TSP instance over the last two decades.

Alternatively, a heuristic algorithm for solving the TSP can be used. Heuristic algorithms do not guarantee finding the optimal solution, but attempt to find the best possible solution given a time constraint. We used LKH as the heuristic TSP solver, which is an effective implementation of the Lin-Kernighan heuristic (Helsgaun, 2000). It currently holds the record for many large TSP instances with unknown optima.

The use of a TSP solver makes it necessary to transform the instances of ATSP and GATSP into regular TSP instances. We applied two graph transformations as necessary: (1) GATSP to ATSP transformation described by Dimitrijević and Šarić (1997) and (2) ATSP to TSP transformation described by Jonker and Volgenant (1983). These transformations allow application of the general TSP solver, although they each double the number of vertices in the graph.

Table 2 shows the total size of the vertex set after applying the transformations.

LM	5	10	15	20
2-gram	14	24	34	44
3-gram	196	576	1,156	1,936
4-gram	1,372	6,912	19,652	42,592

Table 2: The vertex set size after applying the transformations for several N-gram language models at increasing sentence length.

4 Evaluation

We evaluated three different versions of the graph-based approach based on 2-gram, 3-gram, and 4-gram language models. We evaluated each version of the system on three datasets of news sentences by computing the dataset-wide BLEU scores.

4.1 Language models

For the experimental evaluation of our graph-based approach we used 2-gram, 3-gram, and 4-gram language models with interpolated modified Kneser-Ney smoothing (Chen and Goodman, 1998). They were built using the SRI Language Modeling Toolkit (Stolcke, 2002) and KenLM Language Model Toolkit (Heafield, 2011).

The language models were estimated on the English Gigaword collection (V2, AFP and XIN parts) and the NIST OpenMT12 Evaluation Dataset (target sides of parallel data for Ar-Eng and Ch-Eng tasks).

The total size of the corpus for estimating the language models was 1.16 billion words.

4.2 Evaluation metric

The BLEU evaluation metric was developed by Papineni et al. (2002) as an inexpensive and fast method of measuring incremental progress of SMT systems. BLEU measures closeness of a candidate translation to a reference translation using N-gram precision. Similarly, in the string regeneration problem we measure the closeness of the regenerated sentence to the original sentence. We used the case insensitive NIST BLEU script v13 against tokenized references to compute the BLEU scores.

Espinosa et al. (2010) have investigated the use of various automatic evaluation metrics to measure the quality of NLG output. They found that BLEU correlates moderately well with human judgements of fluency and that it is useful for evaluation of NLG output, but should be used with caution, especially when comparing different systems. As the string regeneration problem is a basic form of NLG, BLEU is an appropriate measure of the system's performance with regards to fluency of the output. We provide examples of output and conduct a manual evaluation to confirm that the BLEU scores of individual systems reflect actual changes in output quality.

4.3 Automatic Evaluation

We evaluated the graph-based approach on three datasets:

MT08 The target side of the Ar-Eng newswire part of the NIST OpenMT08.

MT09 The target side of the Ar-Eng newswire part of the NIST OpenMT09.

SR11 The plain text news dataset of the Surface Realisation Task at GenChal'11.

The MT08, MT09, and SR11 datasets contain 813, 586, and 2398 sentences respectively.

We have taken preprocessing steps to chop long sentences into manageable parts, which is a common practice in translation. Based on preliminary experiments we decided to limit the maximum length of the chopped sentence to 20 words. N-gram models cannot be used to model sentences shorter than N words in this approach. In order to make the models comparable we ignored short sentences containing 4 or fewer words. Each chopped sentence was regenerated separately and the regenerated chopped sentences were concatenated to form the original number of dataset sentences. We expect that the preprocessing steps increased the reported BLEU scores to a certain degree. However, all systems compared in the experimental evaluation were subject to the same conditions and their scores are therefore comparable.

The graphs constructed under a 4-gram language model are too large to solve optimally in reasonable time (i.e. under half an hour per sentence). Because of this, we employ two approaches to regenerate long sentences with the 4-gram language model: (1) Use the LKH heuristic algorithm with a set time limit, and (2) back-off to the trigram language model. We refer the reader to Horvat (2013) for details.

The BLEU scores for the four systems are reported in Table 3. The 3-gram graph-based approach performed considerably better than the 2-gram approach, increasing the BLEU score for 10 BLEU points or more on all three datasets. The 4-gram approach augmented with a heuristic TSP solver performed significantly worse than the 3-gram approach on MT08 and MT09 datasets, while performing better on SR11 dataset. The reason for this difference is the different distribution of chopped sentence lengths between the three datasets. Around one fourth of all chopped

LM	Solver	MT08	MT09	SR11
2g	opt	44.4	45.1	40.6
3g	opt	57.9	58.0	50.2
4g	opt +heur	44.8	42.6	51.7
4g +3g	opt	59.1	59.5	51.8

Table 3: BLEU scores for four versions of the graph-based approach, based on 2-gram, 3-gram, and 4-gram language models. We used the 4-gram approach on sentences of up to length 18. The remaining sentences were computed using either a heuristic TSP solver (opt +heur) or by backing-off to a 3-gram approach (4g +3g).

sentences in MT08 and MT09 datasets are longer than 18 words. On the other hand, less than 1% of chopped sentences of the SR11 dataset are longer than 18 words. This means that significant parts of the MT08 and MT09 datasets were solved using the heuristic approach, compared to a small part of the SR11 dataset. Using a heuristic TSP solver therefore clearly negatively affects the performance of the system. The 4-gram approach backing-off to the 3-gram approach achieved a higher BLEU score than the 3-gram approach over all datasets.

In Figure 3 we show examples of regenerated sentences for three versions of the system. In the first example, we can see the improvements in the output fluency with better versions of the system. The improvements are reflected by the BLEU scores. The 4-gram output can be considered completely fluent. However, when compared to the original sentence, its BLEU score is not 100, due to the fact that the number of people killed and people injured are switched. In this regard, BLEU score is harsh and not an ideal evaluation metric for the task. In the second example, the original sentence contains complicated wording which is reflected in poor performance of all three versions of the system, despite the high BLEU score of the 3-gram system. In the final example, we can observe the gradual improvement of fluency over the three versions of the system. This is reflected by the BLEU score, which reaches 100.0 for the 4-gram system, which produced an identical sentence to the original.

4.4 Manual Evaluation

We manually evaluated three versions of the graph-based approach: 2-gram, 3-gram, and 4-gram system using 3-gram as back-off. We conducted a pairwise comparison of the three systems: for each evaluation sentence, we compared the output of a pair of systems and asked which output is more fluent.

We used the crowdsourcing website CrowdFlower[1] to gather fluency judgments. Judges were asked ‘Please read both sentences and compare the fluency of sentence 1 and sentence 2.’ They were given three options: ‘Sentence 1 is more fluent’, ‘Sentence 2 is more fluent’, ‘Sentence 1 and Sentence 2 are indistinguishable in fluency’. The order of presentation of the two systems was randomized for each sentence.

100 sentences of length between 5 and 18 words were chosen randomly from the combined MT08 and MT09 dataset. We gathered 5 judgements for each sentence of a single pairwise comparison of two systems. Each pairwise comparison of two systems is therefore based on 500 human judgements.

The platform measures the reliability of judges by randomly posing gold standard questions in between regular questions. If any judge incorrectly answered a number of gold standard questions, their judgements were deemed unreliable and were not used in the final result set. A thorough discussion of suitability and reliability of crowdsourcing for NLP and SMT tasks and related ethical concerns can be found in: Snow et al. (2008), Zaidan and Callison-Burch (2011), and Fort et al. (2011).

The pairwise comparison results are shown in Table 4. Each number represents the proportion of the human judgements that rated the output of the row system as better than the column system. The raw numbers of pairwise comparison judgements in favor of each system are shown in Table 5. A one-sided sign test indicated that we can reject the null hypothesis of the two systems being equal in favor of the alternative hypothesis of the first system being better than the second for all three system pairings: 3g and 2g, 4g and 2g, and 4g and 3g, $p < 0.001$ for all three comparisons. The manual evaluation results therefore confirm the BLEU score differences between the three graph-based systems.

Interestingly, in automatic evaluation the difference in BLEU scores between 2g and 3g systems was much bigger (around 10 BLEU points) than

[1] http://crowdflower.com/

	Hypothesis	**BLEU**
1. REF	meanwhile , azim stated that 10 people were killed and 94 injured in yesterday 's clashes .	
(a)	meanwhile , azim and 10 people were injured in clashes yesterday 's stated that killed 94 .	21.4
(b)	azim , meanwhile stated that 94 people were killed and 10 injured in yesterday 's clashes .	50.4
(c)	meanwhile , azim stated that 94 people were killed and 10 injured in yesterday 's clashes .	66.3
2. REF	zinni indicated in this regard that president mubarak wants egypt to work with the west .	
(a)	egypt wants zinni in this regard to work with president mubarak indicated that the west .	24.9
(b)	zinni wants egypt to work with the west that president mubarak indicated in this regard .	63.4
(c)	work with zinni indicated that president mubarak wants the west to egypt in this regard .	30.6
3. REF	he stressed that this direction is taking place in all major cities of the world .	
(a)	he stressed that the world is taking place in this direction of all major cities .	33.9
(b)	he stressed that all major cities of the world is taking place in this direction .	58.0
(c)	he stressed that this direction is taking place in all major cities of the world .	100.0

Figure 3: Output examples of three versions of the graph-based approach: (a) 2-gram, (b) 3-gram, and (c) 4-gram with 3-gram back-off. The original sentence is given for each of the three examples. Sentence BLEU scores are shown for each regenerated sentence.

LM	2g	3g	4g
2g	-	-	-
3g	65.4	-	-
4g	72.9	69.2	-

Table 4: Manual evaluation results of pairwise comparison between three versions of the system: 2-gram, 3-gram, and the 4-gram system with 3-gram back-off. The numbers represent the percentage of judgements in favor of the row system when paired with the column system.

the difference between 3g and 4g systems (around 1 BLEU point). However, in manual evaluation the difference between 3g and 4g systems is noticeably bigger (69.2%) than the difference between 2g and 3g systems (65.4%).

sys1	**sys2**	**sys1**	**equal**	**sys2**	**Total**
2g	3g	124	142	234	500
2g	4g	102	124	274	500
3g	4g	92	201	207	500

Table 5: The raw numbers of pairwise comparison judgements between the three systems. The columns give the number of judgements in favor of each of the three options.

5 Related Work

The basic task of all natural language realization approaches is to take a meaning representation as input and generate human-readable output. The approaches differ on how much information is required from the meaning representation. Deep representation include dependency graphs annotated with semantic labels and other syntactic information (Belz et al., 2011). Shallow representations include syntactic dependency trees annotated with POS tags and other syntactic information (Belz et al., 2011), IDL-expressions (Soricut and Marcu, 2005), and Abstract Meaning Representation (Langkilde and Knight, 1998).

Soricut and Marcu (2005) consider NLG in context of other popular natural language applications, such as Machine Translation, Summarization, and Question Answering. They view these as text-to-text applications that produce textual output from textual input. Because of this, many natural language applications need to include some form of natural language generation to produce the output text. However, the natural language generation in these applications is often handled in an application-specific way. They propose to use IDL-expressions as an application-independent representation language for text-to-text NLG. The IDL-expressions are created from strings using operators to combine them. The authors evaluate their approach on the string regeneration task and achieve moderate BLEU scores.

Wan et al. (2009) approach the string regeneration problem using dependency spanning trees. Their approach is to search for the most probable dependency tree containing each word in the input or, equally, finding the optimal spanning tree. Zhang and Clark (2011) propose a similar approach using Combinatory Categorial Grammar (CCG) which imposes stronger category constraints on the parse structure compared to dependency trees investigated by Wan et al. (2009). They primarily focus on the search problem of finding an optimal parse tree among all possible trees containing any choice and ordering of the input words. The CCG approach achieved higher BLEU scores compared to the approach proposed by Wan et al. (2009). Zhang et al. (2012) improve the CCG approach by Zhang and Clark (2011) by incorporating an N-gram language model. de Gispert et al. (2014) present a similar N-gram language model approach to ours with a different decoder that does not guarantee optimal results. In their comparison with approach by Zhang et al. (2012) they report gains of more than 20 BLEU points.

The purpose of studying and building approaches to solving the string regeneration problem is to improve grammaticality and fluency of machine generated text. An approach using a TSP reordering model by Visweswariah et al. (2011) focused on the preordering task in SMT. In the preordering task the words of the source sentence are reordered to reflect the word order expected in the target sentence which helps improve the performance of the SMT system.

6 Conclusions and Future Work

In the paper we explored the N-gram language model approach to the string regeneration problem of recovering a fluent version of the original sentence given a bag of words. The N-gram language model approach computes the highest probability permutation of the input bag of words under an N-gram language model. We described a graph-based approach for finding the optimal permutation. Finding the permutation with the highest probability in the graph formulation is equal to finding the shortest tour in the graph or, equally, solving the Travelling Salesman Problem.

We evaluated the proposed approach on three datasets. The BLEU scores and example output indicated that the graph-based approach is successful in constructing a fairly fluent version of the original sentence. The 2-gram based approach performed moderately well but was surpassed by the 3-gram based approach. The 4-gram based approach offered an improvement on the 3-gram but is not of much practical use due to its long computation times. Approximate solutions computed using a heuristic TSP solver significantly reduced the quality of the output and resulting BLEU score. We confirmed the results of automatic evaluation by conducting a manual evaluation.

The BLEU scores of our approach and the approach by Wan et al. (2009) can't be directly compared as we used different evaluation datasets and preprocessing procedures. Nonetheless, the difference in BLEU scores is stark, our best system outperforming theirs by more than 20 BLEU points.

The work presented in this paper can be extended in a number of ways. More extensive comparison between optimal and approximate approaches would help draw stronger conclusions regarding the need for optimality. A direct comparison between our N-gram language model based approach and approaches presented by Wan et al. (2009), Zhang et al. (2012), and others is needed to determine its performance relative to other approaches.

The graph-based approach itself can be extended in a number of ways. Emulating methods from Statistical Machine Translation, the approach could be extended to generate an N-best list of reorderings. A different method could then be used to rerank the N-best list to choose the best one. The methods can range from rescoring the outputs with a higher-order language model or a dependency language model, to using discriminative machine learning. The approach could also be extended to handle additional constraints in the input, such as phrases instead of words, by modifying the edge weights of the graph.

Another interesting area of future research relating to the wider string regeneration problem is determining the human performance on the task. Based on a simple trial of trying to regenerate a long sentence by hand, it is clear that human performance on the task would not equal 100 BLEU points. It would therefore be interesting to determine the human performance on the string regeneration problem to provide a contrast and a point of comparison to the performance of machine systems.

Finally, the string regeneration problem can be viewed as a constraint satisfaction approach where the constraints are minimal. However, in many instances there is more information available regarding the final output of a system, for example syntactic or semantic relationship between words. This information introduces additional constraints to the simple bag of words that need to be included in the output. In future, we will explore methods of generating from a set of constraints in a robust manner to produce output that is fluent and grammatical.

Acknowledgments

We thank Juan Pino for his help with language modeling and experiments. We thank Ann Copestake and the anonymous reviewers for their discussions and suggestions.

Matic Horvat was supported by the Ad Futura Scholarship during his work on the MPhil thesis.

References

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. 2006a. Concorde TSP Solver.

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. 2006b. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

David L. Applegate, Robert E. Bixby, Vašek Chvátal, William Cook, Daniel G. Espinoza, Marcos Goycoolea, and Keld Helsgaun. 2009. Certification of an optimal TSP tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15, January.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The First Surface Realisation Shared Task : Overview and Evaluation Results. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, volume 2, pages 217–226.

Stanley F. Chen and Joshua T. Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical report, Harvard University.

Adrià de Gispert, Marcus Tomalin, and William Byrne. 2014. Word Ordering with Phrase-Based Grammars. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, number 2009.

Vladimir Dimitrijević and Zoran Šarić. 1997. An Efficient Transformation of the Generalized Traveling Salesman Problem into the Traveling Salesman Problem on DIgraphs. *Information Sciences*, 102:105–110.

Dominic Espinosa, Rajakrishnan Rajkumar, Michael White, and Shoshana Berleant. 2010. Further Meta-Evaluation of Broad-Coverage Surface Realization. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 564–574.

Karën Fort, Adda Gilles, and K. Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 37(2):413–420.

Kenneth Heafield. 2011. KenLM : Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, number 2009, pages 187–197.

Michael Held and Richard M. Karp. 1962. A Dynamic Programming Approach to Sequencing Problems. *Society for Industrial and Applied Mathematics*, 10(1):196–210.

Keld Helsgaun. 2000. An effective implementation of the LinKernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Matic Horvat. 2013. *A Graph-Based Approach to String Regeneration*. Ph.D. thesis.

Roy Jonker and Ton Volgenant. 1983. Transforming Asymmetric into Symmetric Traveling Salesman Problems. *Operations Research Letters*, 2(4):161–163.

Kevin Knight. 2007. Automatic language translation generation help needs badly. In *MT Summit XI Work- shop on Using Corpora for NLG: Keynote Address*, pages 5–8.

Irene Langkilde and Kevin Knight. 1998. Generation that Exploits Corpus-Based Statistical Knowledge. In *Proceedings of the 17th international conference on Computational linguistics*, pages 704–710.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. 2002. BLEU : a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, number July, pages 311–318, Philadelphia.

Rion Snow, Brendan O Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and Fast But is it Good ? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, number October, pages 254–263.

Radu Soricut and Daniel Marcu. 2005. Towards Developing Generation Algorithms for Text-to-Text Applications. In *Proceedings of the 43rd Annual Meeting of the ACL*, number June, pages 66–74, Ann Arbor.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904.

Karthik Visweswariah, Rajakrishnan Rajkumar, Ankur Gandhe, Ananthakrishnan Ramanathan, and Jiri Navratil. 2011. A Word Reordering Model for Improved Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 486–496.

Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2009. Improving Grammaticality in Statistical Sentence Generation : Introducing a Dependency Spanning Tree Algorithm with an Argument Satisfaction Model. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, number April, pages 852–860.

Omar F Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229.

Yue Zhang and Stephen Clark. 2011. Syntax-Based Grammaticality Improvement using CCG and Guided Search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157.

Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-Based Word Ordering Incorporating a Large-Scale Language Model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 736–746, Avignon, France.